Revisiting a kNN-based Image Classification System with High-capacity Storage (*Supplementary materials*)

Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki,
Yu-Chieh Lin, and Jun Deguchi

Kioxia Corporation, Kawasaki, Japan
{kengo1.nakata, youyang.ng, daisuke1.miyashita, asuka.maki,
yuchieh.lin, jun.deguchi}@kioxia.com

A Applications of our system

Our system registers feature maps extracted from images and their corresponding labels to the database, as described in Section 3.2 of the main paper. A feature map can be linked with not only a single label but multiple labels as well. For example, when extracting a feature map from a *cat* image, our system can register not only the *cat* label but also the labels *tabby* (the subcategory) and *mammal* or *animal* (the broad category or superclass) with the feature map. In the main paper, we introduced the procedure of linear search for all feature maps in the storage when calculating the distance with the feature maps of the query image. Our system can use multiple registered labels to refine the search according to the classification setting. For instance, in the case of an animal classification problem, our system can selectively list the files of image feature maps linked to the *animal* label before the distance calculation. Then, by loading only the listed image feature maps to the computing device, the distance calculation can be processed even with limited memory resources.

By analyzing the number of references for each sample in the database, we can identify frequently or rarely referred samples, and interpret what knowledge is currently required or unnecessary. The information on the number of references can be used for prioritization when verifying the validity of stored knowledge. For example, when checking labels of samples within a limited amount of time, users can review the labels of frequently referred samples first, instead of checking all the samples in the storage. Moreover, if users need to reduce the storage size, rarely referred samples can be deleted from the storage like pruning model parameters.

In addition, the inference time of our system is as short as a few milliseconds, as shown in Fig. 5 of the main paper. Therefore, we believe that our system can be implemented in real-time applications, e.g., infrastructure monitoring systems, where objects to be detected change temporally and need to be identified in real time without training. Our system can log the samples referred for inference, along with the original images and file names. When some anomaly or error occurs, operators can effectively investigate the cause later by checking the logs.

Table A1. Details of image encoder models.

Model	Output feature dimension	Input resolution	ResNet blocks	ResNet width
RN50	1024	224	(3,4,6,3)	2048
RN101	512	224	(3,4,23,3)	2048

Model	Output feature dimension	Input resolution	Vision Transformer layers	Vision Transformer width	Vision Transformer head
ViT-B/32	512	224	12	768	12
ViT-B/16	512	224	12	768	12
ViT-L/14	768	224	24	1024	16

B Pretrained image encoder models

In the main paper, we primarily employed image encoder models pretrained by CLIP [9]. We summarize the details of the image encoders in Table A1.

C Processing time for distance calculation

As described in Section 3.3 of the main paper, our system retrieves nearest neighbor samples based on the cosine distance between a query and the stored feature maps. In the experiments in this work, the memory capacity required for using the stored feature maps is 5.3 GB at most. Therefore, all of them can be fully loaded into GPU memory, and a linear search based on the distance calculation is executed on GPU. Fig. A1 shows the processing time for distance calculation between a query and the stored feature maps measured on the NVIDIA A100 GPU. As shown in Fig. A1, the processing time increases with the number of stored feature maps. However, in terms of the overall inference procedure, the processing time for query encoding is more dominant than that for distance calculation, so the total processing time does not increase much as the number of stored feature maps increases, as shown in Fig. 5 in the main paper.

If the number of stored feature maps is scaled up (e.g., 10×), all the feature maps may not fit completely into the memory. Then, the time required for memory access will be non-negligible. As methods for scaling up our system, we can utilize (i) large-scale fast ANN (approximate nearest neighbor) search with storage [4,6], (ii) larger memory with the latest GPUs, and/or (iii) computational storage [12], and we are currently investigating them as future work.

D Hyperparameter settings in continual learning scenarios

In Section 4.2 of the main paper, we compared the performance of our system with those of conventional methods proposed for continual learning scenarios

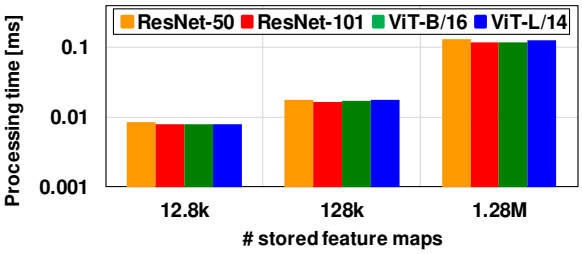

Fig. A1. Processing time for distance calculation measured on the NVIDIA A100 GPU.

(iCaRL, ER, GEM, A-GEM, and EWC [10,11,8,3,7]). We employed a fully connected layer as a classifier head for these methods, and applied pretrained models by CLIP to the initial values. We employed a stochastic gradient descent algorithm as an optimizer and trained the image encoder and classifier models for 50 epochs on the Split CIFAR-100 dataset [3,13]. We set the learning rate to 0.01 and the mini-batch size to 32. In iCaRL, ER, GEM, and A-GEM, we set memory buffer size to 500 for storing a part of training data learned in the past in the memory. In EWC, we set λ for the penalty term to 0.5 without storing any training data learned in the past. For these hyperparameter settings, we referred to the values in Refs. [1,2].

E Final accuracy evaluation in class incremental setting

In Table 4 in Section 4.2 of the main paper, we evaluated the performance in class-incremental learning with the averaged accuracy values from the first to the last set of classes. We can also evaluate the performance with the final accuracy values after the last set of classes has been learned. Table A2 summarizes the final accuracy in class-incremental learning with ResNet-50 on Split CIFAR-100. As shown in Table A2, our approach is also effective when evaluating the final accuracy.

Table A2. Final accuracy evaluation in class incremental setting with ResNet-50 on Split CIFAR-100.

ResNet-50 on Split CIFAR-100							
Method	**Ours**	CLIP	iCaRL	ER	GEM	A-GEM	EWC
Accuracy [%]	**55.9**	42.8	27.8	9.8	7.0	4.8	3.0

F Future works for better pretraining

In Section 3.1 of the main paper, we described that the performance of our system depends on the pretraining methods of image encoder models. In this work, we directly used the image encoder models pretrained by CLIP, and we did not optimize the pretraining condition or the architecture of the pretrained models (e.g., the number of output feature dimensions) for our system.

CLIP contrastively learns image and text encoders such that relevant image-text pairs are mapped to the neighborhood in the latent space. In this pretraining, a kNN classifier is not employed as a head model to calculate the contrastive loss, and the pretrained models are not optimized for the kNN classifier. Recently, Ref. [5] has proposed a contrastive learning method that explicitly incorporates a nearest neighbor algorithm into the pretraining for image encoders. This method contrastively learns image encoders so that input images and retrieved ones from a support set by a nearest neighbor search are mapped to the neighborhood in the latent space. To further improve the accuracy of our system, we need to devise better pretraining method and condition suitable for similarity-based retrieval using kNN, which is a future work.

References

1. Buzzega, P., Boschini, M., Porrello, A., Abati, D., CALDERARA, S.: Dark Experience for General Continual Learning: a Strong, Simple Baseline. In: NeurIPS (2020)
2. Buzzega, P., Boschini, M., Porrello, A., Calderara, S.: Rethinking Experience Replay: a Bag of Tricks for Continual Learning. In: ICPR (2021)
3. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient Lifelong Learning with A-GEM. In: ICLR (2019)
4. Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., Wang, J.: SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In: NeurIPS (2021)
5. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: With a Little Help From My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations. In: ICCV (2021)
6. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2019)
7. 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., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114** (2017)
8. Lopez-Paz, D., Ranzato, M.: Gradient Episodic Memory for Continual Learning. In: NIPS (2017)
9. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning Transferable Visual Models From Natural Language Supervision. In: ICML (2021)
10. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental Classifier and Representation Learning. In: CVPR (2017)

11. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience Replay for Continual Learning. In: NeurIPS (2019)
12. Torabzadehkashi, M., Rezaei, S., HeydariGorji, A., Bobarshad, H., Alves, V., Bagherzadeh, N.: Computational storage: an efficient and scalable platform for big data and hpc applications. Journal of Big Data **6**(1), 1–29 (2019)
13. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML (2017)