Abstract

In visual search, the gallery set could be incrementally growing and added to the database in practice. However, existing methods rely on the model trained on the entire dataset, ignoring the continual updating of the model. Besides, as the model updates, the new model must re-extract features for the entire gallery set to maintain compatible feature space, imposing a high computational cost for a large gallery set. To address the issues of long-term visual search, we introduce a continual learning (CL) approach that can handle the incrementally growing gallery set with backward embedding consistency. We enforce the losses of inter-session data coherence, neighbor-session model coherence, and intra-session discrimination to conduct a continual learner. In addition to the disjoint setup, our CL solution also tackles the situation of increasingly adding new classes for the blurry boundary without assuming all categories known in the beginning and during model update. To our knowledge, this is the first CL method both tackling the issue of backward-consistent feature embedding and allowing novel classes to occur in the new sessions. Extensive experiments on various benchmarks show the efficacy of our approach under a wide range of setups.

1. Introduction

Continual learning (CL) aims to learn new tasks while keeping the functions learned from the old sessions. The technology has been rapidly evolving; nevertheless, the active research area in CL focuses on image classification but ignores the demand for image retrieval (aka visual search). For obtaining powerful feature representations in image retrieval, most works [37, 44, 57, 67] still require a model to be trained on an entire dataset simultaneously instead of in an incremental manner. However, a practical visual search system should be capable of continually learning from new materials while consolidating the old knowledge to cope with the data accumulated with time.

As data grows, despite updating the model by simply fine-tuning, many observations [11, 69] reveal that catastrophic forgetting happens. A series of strategies have been developed in CL to address the problem [11, 29, 43]. The methods can make a single deep model capable of updating itself successively while avoiding disappointing overall performance. However, there are still several ongoing issues.

First, many works in CL emphasize the disjoint setup where the data from the old class will not show during training in a new task (or session). The task boundary arising over classes restricts the usage of CL since many retrieval systems need to collect extra data of the seen labels for improving their models in new sessions. Although recent studies (e.g., [2, 4]) allow the class overlapping among the tasks, the blurry setup in these works assumes that all the class labels in the future sessions are pre-given in advance; only the instance ratios in the classes vary with the session (Fig. 1). This kind of data scenario is impractical for most visual-search applications (e.g., in an e-commerce system, the new
product arrives over season). Moving toward a more general setup is desirable to fulfill real-world scenarios.

Second, a model updated from new data will deploy online in retrieval. An essential step is to re-extract the feature embedding from previous gallery images to maintain a consistent feature space on the pairwise distance measurement. For visual search on large-scale data, feature re-extraction is computationally intensive. Thus, an ideal design for continual visual search is that an updated model only extracts features for incoming gallery data while keeping the previously generated feature representations unchanged. However, it leads to the further difficulty of pairwise similarity measurement in uneven feature spaces. Motivated by this, we argue that current CL studies lack consideration for feature compatibility between the ongoing and previous data. Hence, designing a CL algorithm with backward consistent feature embedding is demanded.

To address the above issues, we introduce a novel CL approach, namely, CVS (Continual-learner for Visual Search), for a generally incremental setup of visual search. CVS can learn effective feature representations with backward consistency. For learning new knowledge, our learner obtains discriminating features for the current task. We introduce a cross-task gallery embedding consistency constraint that keeps the currently learned features compatible with the representatives from the obsolete gallery features. For consolidating old knowledge while keeping feature consistency, we develop a metric-based knowledge distillation to draw together the embedding from the different feature spaces. By coordinating the components, CVS can achieve continual visual search with backward-compatible feature embedding effectively. Also, we conduct extensive experiments under various incremental data distributions, especially for the general-incremental setup, to validate the effectiveness of CVS. The main characteristics include:

**General Incremental Setup:** We introduce a new CL scenario to simulate real-world visual search application systems. It includes the previous disjoint and blurry setups as special cases and tackles the general setting that the classes in a coming session can be either seen or unseen before.

**Backward Consistent Feature Space Learning:** Our learner can learn discriminative features for unseen classes. It can also maintain the distance metric learned effective for the seen classes in both new and old sessions, where the old-session features can be kept unchanged without the need to be re-extracted every time in a visual search system.

Extensive experiments on multiple datasets under incremental data distributions show that our method achieves state-of-the-art results. Fig. 2 shows our system diagram.

### 2. Related Work

We briefly review the recent progress of image retrieval in Section 2.1. We then summarize the CL and describe its challenge on similarity-based visual search in Section 2.2.

#### 2.1. Image Retrieval

Image retrieval ranks the gallery images in order given the query image. Previous studies rely on either descriptors by local feature aggregation [23, 39, 49] or low-level visual clues [5, 35, 62] and then perform a nearest neighbor search. The modern way utilizes embeddings from the neural network (i.e., neural mapping) instead because the learnable descriptors show superior accuracy [10, 36, 45, 50, 51, 67] and compact storage [7, 30, 64, 65]. To derive the neural mapping, metric learning optimizes the model in a pairwise manner or pointwise manner. Pairwise methods derive the discriminating space where positive pairs become closer and negative pairs repel each other. A fair benchmark [44] shows the state-of-the-art results of the pairwise method. However, an issue is that the complexity of the sample mining grows in magnitude when the number of samples in a group is beyond two, (e.g., triplet loss [45] and quadruplet [10]). Even though many studies [17, 57-60] delve into the mining issue, sampling the informative pairs is intrinsically hard without clear indicators. On the other hand, pointwise methods [25, 36, 41, 52, 55, 67] treat the optimization as data points against representative samples per class. Instead of meticulous mining, the representative samples could be learnable proxies [25, 36, 41, 52] or randomly sampled points [55, 67], bringing faster convergence for training without pairwise comparisons. For example,
Though an automatic task selector [1] can help extend an regularization-based methods [8, 26, 46, 66] impose ods [21,33, 34, 47] assign the subnetwork capacity per task. replay-based methods [4,6,15,31,43] revisit a small amount knowledge from the previous learner to the current one, distillation-based methods [13,14, 29, 42, 68] transfer the constraints on change of the critical neural weights, i.e task-incremental and class-incremental CL, where the for-

2.2. Continual Learning

CL aims for a single learner that can sequentially update knowledge without forgetting the previously learned information. The existing works [20,53] can be categorized into task-incremental and class-incremental CL, where the former assumes the task index presented at the inference time while the later assumes a task-agnostic scenario during inference. Most works assume the categories in the successive tasks (i.e., sessions) to be disjoint to each other.

To avoid overfitting to the current session, regularization-based methods [8, 26, 46, 66] impose constraints on change of the critical neural weights, distillation-based methods [13, 14, 29, 42, 68] transfer the knowledge from the previous learner to the current one, replay-based methods [4,6,15,31,43] revisit a small amount of old data to prevent forgetting, and isolation-based methods [21,33, 34, 47] assign the subnetwork capacity per task. Though an automatic task selector [1] can help extend an isolation-based method to class-incremental, it still requires the disjoint assumption of the classes among tasks. To break the disjoint-class limitation in CL, recent studies propose the blurry setup (i.e., class-overlapping [2, 4, 40]). However, it requires the presentation of all predefined classes across subsequent sessions, and is thus still impractical for real-world visual search applications.

Besides, most works focus on the classification problem, ignoring the demand for image retrieval. MMD [11] studies retrieval-based CL in a distillation manner. By narrowing down the mean difference between two distributions in reproducing kernel space, the current model distills the knowledge of the old one. However, the method ignores the backward-compatible needs for retrieval; hence the embedding have to be re-extracted for the gallery data as the task grows in CL. It is thus impractical for a realistic system.

3. Continual Learner for Visual Search (CVS)

Consider a CL retrieval problem with J sessions. In session j ∈ {1 ··· J}, a neural-network model is learned and let f_j be the mapping from the input image to the feature embedding of the model. During CL training, the backbone feature extractor f_j+1 is initialized from f_j, the neural mapping obtained immediately before the current session. To fulfill the requirement in our General-Incremental setup, the gallery set is incremen-
tally extensible with the newly added embedding. Specifically, let G_j be the images newly added in time session j and g_j be the feature embedding obtained for G_j (via f_j). The embedding will be added to the gallery set up to the j-th session, g_0:j = ∪_{l=1}^{j} g_l. The feature embedding cumulatively saved (i.e., g_1:j) then serve as the gallery set for retrieval for future sessions l (> j). In addition, to learn f_{j+1} capable of handling the new-session data containing possibly both existing and novel classes and also maintaining the performance on all previous-session data, we conduct the following loss terms in our CVS approach, namely, inter-session term of data coherence (L^D_{1:j+1}), neighbor-session term for model coherence (L^M_{j+1}), and intra-session term for discrimination (L^C_{j+1}).

3.1. Inter-session data coherence

Each training sample of the previous-session data has been already provided with an embedding by one of the old neural mappings, f_1, ···, f_j. To fulfill the condition of not re-extracting the embedding for updating the gallery set every time, they are fixed and kept invariant once having been built upon a session k ∈ {1 ··· J}. That is, after establishing the embedding, they can be used for a series of future tasks in CL. Besides hoping to un-forget the classification power of known labels, we further need to un-forget the capability of retrieval based on the embedding already built.

Replayed Embedding Together with Data: Replayed data are widely adopted to avert forgetting in CL. To resolve stor-
age efficiency and avoid training on all data, a small portion of the data from sessions 1 to \( j \) can be stored and replayed for a joint training with the current data in session \( j+1 \). In our work, the old data embedding have been extracted as \( g_i = f_i(G_i) \). Besides replaying the data, we replay the embedding to facilitate CL, which is more efficient with learning. We call the technique *replayed embedding*, where a small portion of features sampled from the embedding space are replayed for training.

The deep model updated in the current session \( j+1 \) should be confined to the stored embedding of seen classes. Therefore, the goal of our learner is to train \( f_{j+1} \) to extract features from \( G_{j+1} \) \( g_{j+1} = f_{j+1}(G_{j+1}) \) while keeping the existing obsolete features \( g_{1:j} \) unchanged. However, the embedding are established from different neural mappings varying with sessions. The sessions have separated distributions and could be diverse for a long sequence of sessions. As different feature spaces are not necessarily comparable, to avoid an over-fitting to individual sessions, we aggregate the embedding across the previous sessions by taking the expectations as follows:

\[
E_c = \frac{1}{j} \sum_{i=1}^{j} \xi(g_i), \quad c \in C(i),
\]

(1)

where \( C(i) \) is the set of class indices appearing in session \( i \), \( g_i \) is the set of embedding extracted from the data of class \( c \) in \( G_i \), and \( \xi(\cdot) \) denotes the expectation operator. The loss enforcing inter-session data coherence is defined as follows:

\[
L_{\{1:j\};j+1}^{d} = \sum_{c \in \Pi_{j+1}} \sum_{x_i \in c} \| f_{j+1}(x_i) - E_c \|^2,
\]

(2)

where \( \Pi_{j+1} = \bigcup_{i=1}^{j} C(i) \cap C(j+1) \) is the indices intersection between the classes in the current and all previous sessions; \( x_i \) denotes the data in the current session \( (j+1) \).

However, the current-session classes may not contain all the previous classes in our General Incremental setup. E.g., for the special case of Disjoint setup, there is no old class sample in the incoming sessions at all. Hence, besides employing the replayed embedding \( E_c \) (in \( \Pi_{j+1} \)), we use the replayed data for the old classes \( \Gamma_{j+1} = \bigcup_{i=1}^{j} C(i) \) too. Without loss of generality, we use the exemplar mining technique iCaRL [43] to conduct a small portion of data for replay, which searches random neighbors around the mean per class. The replayed embedding and data in \( \Gamma_{j+1} \) are co-used as follows:

\[
L_{\{1:j\};j+1}^{d} = \sum_{c \in \Gamma_{j+1}, \tilde{x}_i \in c} \| f_{j+1}(\tilde{x}_i) - E_c \|^2,
\]

(3)

where \( \tilde{x}_i \) denotes the replayed data. Then, the inter-session data coherence loss is a combination of Eqs. 2 and 3, which helps maintain the backward feature consistency:

\[
L_{\{1:j\};j+1}^{d} = (L_{\{1:j\};j+1}^{d} + L_{\{1:j\};j+1}^{d}) / n,
\]

(4)

where \( n \) is the mini-batch size.

The loss combines both the replayed embedding and data to maintain the coherence of the feature spaces of sessions \( 1, \cdots, (j+1) \). Even the aggregated embedding provide strong constraints, as the freedom of neural networks is often large in the capacity, the learner can still easily adapt the model from \( f_j \) to \( f_{j+1} \). In our experience, a nice usage of this precedent constraint can help build an effective CL learner because the fixed embedding can act as attractors to regularize the deep model training. Our approach only needs \( |\Gamma_{j+1}| \) replayed embedding, and we set a ratio of around 5% samples of the total data in sessions \( 1 : J \) as a fixed budget shared by all sessions up to now in the replayed data. Since the loss establishes the connection between session \( (j+1) \) to all previous ones \((1 : j)\), unlike BCT [48] employing only the inter relationship between the current session \( (j+1) \) and that immediately before the session \( (j) \), our approach can effectively enhance the backward feature consistency for long-term sessions.

Besides, our replayed-embedding solution coincides with the idea in the cross-batch learning [58,61], where previously learned embedding in the past mini-batches guide the training in the current one. The approaches have been demonstrated effective to boost retrieval performance. Our approach is analogously a cross-session learning method. In the experiments, we show the efficacy of our cross-session CL solution and make an in-depth analysis on the results.

### 3.2. Neighbor-session model coherence

In the above, we utilize the replayed materials of seen classes in all previous sessions and the current session data for training. This section develops the loss using mainly the current-session data while co-using the current model \( f_{j+1} \) and last-time model \( f_j \) in training. It enforces the neighbor-session model coherence.

The loss is designed using the distillation principle, where the mapping \( f_i \) (learned and fixed) serves as the teacher model to guide the training of the student \( f_{i+1} \). Unlike previous studies of distillation techniques that are mainly for classification [18], we conduct a distillation-based loss to regulate the metric learning of \( f_{j+1} \) from the metric space of \( f_j \) based on the triplet loss to better fit the nature of CL on retrieval. In a triple of \((x_a, x_p, x_n)\) for an anchor, a positive sample, and a negative sample respectively are selected from mainly the current-session data. The purpose of training is to shrink the distance between \( x_a \) and \( x_p \) while enlarging that between \( x_a \) and \( x_n \).

To distill knowledge from the embedding space already built from the previous session \( j \), we generate the embedding of positive and negative samples using \( f_j(\cdot) \) and obtain \( f_j(x_p) \) and \( f_j(x_n) \), respectively. Then, we use the teacher-generated embedding to guide the training of the student anchor \( f_{j+1}(x_a) \). The current session data are fed into the cur-
rent and previous models to constrain the embedding distributions produced by \( f_{j+1} \) and \( f_j \), respectively. The loss conducted can thus enforce the model coherence between neighbor sessions.

Besides, we co-use \( x_a \) and \( x_p \) via setting \( x_p = x_a \). Hence, we conduct a 2-samples-3-embedding triplet-loss strategy where the triplet embedding are \( f_{j+1}(x_a) \), \( f_j(x_a) \), and \( f_j(x_n) \). This is because it can facilitate drawing the embedding distributions of the student and the teacher based on the same \( x_a \) for comparison. In the meantime, it also saves the computation of choosing positive samples and enforces efficient sampling. Recent studies have shown that mining easy positive samples (i.e., similar positives) benefits to metric learning [60]. Our approach directly forms the positive sample’s embedding from the anchor, which eliminates the positive mining effort in [60] during pair sampling. For the negative, we follow the hardest negative mining principle [17] to pick the embedding. Denote \( d^{j+1}(x, y) = \| f_{j+1}(x) - f_j(y) \|^2 \). The neighbor-session-model-coherence loss is written as:

\[
L^m_{j,j+1} = \frac{1}{n} \sum_{x_a} \left[ d^{j+1}(x_a, x_a) - d^{j+1}(x_a, x_n) + m \right]_+, \tag{5}
\]

where \( m \) is the margin set as 0.1 by default. Besides, \( f_{j+1}(x) \) and \( f_j(x) \) are \( l_2 \)-normalized to alleviate the issue of forgetting according to previous studies [19, 32]. This loss helps restrict the behavior of the updated model coherent to the previous one.

### 3.3. Intra-session discrimination

Recent studies show that classification is a strong baseline for learning effective feature embedding (if an appropriate \( l_2 \)-norm normalization layer is added) [67]. Without loss of generality, we employ the method to establish the retrieval capability of \( f_{j+1} \) using mainly the current-session data. The loss for intra-session discrimination is

\[
L^c_{j+1} = \frac{1}{n} \sum_{i=1}^n -\log \left( \frac{\exp(w^c_{f_{j+1}(x_i)}/T)}{\sum_k \exp(w^c_{f_{j+1}(x_i)}/T)} \right), \tag{6}
\]

where \( (x_i, y_i) \) are the data and labels, \( w \) denote the \( l_2 \)-normalized weight of the classification layer, \( f_{j+1}(x_i) \) is the \( l_2 \)-normalized embedding extracted for \( x_i \), and \( T \) is the temperature term set as 0.05 by default.

**Total Loss.** The final learning objective is as follows:

\[
\min_{f_{j+1}} L = L^c_{j+1} + \alpha L^m_{j,j+1} + \beta L^d_{(1:j);j+1}. \tag{7}
\]

By default, we set \( \alpha \) to 10 and \( \beta \) to 1 empirically in our experiments, and cosine distance is used for retrieval in this work. Only the first term \( L^c \) is used in session 1. Then, all three terms are used in sessions 2 to \( J \). Fig. 2 gives an illustration of our approach.

In sum, to fulfill the General Incremental setup, the first term \( L^c_{j+1} \) provides a fundamental retrieval ability of the current session (\( j + 1 \)). The second term \( L^m_{j,j+1} \) makes the neural mapping \( f_{j+1} \) close to \( f_j \) based on both the novel- and seen-label data in the current session. It enforces the prediction to mimic the behavior of the previous feature extractor. The third term \( L^d_{(1:j);j+1} \) enforces the backward feature consistency from the replayed embedding and data of all sessions (1 to \( j \)) and a joint training with the current-session data based only on the labels seen before (i.e., \( \bigcup_{i=1}^j C(i) \)). It helps bias the feature space to align with the one obsolete gallery features lie in. By joining the three losses, our CVS approach can handle the General Incremental setup and its special cases (Blurry, Disjoint) well.

### 4. Experiments

We conduct extensive experiments across various datasets under different data distributions. Five datasets are used, including two coarse-grained datasets, CIFAR100 and Tiny ImageNet, and three fine-grained datasets, Stanford Dog, iNaturalist 2017, and Product-10K. We use CIFAR100 for fundamental study and then Tiny Imagenet on a longer sequence of sessions.

The datasets are summarized as follows. **CIFAR100** [27] has 100 categories; each owns 500 \( 32 \times 32 \) images for training and 100 images for testing. **Tiny ImageNet** [28] has 100,000 training and 10,000 testing images (of size \( 64 \times 64 \) sampled from the 200 classes from ImageNet. Due to its compact size and various categories, we conduct a long sequence learning on it. **Stanford Dog** [24] contains 120 dog breed-level categories picked from ImageNet, including 8,580 images for testing and 12,000 images for training. For the training split, each class has 100 images. **iNaturalist 2017** [54] is a large-scale long-tailed image retrieval dataset with 5,089 species-level categories. We sample 527 images per class for the 200 classes; each contains at least 527 samples to avoid very small classes and unbalanced data before partition. We call it **iNat-M** in this work. **Product-10K** [3] is an ultrafine-grained long-tailed dataset covering the top-9691 frequently bought product images from a real e-commerce system; Each class’s images contain diverse appearances of the specific product by collecting offline customer-taken and online in-shop photos. We remove the category with fewer than 20 images in the training set. We then construct the train-test split based on the original training set because the official testing data do no offer labels. Finally, there are 2,743 classes. We call it **Product-M** in this work. Fine-grained images are more challenging as only subtle differences exist.

#### 4.1. Implementation Details

In all of the experiments, the embedding dimension sets as 128 by default. We use ResNet-18 [16] for coarse-
grained datasets, and ResNet-50 [16] for fine-grained datasets. The hyperparameter details are depicted in the supplementary material.

**Disjoint setup** assumes that previously seen classes are unavailable when learning new session data. As shown in Fig. 3a, we partition the CIFAR100 into five sessions with each containing 20 classes. For Tiny ImageNet, the dataset is divided into ten sessions with 20 classes per session.

**Blurry setup** [4] provides all categories in the beginning; each session shows a subset of samples from all of them. The data distribution is controlled via a percentage of the data from the major and minor classes per session. In our experiments, CIFAR100 is divided into five sessions; each contains 20 major classes and 80 minor classes, with 90% samples from the major and 10% from the minor, as shown in 3b. For Tiny ImageNet, we simulate a 10-session CL; each has 70% from the major and 30% from the minor.

**General-incremental setup**: The new-class and novel old-class samples may co-exist. The model is learned from the \( S \) classes at initial in a total of \( L \) sessions. For the later sessions, we add \( C \) classes and assume that \( M \% \) of the data are from the old categories and \( (1 - M)\% \) are from the new, denoted as \((S, C, M, L)\). We set \((20, 20, 10, 5)\) for CIFAR100 (Fig. 3c). For the rest, we apply \((20, 20, 30, 10)\) to Tiny ImageNet, \((60, 20, 30, 4)\) to Stanford Dog, \((100, 25, 30, 5)\) to iNat-M, and \((1343,700,40,3)\) to Product-M.

Replayed data are used for all loss terms. Following [4], we set the memory buffer size for replay as 2,000 for CIFAR100. Note that this is a budget-limited buffer (for the replay-based methods) co-used by all sessions. We assume a memory budget of 600 samples for Dog due to the small scale, 4,000 for Tiny ImageNet, 4,000 for iNat-M, and 3,000 for the ultrafine-grained Product-M, respectively. It is pretty challenging for Product-M as it has 2,743 classes; for each session, only 1 ~ 2 images will be stored for replay.

**4.2. Evaluation**

We follow the train-validation-test protocol. The model achieving the best recall@1 in the validation phase is picked for fair comparisons. Then, we report the final performance by using the original testing set as a query set. For collecting the validation query set for each class in disjoint and general-incremental setups, we sample 5% of current training session data for iNat-M and Tiny ImageNet, and 10% for CIFAR100 and Stanford Dog. We sample 2 images randomly from each class for Product-M due to scarce training data. Note that such a validation query set is accumulated as the session grows in the disjoint and general setup. For the blurry setup, we keep a fixed portion of data from the entire training set as a validation query set (5% for Tiny ImageNet and 10% for CIFAR100) class-wisely since the model can see all predefined categories in the beginning. The embedding extracted for the current-session training data is added to the gallery set after finishing the session as mentioned before; thus the new-classes images first observed in one season are allowed to be searched in the future seasons. Such an extensible gallery set matches the real-world scenarios.

We report the recall@\( K \) [22] as it is the most popular metric in fine-grained retrieval [44, 51]. For the testing query, we evaluate all the classes for the blurry setup, and the classes seen so far for both the disjoint and general setup, respectively. Finally, we average the scores over sessions, denoted as \( \text{AR@K} \). To compare with existing approaches, we re-implement the LWF [29], and MMD [11] and re-run EWC [26], RWalk [8], and Rainbow [4] from the official Github of [4]. Among them, all (except MMD) are for classification since CL is rare to be studied on retrieval yet, and we replace all the plain softmax loss with normalized softmax loss [67] for them for a fair comparison. We also re-implement BCT [48] that is a backward feature compatible approach for retrieval. We apply the same exemplar mining as ours for BCT in disjoint setup as the Naïve BCT is not a CL solution and requires old classes to pass through.

Besides, we provide direct fine-tuning as the lower bound; in contrast, we offer joint training of all gallery images seen so far allowing re-extraction as the upper bound.

**4.3. Results on the Coarse-grained Datasets**

**Results on CIFAR100**: We conduct a 5-session experiments on CIFAR100. The results are shown in Fig. 4 (recall@1) and Table 1(a) (AR@K). For all three setups, except for Joint Train that is an upper bound of this experiment, our CVS performs the best on both evaluation measures. In the blurry and general-incremental setups, we observe that BCT performs even worse than Finetune, a lower
Table 1. Results on (a) CIFAR100 and (b) Tiny ImageNet, and (c) Fine-grained datasets, where Joint Train and Finetune specify the theoretically upper and lower bounds, respectively.

| Dataset | Disjoint | Blurry | General |
|---------|----------|--------|---------|
|        | AR@1     | AR@2   | Ours (CVS) |
|        | Rainbow  | RWalk  | EWC     |
| Joint Train | 85.39 | 66.30 | 84.84 |
| Finetune | 85.89 | 66.30 | 84.84 |
| BCT    | 85.89 | 66.30 | 84.84 |
| LWF    | 49.33 | 29.22 | 72.68 |
| MMD    | 49.33 | 29.22 | 72.68 |
| EWC    | 49.33 | 29.22 | 72.68 |
| RWalk  | 49.33 | 29.22 | 72.68 |
| Rainbow | 74.4 | 74.8 | 77.7 |

Figure 4. Recall@1 on CIFAR100 for the three setups, where Average Recall@1 across sessions is reported in parentheses.

Figure 5. Recall@1 on Tiny ImageNet for the three setups, where Average Recall@1 across sessions is reported in parentheses.

Table 1. Results on (a) CIFAR100 and (b) Tiny ImageNet, and (c) Fine-grained datasets, where Joint Train and Finetune specify the theoretically upper and lower bounds, respectively.

| Dataset | Disjoint | Blurry | General |
|---------|----------|--------|---------|
|        | AR@1     | AR@2   | Ours (CVS) |
|        | Rainbow  | RWalk  | EWC     |
| Joint Train | 85.39 | 66.30 | 84.84 |
| Finetune | 85.89 | 66.30 | 84.84 |
| BCT    | 85.89 | 66.30 | 84.84 |
| LWF    | 49.33 | 29.22 | 72.68 |
| MMD    | 49.33 | 29.22 | 72.68 |
| EWC    | 49.33 | 29.22 | 72.68 |
| RWalk  | 49.33 | 29.22 | 72.68 |
| Rainbow | 74.4 | 74.8 | 77.7 |

(a)

(b)

(c)
runner-up results, we obtain CVS (71.47) v.s. Anglewise-RKD [38] (70.45) in disjoint setup, CVS(47.47) v.s. Absolute MLKD [63] (46.9) in blurry setup, and CVS(73.95) v.s. Dark Knowledge [18] (73.18) in general incremental setup. We detail the results in the supplementary material.

**Comparisons on Classification:** As our CVS can produce the classification results via the NSoftmax layer too, we further compare the results with the CL classifier Rainbow [4] based on CIFAR-100 after finishing all sessions. We have already followed the setting of [4] on both the class distribution ratios and replayed buffer size in the blurry setup. The classification accuracy presented in [4] is 41.35% with an online learning protocol (*i.e.*, only a single epoch is allowed in learning). We rerun the learner to converge and get the accuracy of 50.2%. Surprisingly, CVS can achieve the 54.04% classification accuracy that is even higher. We owe the promising classification results to the replayed embedding that can serve as useful exemplars (*like the principle of cross-batch learning [58, 61]*) to further guide the training in our CVS. We conduct extra experiments on the other two setups (disjoint, general-incremental) and one additional CL classifier RWalk and obtain the results: CVS (50.62) v.s. Rainbow (46.69) v.s. RWalk (46.85) in disjoint setup, CVS (54.04) v.s. Rainbow (50.2) v.s. RWalk (50.89) in blurry setup, and CVS (55.49) v.s. Rainbow (52.26) v.s. RWalk (51.92) in general incremental setup. The results demonstrate the efficacy of CVS for CL.

### 4.4. Results on the Fine-grained Datasets

As for the fine-grained benchmarks, we consider the general-incremental setup only due to its practical usefulness. We initialize the model with ImageNet pretrained weights as it is a common practice in the fine-grained retrieval benchmark. We assume the first session presents samples from half of the classes from the dataset. It is a practical setting for a modern visual search system because a robust service should be well-trained to a certain extent before going online. Due to the fine-grained limits on the data amount, we partition the Stanford Dog dataset into four, iNat-M into five, and Product-M into three sessions, respectively. The results are shown in Table 1(c). Our CVS consistently outperforms the other approaches for all the fine-grained benchmarks and measures, except for AR@k for k = 4 on Stanford Dog Dataset. In sum, the evaluation results with other state-of-the-art approaches demonstrate that our approach is effective for both coarse- and fine-grained datasets on the general-incremental setup because we not only consider backward compatible embedding of the neighboring sessions but also long-term consistency with all the data of past sessions via replay embedding and data. In addition, this backward compatible feature also saves the CL from expensive computational costs of gallery feature re-extraction as model updates during each session.

| Dataset | AR@1 | AR@2 | AR@4 | AR@8 |
|---------|------|------|------|------|
| CIFAR100 | 0.92 | 1.35 | 2.20 | 3.08 |
| Dog | 0.72 | 0.81 | 0.92 | 0.96 |
| Product-M | 0.60 | 0.79 | 0.82 | 0.85 |

Table 2. Ablation study on each component, where L_c, L_d, and L_m are the losses of intra-session discrimination, neighbor-session model coherence, and inter-session data coherence, respectively.

### Discussion:

A *limitation* we find is that almost all methods (including ours) stagnate as the session expands, especially in the blurry setup. The margin between ours and upper bound enlarges gradually even though our method surpasses the remaining approaches. Therefore, there is still room for improvement in the long-term challenge.

**Ablation Study:** Our CVS consists of three loss terms, L_c, L_m, and L_d. To verify their effectiveness, we conduct the ablation study as shown in Table 2. First, we find that L_c + L_m has the greatest impact on the overall performance. It brings the gain in the range of 1.67% to 11.37% in AR@1 compared with L_c alone. Second, the effect of L_c + L_m is weaker despite a slight increase on CIFAR100 and Product-M compared to the one of L_c. Our finding suggests that adopting classification loss with distillation loss alone is insufficient. Therefore, seeking unification of additional information like L_d is practical in our setting. To examine the performance gain from the consistency loss in more depth, we detach the exemplar replay technique from our method. We denote L_c + L_d w/o replay as the aforementioned case. The replay-based trick improves the plain version of the consistency loss on all datasets by a margin, especially for CIFAR100. Without reviewing exemplar data, the performance drop ranges between 1.08% and 7.66% in AR@1. Hence, integrating this design is essential for overall performance.

### 5. Conclusion

In this work, we present a novel general incremental setup which allows the new gallery set of both seen and unseen classes incrementally added to the database and is closer to the real-world retrieval setup than the widely adopted disjoint and the recent blurry setups. Besides, we also propose a CL method for long-term visual search with backward consistent feature embedding. Our method acts as an extension of cross-batch memory to the cross-session memory for feature embedding learning in CL. We introduce a 2-sample-3-embedding strategy in a triplet for distillation learning across neighbor sessions to enforce the model coherence. The extensive experiments show that our approach achieves the state of the art performance in two coarse-grained classification and three fine-grained datasets under different incremental data distributions.
References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In CVPR, pages 3930–3939, 2020.

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In NeurIPS, pages 11816–11825, 2019.

[3] Yalong Bai, Yuxiang Chen, Wei Yu, Linfang Wang, and Wei Zhang. Products-10k: A large-scale product recognition dataset. arXiv preprint arXiv:2008.10545, 2020.

[4] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In CVPR, pages 8218–8227, 2021.

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. PAMI, 24(4):509–522, 2002.

[6] P. P. Brahma and Adrienne Othon. Subset replay based continual learning for scalable improvement of autonomous systems. In CVPRW, pages 1179–11798, 2018.

[7] Zhanjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continual learning. In ICCV, pages 5609–5618, 2017.

[8] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In ECCV, pages 556–572, 2018.

[9] Ken Chen, Yichao Wu, Haoyu Qin, Ding Liang, Xuebo Liu, and Junjie Yan. R³ adversarial network for cross model face recognition. In CVPR, pages 9860–9868, 2019.

[10] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: A deep quadruplet network for person re-identification. In ICCV, pages 1320–1329, 2017.

[11] Wei Chen, Yu Liu, Weiping Wang, Tinne Tuytelaars, Erwin M Bakker, and Michael Lew. On the exploration of incremental learning for fine-grained image retrieval. In BMVC, 2020.

[12] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In AAAI, pages 2852–2859, 2018.

[13] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In CVPR, pages 5133–5141, 2019.

[14] Arthur Douillard, Matthieu Cord, Charles Olsson, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In ECCV, pages 86–102, 2020.

[15] Tyler L. Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In ICRA, pages 9769–9776, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016.

[17] Alexander Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737, 2017.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In NeurIPS, 2015.

[19] Sainui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In CVPR, pages 831–839, 2019.

[20] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In NeurIPS, 2018.

[21] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In NeurIPS, pages 13647–13657, 2019.

[22] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. PAMI, 33:117–128, 2011.

[23] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In CVPR, pages 3304–3311, 2010.

[24] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In CVPRW, 2011.

[25] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In CVPR, pages 3235–3244, 2020.

[26] 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. PNAS, 114(13):3521–3526, 2017.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[28] Ya Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.

[29] Zhizhong Li and Derek Hoiem. Learning without forgetting. PAMI, 40:2935–2947, 2018.

[30] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In CVPRW, pages 27–35, 2015.

[31] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In CVPR, pages 12242–12251, 2020.

[32] Chunjie Luo, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In ECCV, pages 72–88, 2018.

[33] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In ECCV, pages 72–88, 2018.

[34] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. PAMI, 18(8):837–842, 1996.

[35] Yair Movshovitz-Attias, A. Toshev, Thomas Leung, S. Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In ICCV, pages 360–368, 2017.
