Diverse Data Selection Considering Data Distribution for Unsupervised Continual Learning

Naoto Hayashi[a], Naoki Okamoto[b], Tsubasa Hirakawa[c], Takayoshi Yamashita[d] and Hironobu Fujiyoshi[e]

Chubu University, 1200 Matsumoto-cho, Kasugai, Aichi, Japan

{*hayashi29, naok, hirakawa*}*@mprg.cs.chubu.ac.jp*, {*takayoshi, fujiyoshi*}*@isc.chubu.ac.jp*

Keywords: Deep Learning, Image Classification, Self-Supervised Learning, Continual Learning.

Abstract: In continual learning, the train data changes during the learning process, making it difficult to solve previously learned tasks as the model adapts to the new task data. Many methods have been proposed to prevent catastrophic forgetting in continual learning. To overcome this problem, Lifelong Unsupervised Mixup (LUMP) has been proposed, which is capable of learning unlabeled data that can be acquired in the real world. LUMP trains a model by self-supervised learning method, and prevents catastrophic forgetting by using a mixup of a data augmentation method and a replay buffer that stores a part of the data used to train previous tasks. However, LUMP randomly selects data to store in the replay buffer from the train data, which may bias the stored data and cause the model to specialize in some data. Therefore, we propose a method for selecting data to be stored in the replay buffer for unsupervised continuous learning method.The proposed method splits the distribution of train data into multiple clusters using the k-means clustering. Next, one piece of data is selected from each cluster. The data selected by the proposed method preserves the distribution of the original data, making it more useful for self-supervised learning.

1 INTRODUCTION

With the spread of the Internet of Things, it is expected that a vast amount of data accumulated online can be used for learning deep learning models such as Deep Convolutional Neural Networks (LeCun et al., 1998). On the other hand, it is difficult to retain all the acquired data while learning due to hardware capacity issues. To solve this problem, there is a method called continual learning (Thrun, 1995), in which the data used for learning is discarded as it is learned. In continual learning, the train data changes during the learning process, causing the model to adapt to the data of the new task. This causes a problem called catastrophic forgetting, which makes previously learned tasks difficult to solve. Continual learning uses a variety of devices to prevent catastrophic forgetting. There are three main ways to devise continual learning. The first is an architecture-based method that devises a network structure. The second is a regularization-based method that suppresses catastrophic forgetting by using regularization. The third is a replay-based method that stores a part of the data used in learning a previous task and reuses the data when learning a new task. Many of the continual learning methods proposed until now are supervised continual learning methods that assume learning only on labeled data. However, there is only a small amount of labeled data in the real world.

Therefore, to use a large amount of real-world data, learning using unlabeled data needs to be assumed. To solve this problem, Lifelong Unsupervised Mixup (LUMP) (Madaan et al., 2022) has been proposed, which can learn unlabeled data. LUMP learns by self-supervised learning using only unlabeled data and prevents catastrophic forgetting using a data augmentation method mixup (Zhang et al., 2017) and a replay buffer that stores a part of data used to learn previous tasks. LUMP randomly selects data to store in the replay buffer from the train data. If data is selected randomly, the data to store may be biased, resulting in a model that is specialized to particular data.

Therefore, we propose a method for improving the data selection to store in the replay buffer to in-

[a] https://orcid.org/0009-0000-6909-3346
[b] https://orcid.org/0000-0003-0053-8890
[c] https://orcid.org/0000-0003-3851-5221
[d] https://orcid.org/0000-0003-2631-9856
[e] https://orcid.org/0000-0001-7391-4725

crease accuracy and suppress catastrophic forgetting in unsupervised continual learning. The proposed method splits the distribution of train data into multiple clusters using the k-means clustering (Hartigan and Wong, 1979). Next, a single data is selected from each cluster. The data to store in the replay buffer by using the proposed method is shown in Figure 1. As shown in Figure 1, the data selected by the proposed method preserves the distribution of the original data and has generalizability, making it more useful for self-supervised learning. Our experimental results demonstrate the effectiveness of the proposed method by comparing quantitative and qualitative evaluations of the proposed and conventional methods.

In summary, our contributions are as follows:

- We propose a data selection method using the k-means clustering. From the qualitative evaluation, we found that the proposed method selects diverse data without bias in the latent space.
- This research demonstrates that keeping diverse data in the replay buffer improves classification accuracy and the forgetting rate and shows that the selection of diverse data has a positive effect on self-supervised learning.

2 RELATED WORK

This chapter introduces self-supervised learning and continual learning.

2.1 Self-Supervised Learning

Self-supervised learning is a pre-train method using large amounts of unlabeled data. Contrastive learning (Ye et al., 2019; He et al., 2020; Li et al., 2021; Zbontar et al., 2021; Zhang et al., 2022) and non-contrastive learning without negative pairs (Grill et al., 2020; Caron et al., 2020) are typical Self-supervised learning methods. Self-supervised learning requires learning on more diverse data to create a general-purpose model.

SimCLR (Chen et al., 2020) is a typical contrastive learning method. In this method, the data in the batch that is not one's own data is used as the negative sample, and the data augmented to one's own data is input to the network as the positive sample, and the network learns to separate the negative sample from one's own data in the latent space and to make the positive sample close to one's own data. A feature of SimCLR is that larger batch sizes increase negative sample variation and improve performance.

SimSiam (Chen and He, 2021) is a typical non-contrastive learning method. This method applies dif-

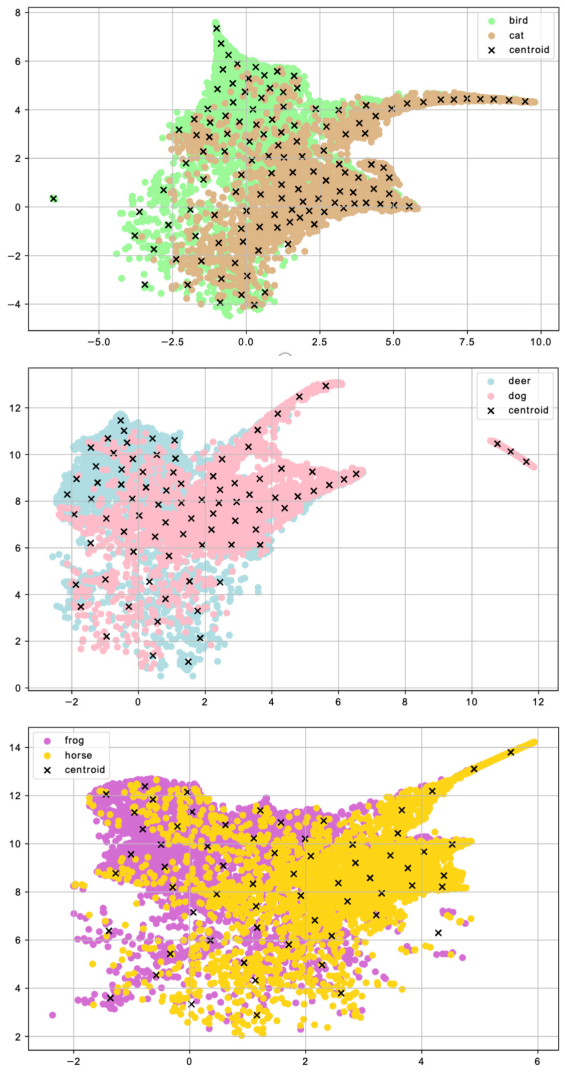

Figure 1: Data selected by the proposed method. In practice, the proposed method selects the data with the maximum for each centroid and cosine similarity.

ferent data augmentations to a single piece of data to produce two pieces of data. The two pieces of generated data are input to the network. The network learns to increase the similarity of the two feature vectors output from the network. In this method, stop-grad is introduced to solve the problem of collapse, where any data input to the network produces the same output to reduce losses. A feature of SimSiam is that it does not require negative samples, and its network structure and loss calculations are simpler than those of the contrastive learning method.

This research uses SimSiam, which can be trained simply as self-supervised learning.

2.2 Continual Learning

Continual learning is a method of learning by obtaining new data at regular intervals. In continual learning, the train data changes during the learning process. This causes a phenomenon called catastrophic forgetting, in which the model adapts to the new task data, making previously learned tasks difficult to solve. To prevent catastrophic forgetting, several continual learning methods have been proposed, such as architecture-based methods (Mallya and Lazebnik, 2018; Yoon et al., 2018) that devise network structures, regularization-based methods (Li and Hoiem, 2016; Kirkpatrick et al., 2017; Chaudhry et al., 2018), and replay-based methods (Shin et al., 2017; Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019; Aljundi et al., 2019; Buzzega et al., 2020; Purushwalkam et al., 2022; Wang et al., 2022; Tiwari et al., 2022; Lin et al., 2023) that reproduce datasets used in previous learning when learning new data.

Progressive Neural Networks (PNN) (Rusu et al., 2016) is one of the architecture-based methods. PNN prepares a network for each task and trains each task in turn. For the second and later tasks, in addition to the usual vertical connection of the layers of the network, a horizontal connection is made with the layer of the network where the previous task was learned, so that the representation of the previously learned task is reflected in the latest representation. The process in PNN is as follows:

$$
\begin{aligned}
h_i^{(k)} &= f(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}) \\
f(x) &= \max(0, x)
\end{aligned}
\tag{1}
$$

where $W_i^{(k)}$ is the vertically oriented layer weights from $i-1$ to i layers of task k, $h_{i-1}^{(k)}$ is the output value of $i-1$ layer of task k, $U_i^{(k:j)}$ is the horizontally oriented layer weights from $i-1$ layer of task j to i layer of task k, and $h_{i-1}^{(j)}$ is the output value of the $i-1$-layer of task j.

Synaptic Intelligence (SI) (Zenke et al., 2017) is one of the regularization-based methods. SI uses three-dimensional weights and a measure of importance to control catastrophic forgetting. When using one-dimensional weights, the network easily paints over the knowledge learned in the previous task as it learns the next task. Therefore, this method mimics the human brain and prevents catastrophic forgetting by using three-dimensional weights to fix knowledge. Importance is a measure of how much each weight influences the change in the loss function. The importance is calculated as follows:

$$
\begin{aligned}
L(w(t^u)) - L(w(t^{u-1})) &\approx \sum_k \int_{t^{u-1}}^{t^u} g_k(w(t)) w_k'(t) dt \\
&\approx \sum_k p_k^u
\end{aligned}
\tag{2}
$$

where t^u is the time when task u is learned, t^{u-1} is the time when task $u-1$ is learned, $w(t^u)$ is the weight when task u is learned, and $w(t^{u-1})$ is the weight when task $u-1$ is learned. In addition, g_k is calculated by $\frac{\partial L}{\partial w_k}$ and w_k' by $\frac{\partial w_k}{\partial t}$. By stopping the updating of weights of high importance in past tasks as a penalty, catastrophic forgetting can be prevented.

Many previous methods of continual learning assumed learning on labeled data. However, most data available in the real world is unlabeled data, so it cannot be used as data for traditional continual learning methods. To solve this problem, Lifelong Unsupervised Mixup (LUMP) has been proposed, which performs continual learning by self-supervised learning using unlabeled data. This method is a replay-based unsupervised continual learning method that prevents catastrophic forgetting by storing some of the data from previously learned tasks in a replay buffer and using that data when learning new tasks. When data from past tasks is used, it is combined with data from new tasks using a mixup of data augmentation methods, and the data are then used in self-supervised learning to prevent catastrophic forgetting.

There are task-incremental, class-incremental, and domain-incremental scenarios as problem settings for continual learning. This research focuses on how to select data to be stored in the replay buffer in task-incremental and class-incremental problem settings.

3 PROPOSED METHOD

This chapter describes the proposed method. In this paper, we investigate the data to be stored in the replay buffer in LUMP, which is a conventional continual learning method, and proposes an improved method for selecting the data to be stored. Our proposed method uses the k-means clustering for data selection and the selection of data to be deleted from the replay buffer.

3.1 Preliminary Survey

Before going to the details of the proposed method, we investigate the data to be selected by LUMP. As mentioned before, LUMP randomly selects data from

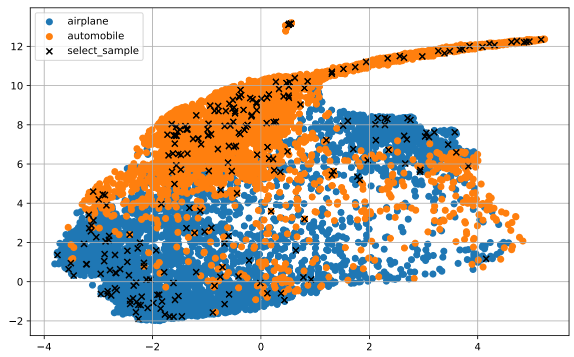

Figure 2: Data selected by LUMP. The data marked with a cross in the figure is selected and stored in the replay buffer.

the data in the batch when storing data in the replay buffer. Figure 2 shows the distribution of data selected by LUMP. This figure shows that when data is selected randomly, a large amount of data is selected from areas where the density of the original train data is high, and only a small amount of data is selected from areas where the density of the train data is low. This suggests that when data is randomly selected, the data in the replay buffer becomes biased and loses generalizability, preventing the acquisition of a variety of features. Therefore, we propose a continual learning method that introduces a method for selecting data so that the data to be stored in the replay buffer is more diverse.

3.2 Diverse Data Selection for Unsupervised Continual Learning

We propose an unsupervised continual learning method that takes into consideration the data distribution of the train data by introducing the k-means clustering and performs various data selections. The flow of the proposed method is shown in Algorithm 1. The flow of the first task is shown in Figure 3, and the flow of the second and subsequent tasks is shown in Figure 4.

In the first task (see Fig. 3), after learning with the train data, the data to be stored in the replay buffer is selected. The centroid of the clusters obtained by the k-means clustering is scattered throughout the data distribution, and by selecting the closest data from the centroids of each cluster. It is possible to consider the original data distribution and learn diverse data, which is expected to improve accuracy and suppress catastrophic forgetting in the classification task.

In the second and subsequent tasks (see Fig. 4), the train data and the data in the replay buffer are combined by mixup, and the data is used for learning. After that, the data to be deleted from the replay

```
Data: Total number of tasks $T$, Size of replay
      buffer $B$, Data of the new task $x_i$, Data
      in the replay buffer $x_j$, Number of data
      to be stored in replay buffer $n$
for $t \leftarrow 1$ to $T$ do
    $n \longleftarrow \frac{B}{t}$;
    if $t = 1$ then
        Self-supervised learning with
          SimSiam($x_i$);
    else
        $x \longleftarrow$ Combine $x_i$ and $x_j$ by mixup;
        Self-supervised learning with
          SimSiam($x$);
        Select $B - n$ pieces of data to be
          deleted from up to $t - 1$ task in the
          replay buffer;
    end
    Select $n$ pieces of data to store in the
      replay buffer by k-means clustering
      from the data in task $t$;
end
```

Algorithm 1: The proposed method.

buffer is selected, and the data for a new task is selected to be stored in the free replay buffer space.

3.2.1 Learning Method

Learning of the proposed method is based on the LUMP setting. The first task is self-supervised learning using train data. The second and subsequent tasks use mixup of a data augmentation method to combine data from previous tasks already stored in the replay buffer with train data for the new task, and the combined data is used for self-supervised learning. The method of generating data X by mixup is as follows:

$$X = \lambda X_i + (1 - \lambda) X_j \tag{3}$$

where X_i is the train data for the current task, X_j is the data from the previous task in the replay buffer, and λ is the hyperparameter that determines the mixing ratio.

3.2.2 Data Selection by k-means Clustering

When selecting data to be stored in the replay buffer, the train data for each task is input to the encoder after learning is completed and features are acquired. Clustering is performed by applying the k-means clustering to the acquired features. The number of clusters set by the k-means clustering is the number of data to be stored in the replay buffer. Next, the data corresponding to the feature with the highest cosine similarity from the centroid of each cluster is selected and

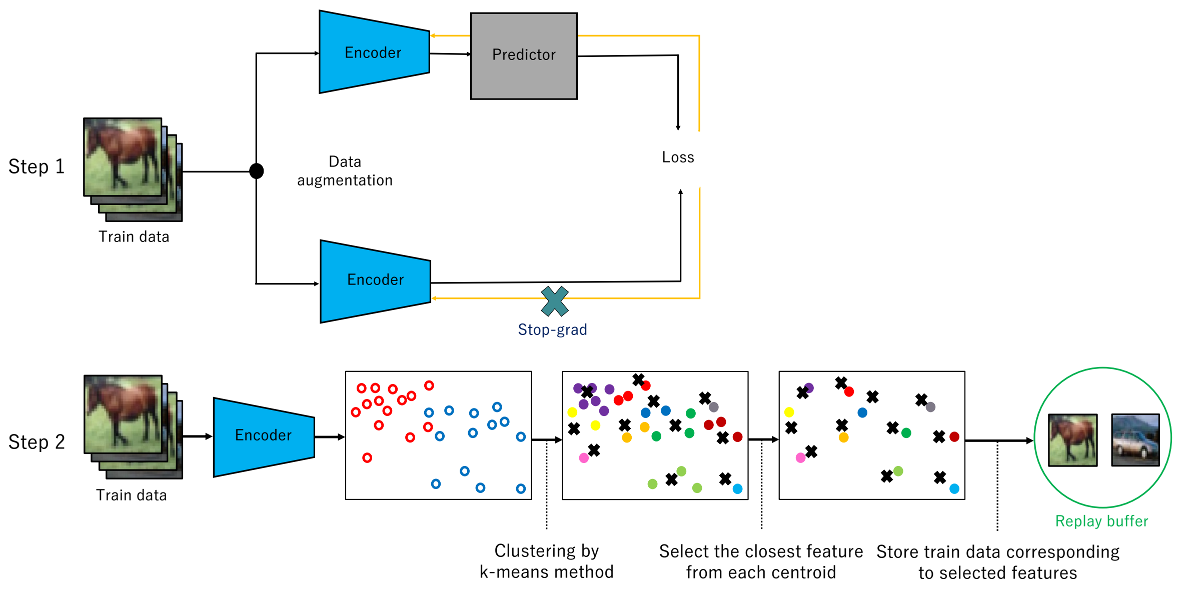
Step 1
Train data
Encoder
Predictor
Data augmentation
Loss
Encoder
Stop-grad
Step 2
Train data
Encoder
Replay buffer
Clustering by k-means method
Select the closest feature from each centroid
Store train data corresponding to selected features

Figure 3: Flow of the first task of the proposed method. In the first task, after learning with the train data, the data to be stored in the replay buffer is selected.

stored in the replay buffer. The number n of data to be stored in the replay buffer is calculated as follows:

$$n = \frac{B}{t_i} \tag{4}$$

where B is the buffer size, t_i is the task id, $t_i = 1$ for the first task, and any decimal point value is truncated so that n is a natural number. Before storing the data for each task in the replay buffer, the data in the current replay buffer is input to the encoder to acquire features and select the data to be left in the replay buffer as well as the data selection from the train data. The number of data to be left in the replay buffer is $B - n$. Finally, the replay buffer stores the data selected for the new task in the space where the deleted data existed.

4 EXPERIMENT

This chapter presents a quantitative evaluation of the effectiveness of the proposed method by comparing it with a conventional method, LUMP, using the indices of classification accuracy and oblivion rate for the class classification problem. We also visualize the data stored in the replay buffer with LUMP and the proposed method and qualitatively evaluate whether the proposed method selects data that preserves the data distribution.

4.1 Experimental Settings

In this experiment, the dataset is Split CIFAR-10 (Krizhevsky, 2009) divided into 5 tasks of 2 classes each and Split CIFAR-100 divided into 5 tasks of 20 classes each, the network is ResNet-18 (He et al., 2016), and the self-supervised learning method is SimSiam. The number of learning epochs is 200, the batch size and replay buffer size is 256 for the Split CIFAR-10 experiment, and the batch size and replay buffer size is 512 for the Split CIFAR-100 experiment.

The k-Nearest Neighbor algorithm (k-NN) is used as the evaluation method. In k-NN, the data to be estimated is plotted on the plotted coordinates of the train data, and the class with the highest number of correct labels assigned to the k train data closest to the plotted data is used as the result of estimation. The qualitative evaluation visualizes the data selected by LUMP and the data selected by the proposed method when using the model of after learning the first task of Split CIFAR-10.

4.2 Evaluation Metrics

We use the average accuracy and average forgetting rate at the end of learning all tasks as the evaluation metrics in the task-increasing problem setting. The average accuracy is the average classification accuracy for each task when using a model that has completed learning on all tasks. The average forgetting rate is the average difference between the best classifi-

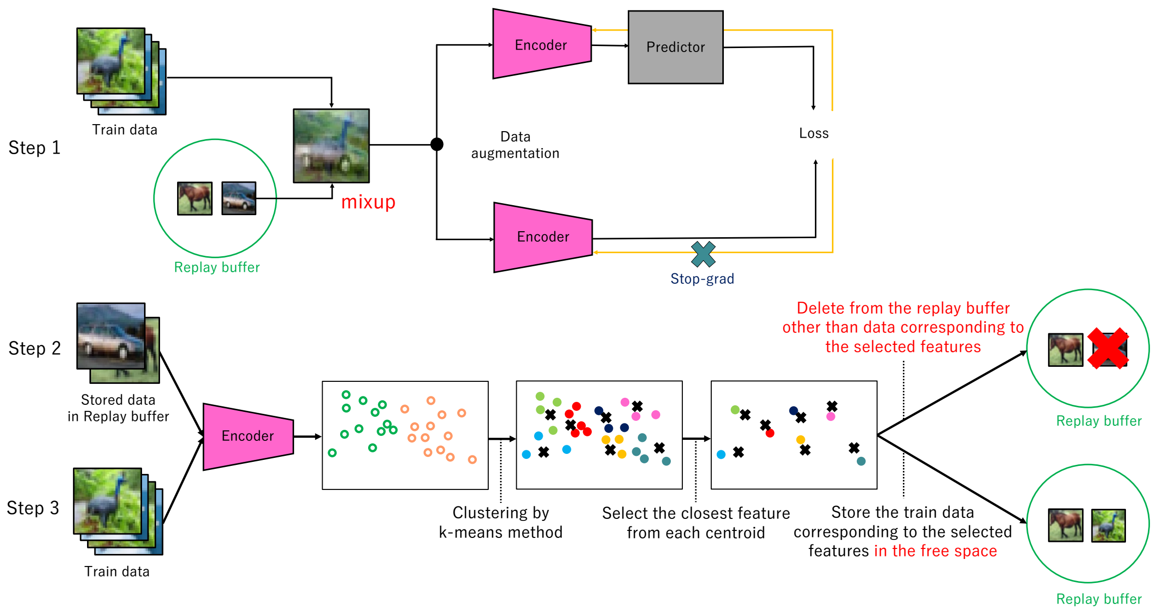
Step 1
Train data
Replay buffer
mixup
Data augmentation
Encoder
Predictor
Loss
Encoder
Stop-grad
Step 2
Stored data in Replay buffer
Step 3
Train data
Encoder
Clustering by k-means method
Select the closest feature from each centroid
Store the train data corresponding to the selected features in the free space
Delete from the replay buffer other than data corresponding to the selected features
Replay buffer
Replay buffer

Figure 4: Flow of the second and subsequent tasks of the proposed method. In the second and subsequent tasks, the train data and the data in the replay buffer are combined by mixup, and the data is used for learning. After that, the data to be deleted from the replay buffer is selected, and the data for a new task is selected to be stored in the free replay buffer space.

cation accuracy for each task and the classification accuracy when using a model that has completed learning on all tasks. The average accuracy and average forgetting rate are calculated as follows:

$$A_\tau = \frac{1}{\tau}\sum_{i=1}^{\tau} a_{\tau,i} \tag{5}$$

$$F = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{\tau\in(1,\dots,T)} (a_{\tau,i} - a_{T,i}) \tag{6}$$

where T is the number of tasks, $a_{\tau,i}$ is the accuracy of learning on task τ and evaluating on task i, and $a_{T,i}$ is the accuracy of learning on the last task T and evaluating on task i. This experiment uses $\tau = T$ in equation (5) and evaluates the average accuracy when all tasks have completed learning.

As an evaluation measure in the class-increasing problem setting, the classification accuracy of all learned classes when all classes have completed learning is used.

4.3 Quantitative Evaluation

Evaluation was conducted in task-incremental and class-incremental problem settings.

4.3.1 Task-Incremental Problem Setting

The average accuracy and average forgetting rate for 5 tasks of 2-class classification using Split CIFAR-10 and 5 tasks of 20-class classification using Split CIFAR-100 are shown in Table 1. The values in Table 1 are the averages of the experiments with three different seed values. In Table 1, the proposed method improves the average accuracy by 0.69 pt and the average forgetting rate by 0.12 pt compared with LUMP in the Split CIFAR-10 experiment. In the result of the Split CIFAR-100 experiment, the proposed method improves the average accuracy by 1.57 pt and the forgetting rate by 1.72 pt compared with LUMP. These results suggest that the proposed data selection method is effective.

4.3.2 Class-Incremental Problem Setting

Table 2 shows the accuracy of 10-class classification when using a model that has completed learning all of Split CIFAR-10 and the accuracy of 100-class classification when using a model that has completed learning all of Split CIFAR-100. The values in Table 1 are the averages of the experiments with three different seed values. In Table 2, the proposed method improves the accuracy by 2.42 pt for the 10-class classification and by 1.26 pt for the 100-class classification compared with LUMP. This results suggests that the proposed method is effective even in class-incremental problem setting.

4.4 Qualitative Evaluation

The data selected by LUMP and the proposed method are shown in Figures 2 and 5, respectively. The data

Table 1: Average accuracy [%] and average forgetting rate [pt] in a task-incremental problem setting.

	Average Accuracy		Average Forgetting Rate	
Method	Split CIFAR-10	Split CIFAR-100	Split CIFAR-10	Split CIFAR-100
LUMP	90.81	61.38	2.02	3.51
Ours	**91.50**	**62.95**	**1.90**	**1.79**

Table 2: Accuracy of 10-class classification using CIFAR-10 [%] and 100-class classification using CIFAR-100 [%] in a class-incremental problem setting.

	CIFAR-10	CIFAR-100
LUMP	65.41	40.02
Ours	**67.83**	**41.28**

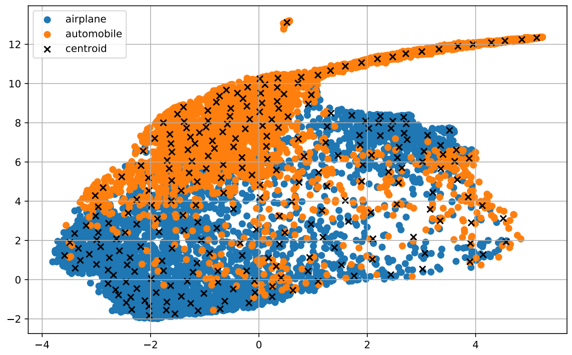

Figure 5: Data selected by the proposed method. The proposed method selects the data with the maximum for each centroid and cosine similarity.

selected by the proposed method is that with the maximum centroid and cosine similarity to each of the centroids in Figure 5. Comparing Figures 2 and 5, the proposed method can select data considering the original data distribution and store a wide diversity of data. These results suggest that the selection of diverse data is effective in improving classification accuracy and forgetting rate.

5 CONCLUSIONS

In this research, we proposed an unsupervised continual learning method using a diverse data selection method based on k-means clustering. In the experiments, we observed that the proposed method preserves the distribution of train data and selects a diversity of data. We observed an improvement in classification accuracy and forgetting rate with the proposed method. In the future, we plan to evaluate the proposed method when other self-supervised learning methods are used.

REFERENCES

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. *Neural Information Processing Systems conference*.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Neural Information Processing Systems conference*.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Neural Information Processing Systems conference*.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. S. (2018). Riemannian walk for incremental learning: Understanding forgetting and intransigence. *European Conference on Computer Vision*.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with a-GEM. *International Conference on Learning Representations*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*.

Chen, X. and He, K. (2021). Exploring simple siamese representation learning. *Conference on Computer Vision and Pattern Recognition*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Neural Information Processing Systems conference*.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. *Conference on Computer Vision and Pattern Recognition*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017).

Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Technical Report*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, J., Zhou, P., Xiong, C., and Hoi, S. (2021). Prototypical contrastive learning of unsupervised representations. *International Conference on Learning Representations*.

Li, Z. and Hoiem, D. (2016). Learning without forgetting. *European Conference on Computer Vision*.

Lin, H., Zhang, B., Feng, S., Li, X., and Ye, Y. (2023). Pcr: Proxy-based contrastive replay for online class-incremental continual learning. *Conference on Computer Vision and Pattern Recognition*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Neural Information Processing Systems conference*.

Madaan, D., Yoon, J., Li, Y., Liu, Y., and Hwang, S. J. (2022). Representational continuity for unsupervised continual learning. *International Conference on Learning Representations*.

Mallya, A. and Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. *Conference on Computer Vision and Pattern Recognition*.

Purushwalkam, S., Morgado, P., and Gupta, A. (2022). The challenges of continuous self-supervised learning. *European Conference on Computer Vision*.

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

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *Neural Information Processing Systems conference*.

Thrun, S. (1995). A lifelong learning perspective for mobile robot control. *Elsevier*.

Tiwari, R., Killamsetty, K., Iyer, R., and Shenoy, P. (2022). Gcr: Gradient coreset based replay buffer selection for continual learning. *Conference on Computer Vision and Pattern Recognition*.

Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Hong, L., Zhang, S., Li, Z., Zhong, Y., and Zhu, J. (2022). Memory replay with data compression for continual learning. *International Conference on Learning Representations*.

Ye, M., Zhang, X., Yuen, P. C., and Chang, S.-F. (2019). Unsupervised embedding learning via invariant and spreading instance feature. *Conference on Computer Vision and Pattern Recognition*.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. *International Conference on Learning Representations*.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. *International Conference on Machine Learning*.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. *International Conference on Machine Learning*.

Zhang, C., Zhang, K., Pham, T. X., Niu, A., Qiao, Z., Yoo, C. D., and Kweon, I. S. (2022). Dual temperature helps contrastive learning without many negative samples: Towards understanding and simplifying moco. *Conference on Computer Vision and Pattern Recognition*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.