On the Exploration of Incremental Learning for Fine-grained Image Retrieval

Wei Chen[1]
w.chen@liacs.leidenuniv.nl
Yu Liu[2]
yu.liu@esat.kuleuven.be
Weiping Wang[3]
wangwp@nudt.edu.cn
Tinne Tuytelaars[2]
Tinne.Tuytelaars@esat.kuleuven.be
Erwin M. Bakker[1]
e.m.bakker@liacs.leidenuniv.nl
Michael Lew[1]
m.s.k.lew@liacs.leidenuniv.nl

[1] MediaLab, LIACS,
Leiden University,
Leiden, The Netherlands

[2] ESAT-PSI,
KU Leuven,
Belgium

[3] College of Systems Engineering,
NUDT,
Changsha, China

Abstract

In this paper, we consider the problem of fine-grained image retrieval in an incremental setting, when new categories are added over time. On the one hand, repeatedly training the representation on the extended dataset is time-consuming. On the other hand, fine-tuning the learned representation only with the new classes leads to catastrophic forgetting. To this end, we propose an incremental learning method to mitigate retrieval performance degradation caused by the forgetting issue. Without accessing any samples of the original classes, the classifier of the original network provides soft "labels" to transfer knowledge to train the adaptive network, so as to preserve the previous capability for classification. More importantly, a regularization function based on Maximum Mean Discrepancy is devised to minimize the discrepancy of new classes features from the original network and the adaptive network, respectively. Extensive experiments on two datasets show that our method effectively mitigates the catastrophic forgetting on the original classes while achieving high performance on the new classes.

1 Introduction

In an era when the number of images is increasing, deep models for fine-grained image retrieval (FGIR) are required to be adaptable for new incoming classes. However, current image retrieval approaches are focusing mainly on static datasets and are not suited for incremental learning scenarios. To be specific, deep networks well-trained on original classes will under-perform on new incoming classes.

When new classes are added into an existing dataset, joint training on all classes allows to guarantee the performance. However, as the number of new classes increases sequen-

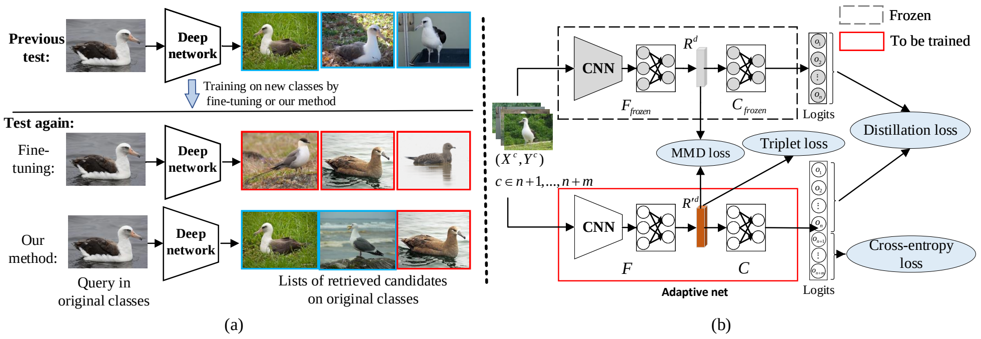


Figure 1: (a) Illustration of catastrophic forgetting for FGIR. Our method aims to maintain good performance on the original classes where the inaccurate returned images are in red box and correct results are in blue box. (b) Framework of our method. The only inputs for the adaptive net $\mathcal{B}$ are m new classes and labels $(\mathbf{X}^{c'}, \mathbf{Y}^{c'})$, $c' \in (n+1, \ldots, n+m)$. The frozen net $\mathcal{A}$ is firstly trained on n original classes and then copied as initialization for net $\mathcal{B}$.

tially, the repetitive re-training is time-consuming. Alternatively, fine-tuning makes the network adapt to new classes and achieve good performance on these classes. However, when the original classes become inaccessible during fine-tuning, the performance of the original classes degrades dramatically because of catastrophic forgetting, a phenomenon that occurs when a network is trained sequentially on a series of new tasks and the learning of these tasks interferes with performance on previous tasks, as shown in Figure 1(a).

Most of incremental learning methods are exploited for image classification, which is robust and forgiving as long as features remain within the classification boundaries. In contrast, image retrieval focuses more on the discrimination in the feature space rather than the classification decisions. Especially for FGIR, small changes on visual features may have a big impact on the retrieval performance. Additionally, we find that standard methods like Learning without forgetting (*i.e.* LwF [12]) and Elastic Weight Consolidation (*i.e.* EWC [9]) are insufficient for this problem because the distillation is not on the actual feature space (see Section 4.2 and 4.3).

Considering the above limitations, we propose a deep learning model to tackle the problem of incremental fine-grained image retrieval. We regularize the updates of the model to simultaneously retain preservation on original classes and adaptation on new classes. Importantly, to avoid the repeated training, the samples of the original classes are not used when learning the new classes. The classifier of the original network provides soft "labels" to transfer knowledge to train the adaptive network using the distillation loss function [4][25]. This focuses on pair-wise similarity but can not well quantify the distance between two feature distributions. This limitation inspires us to adopt a regularization term based on Maximum Mean Discrepancy (MMD) [3] to minimize the discrepancy between the features derived from an original network and an adaptive network, respectively. Moreover, the cross-entropy loss and triplet loss are utilized to identify subtle differences among sub-categories.

In summary, our contributions are two-fold. First, our work extends FGIR in the context of incremental learning. This is the first work to study this problem, to the best of our knowledge. Second, we propose a deep network, which includes a knowledge distillation loss and a MMD loss, for incremental learning without using any samples from the original classes. It achieves significant improvements over previous incremental learning methods.

2 Related Work

Incremental learning is the process of transferring learned knowledge from an original model to an incremental model. It has been researched in a few applications like image classification [12][28][11][30], image generation [29][26], object detection [19], hashing image retrieval [24] and semantic segmentation [15]. To overcome the so-called catastrophic forgetting, numerous methods have been proposed. For example, a subset of data (exemplars) of original classes are stored into an external memory, and the forgetting is thereby avoided by replaying these exemplars [6][14][25]. Recently, GANs [2] are used to synthesize samples with respect to the previous data distributions [18][21], which avoids the shortcomings of memory-consuming and exemplar-choosing, but generating real-like images with complex semantics is a challenging task. Alternatively, regularization methods constrain the objective functions or parameters of deep networks to preserve the previously learned knowledge. The distillation loss function [4] is used to transfer knowledge of old classes [12][25]. The importance weight per parameter is estimated based on the old classes, and then is used as regularization to penalize essential parameter changes when training on new incoming classes [9].

3 Proposed Approach

Problem Formulation Given a fine-grained dataset which includes n class labels $(\mathbf{X}^c, \mathbf{Y}^c)$ where $c \in (1, ..., n)$, each sub-category c has a different amount of images in $\mathbf{X}^c$ and the ground-truth labels $\mathbf{Y}^c$. A deep network is trained to perform the retrieval task for the n classes. Consider the incremental learning scenario, images from m new classes are added sequentially or at once. We take as input only the images from m new incoming classes, *i.e.* $(\mathbf{X}^{c'}, \mathbf{Y}^{c'})$, where $c' \in (n+1, ..., n+m)$, to incrementally train the deep network. In this way, it is efficient to update the network with no need of re-training the original classes again. Besides, the image instances from the original classes are not always accessible due to privacy issue or memory limit. Finally, the aim is to continually train the network, to make it preserve promising performance for all seen classes.

Overall Idea As shown in Figure 1(b), our method includes two training stages. First, we train a network $\mathcal{A}$ on the original classes using cross-entropy and triplet loss on the output logits and representations. After $\mathcal{A}$ is well-trained, we make two copies of $\mathcal{A}$: one freezing its parameters when incrementally training, and the other adapting its parameters for the m incremental classes. We refer to this adaptive network as $\mathcal{B}$. It is initialized with parameters from $\mathcal{A}$, including the feature extraction layers F_{frozen} and classifier C_{frozen}, but extends the number of neurons in its classifier C, from which the output logits are $(o'_1, o'_2, \dots, o'_n, o'_{n+1}, \dots, o'_{n+m})$, and previous n neurons are copied from C_{frozen}. To overcome catastrophic forgetting, we propose to integrate two regularization strategies based on knowledge distillation and maximum mean discrepancy, respectively. Given a query image from either the original classes or newly added classes, we extract the features from the fully-connected layer for image retrieval. We introduce the details of our method below.

3.1 Semantic Preserving Loss

First, we train the model with the standard cross-entropy loss. Given the logits $(o_1, o_2, ..., o_n)$ and its class label $(y_1, y_2, \dots, y_n)$, the loss is $H(\mathbf{y}, \mathbf{o}) = -\sum(\mathbf{y} * log(softmax(\mathbf{o})))$. Note that we only use images from the new classes during incremental training, thus the classification is performed on $(o'_{n+1}, o'_{n+2}, \dots, o'_{n+m})$, the categorical cross-entropy loss function L_{ce} is

$$L_{ce} = -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(\frac{e^{o'_i(x)}}{\sum_{j=n+1}^{n+m} e^{o'_j(x)}})) \tag{1}$$

To identify subtle differences among categories, we adopt the triplet loss $L_{triplet}$ by mining the hard positive pairs and hard negative pairs based on feature vectors $\mathbf{R}$.

$$L_{triplet} = \frac{1}{N}\sum_{i=1}^{N}(\max(0, \lambda + S_{i,neg} - S_{i,pos})) \tag{2}$$

where $S_{i,neg}$ and $S_{i,pos}$, based on matrix multiplication (*i.e.* $S_{i,neg} = R_i R_{neg}^{\top}$), indicate the similarity of i^{th} hard negative and positive pairs, respectively. λ is the margin parameter.

3.2 Knowledge Distillation Loss

We rewrite (F_{frozen}, C_{frozen}) as (F_f, C_f) for simplicity. Knowledge distillation loss [4] is defined to regularize the activations of the output layer in both the old and new model. To be specific, we constrain the first n values in $(o'_1, o'_2, ..., o'_n, o'_{n+1}, ..., o'_{n+m})$ as close as possible to the logits $(o_1, o_2, ..., o_n)$ from the frozen network $\mathcal{A}$. Following the method in [25][12], when m new classes are added at once, we compute the knowledge distillation loss by

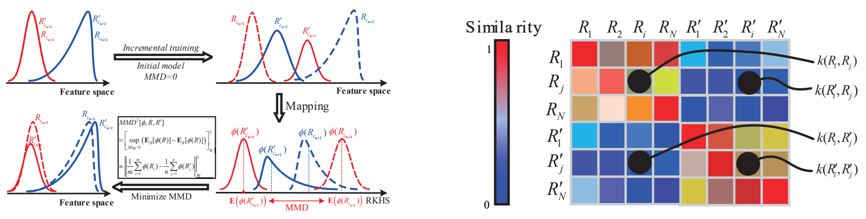
$$L_{dist} = -\frac{1}{|\mathbf{X}^{c'}|}\sum_{x\in\mathbf{X}^{c'}}^{|\mathbf{X}^{c'}|}\sum_{k=1}^{n}\{p_k(x) * log[p'_k(x)]\} \tag{3}$$

where $p_k(x) = \frac{e^{o_k(x)/T}}{\sum_j^n e^{o_j(x)/T}}$ and $p'_k(x) = \frac{e^{o'_k(x)/T}}{\sum_j^n e^{o'_j(x)/T}}$, T is a temperature factor that is normally set to 2 [12]. $\mathbf{p} = \{p\}_n$ and $\mathbf{p}' = \{p'\}_n$ refer to the probabilities produced by the modified Softmax function in [4]. F_f and C_f denote the parameters of network $\mathcal{A}$. Similarly, F and C denote the parameters of network $\mathcal{B}$, as shown in Figure 1(b). $|\mathbf{X}^{c'}|$ indicates the number of images from the new m classes in a mini-batch. n denotes the number of the original classes. Note that n will be extended accordingly when more new classes are added.

3.3 Maximum Mean Discrepancy Loss

Knowledge distillation loss focuses on constraining classification boundaries to mitigate the forgetting issue. However, for FGIR, it is more important to reduce the difference between feature distributions. For this, we adopt maximum mean discrepancy (MMD) [3] to capture the correlation of feature distributions between network $\mathcal{A}$ and $\mathcal{B}$. MMD has been used to bridge source and target distributions such as in domain adaptation [13][27]. However, our work is the first to impose MMD to regularize the forgetting issue for FGIR.

Given the features R^d (d is feature dimension) from network $\mathcal{A}$ and $\mathcal{B}$, MMD measures the distance between the means of two feature distributions after mapping them into a reproducing kernel Hilbert space (RKHS). In Figure 2(a), we illustrate how MMD mitigates the



(a) (b)

Figure 2: (a) The red and blue color depict the feature distributions of two categories. The dashed line indicates the distributions from the network $\mathcal{A}$, the solid line indicates that from the network $\mathcal{B}$. Since $\mathcal{A}$ is copied as the initial model for network $\mathcal{B}$, MMD=0 in the beginning. As training progresses, $\mathcal{B}$ is updated to change its output features and the MMD is expected to increase. (b) MMD for instance-to-instance similarity.

catastrophic forgetting issue. Note that, in the Hilbert space $\mathcal{H}$, norm operation can be equal to the inner product [1][3]. Finally, the squared MMD distance is:

$$\begin{aligned}
&\mathrm{MMD}^2(\mathbf{R},\mathbf{R}') = ||\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\phi(\mathrm{R_i}) - \frac{1}{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\phi(\mathrm{R'_j})||^2_{\mathcal{H}} \\
&= \frac{1}{N^2} < \sum_{i=1}^{N}\phi(R_i) - \sum_{j=1}^{N}\phi(R'_j), \sum_{i=1}^{N}\phi(R_i) - \sum_{j=1}^{N}\phi(R'_j) >_{\mathcal{H}} \\
&= \frac{1}{N^2}\Big[\sum_{i=1}^{N}\sum_{j=1}^{N} <\phi(R_i),\phi(R_j)>_{\mathcal{H}} + \sum_{i=1}^{N}\sum_{j=1}^{N} <\phi(R'_i),\phi(R'_j)>_{\mathcal{H}} - 2\sum_{i=1}^{N}\sum_{j=1}^{N} <\phi(R_i),\phi(R'_j)>_{\mathcal{H}}\Big] \\
&\quad s.t.\ R = F_f(x),\ R' = F(x)
\end{aligned} \tag{4}$$

where N is batch size, and $\phi(\cdot)$ denotes the mapping function. However, it is hard to determine $\phi(\cdot)$. In RKHS, the kernel trick is used to replace the inner product in Eq. 4, *i.e.* $<\phi(R),\phi(R')>=k(R,R')$. Considering all the features in a mini-batch, $\mathbf{R}=\{R\}_N$ and $\mathbf{R}'=\{R'\}_N$, we define the MMD loss L_{mmd} as:

$$L_{mmd} = \mathrm{MMD}(\mathbf{R},\mathbf{R}') = \frac{1}{\mathrm{N}}\Big[\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{k}(\mathrm{R_i},\mathrm{R_j}) - 2\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{k}(\mathrm{R_i},\mathrm{R'_j}) + \sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\mathrm{k}(\mathrm{R'_i},\mathrm{R'_j})\Big]^{\frac{1}{2}} \tag{5}$$

where $\mathrm{k}(\mathrm{R},\mathrm{R}')=\exp(-(||\mathrm{R}-\mathrm{R}'||^2_2)/(2\sigma^2_{\mathrm{m}}))$, σ_m means m variances in the Gaussian kernel.

Discussion. Knowledge distillation loss focuses on constraining pair-wise similarity. However, MMD loss measures the distance between each feature vector, as depicted in Figure 2(b). Finally, it captures the distance of two feature distributions from the frozen net and adaptive net. Thus, MMD loss is more powerful to quantize the correlation of two models.

Overall, the objective function in our method for incremental FGIR learning is:

$$L = \alpha L_{dist} + \beta L_{mmd} + (L_{ce} + L_{triplet}) \tag{6}$$

4 Experiments

4.1 Datasets and Experimental Settings

Datasets. We demonstrate our method on the Stanford-Dogs [7] and CUB-Birds [22] datasets. For the former, we use the official train/test splits. When training incrementally, we split the first 60 sub-categories (in the order of official classes) as the original classes and images from the remaining 60 sub-categories are added at once or sequentially. For the latter, we choose 60% of images from each sub-category as training set and 40% as testing set. Afterwards, we split the first 100 sub-categories (in the order of official classes) as the original classes and the remaining 100 sub-categories as new classes. The details are shown in Table 1.

Datasets	Training set (#Image/#Class)			Testing set (#Image/#Class)		
	Original cls.	New cls.	Total	Original cls.	New cls.	Total
Stanford-Dogs	6000/60	6000/60	12000/120	4651/60	3929/60	8580/120
CUB-Birds	3504/100	3544/100	7048/200	2360/100	2380/100	4740/200

Table 1: Statistics of the datasets used in our experiments.

Experimental Settings. We use the Recall@K [6][16] (K is the number of retrieved samples), mean Average Precision (mAP), the precision-recall (PR) curve and feature distribution visualizations for evaluation. We adopt the Google Inception [20] to extract image features. During training, the parameters in Inception are updated using the Adam optimizer [8] with a learning rate of 1×10^{-6}, while parameters in fully-connected layers and classifier are updated with a learning rate of 1×10^{-5}. We follow the sampling strategy in [23] and each incremental process is trained 800 epochs. Following the practice in [16][23], the output 512-D features (R^d) from fully-connected layers are used for retrieval. We set the hyper-parameters $\alpha = \beta = 1$ in Eq. 6, and the margin $\lambda = 0.5$ in Eq. 2. Note that we mainly report the results tested on the CUB-Birds dataset in main paper. For the results of the Stanford-Dogs dataset, we show them in the *supplementary material*.

4.2 One-step Incremental Learning for FGIR

We report the results for multiple classes added at once. The process includes two stages. First, we use the cross-entropy and triplet loss to train the network $\mathcal{A}$ on the original classes (100 classes for the CUB-Birds dataset), denoted as $\mathcal{A}(1\text{–}100)$. Second, only images of new classes are added at once to train network $\mathcal{B}$, denoted as $\mathcal{B}(101\text{–}200)$. DIHN [24] has been explored the incremental learning for hashing-based image retrieval. However, its main difference with ours is to depend on the usage of old data as query set to avoid forgetting in their assumption. Considering no previous works for the fine-grained incremental image retrieval, we apply Learning without Forgetting (LwF) [12], Elastic Weight Consolidation (EWC) [9], ALASSO [17], and the incremental learning for semantic segmentation (dubbed L2 loss) [15] for comparison. LwF, EWC, and ALASSO distill knowledge on classifier and network parameters, which are insufficient for incremental FGIR. L2 loss in [15] is more similar with ours where the knowledge is distilled on the classifier and intermediate feature space. Note that cross-entropy and triplet loss (*i.e.* semantic preserving loss) are combined with these three algorithms for fair comparison. The Recall@K are reported in Table 2.

The "w feature extraction" depicts when $\mathcal{A}$ directly extracts features on the new classes without re-training. The "w fine-tuning" depicts using L_{ce} and $L_{triplet}$ to train $\mathcal{A}$ on the new classes but without using L_{dist}. Overall, the network $\mathcal{B}$ suffers from the catastrophic

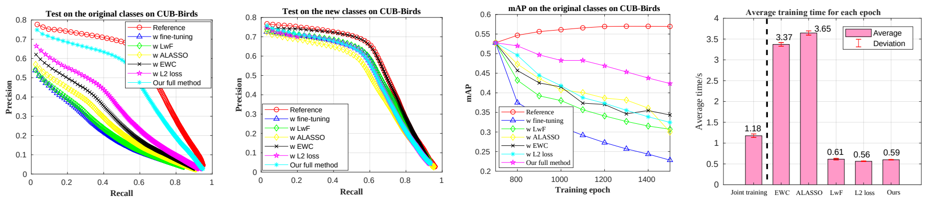
Configurations	Original classes			New classes		
Recall@K(%)	K=1	K=2	K=4	K=1	K=2	K=4
$\mathcal{A}$(1-100) (initial model)	79.41	85.64	89.63	-	-	-
+$\mathcal{B}$(101-200) w feature extraction	-	-	-	47.02	57.44	67.86
+$\mathcal{B}$(101-200) w fine-tuning	53.90	64.56	73.56	76.18	82.56	87.39
+$\mathcal{B}$(101-200) w LwF (L_{dist})	54.92	66.40	75.42	75.76	82.69	86.93
+$\mathcal{B}$(101-200) w ALASSO	56.91	66.65	76.57	72.48	79.50	85.67
+$\mathcal{B}$(101-200) w EWC	62.03	72.16	80.08	73.32	80.92	86.01
+$\mathcal{B}$(101-200) w L2 loss	66.48	75.68	82.67	**77.44**	**83.78**	**88.07**
+$\mathcal{B}$(101-200) w Our method	**74.41**	**82.57**	**88.52**	73.11	80.84	86.64
$\mathcal{A}$(1-200) (reference model)	77.33	85.08	89.03	76.64	83.53	89.12

Table 2: Recall@K (%) of incremental FGIR on the CUB-Birds dataset when new classes are added at once. The best performance in the original class and the new class are in bold.

forgetting issue and has lower performance on the original classes, whereas our method outperforms the others. As for the new classes, other three algorithms outperform ours. For example, " w L2 loss" method achieves on Recall@1 by 4.33% compared to ours (77.44% $\rightarrow$ 73.11%). However, it suffers from significant performance degradation on the original classes with Recall@1 dropping by 12.93% compared to the initial model (79.41% $\rightarrow$ 66.48%). For our method, the Recall@1 on the original classes is 74.41% (dropped by 5.00% from 79.41% of the initial model); the Recall@1 on the new classes is 73.11% compared to the reference model from $\mathcal{A}(1\text{-}200)$ (*i.e.* Recall@1=76.64%).

We report the PR curves and mAP results in Figure 3(a), 3(b), and 3(c), respectively. Overall, when tested on the new classes, all methods share similar trends. When tested on the original classes, our method has better performance although it still has gap to reference performance. For mAP results, the reference results are the same as in Table 2. We utilize the well-trained network $\mathcal{A}$ at epoch=700 as initial model to train $\mathcal{B}$ on the new classes until convergence, we test the mAP of network $\mathcal{B}$ on the original classes. As the curves show, the network trends to degrade its accuracy on the original classes during incremental training.

Furthermore, we explore the influence of the new classes number. Specifically, we choose 100 classes and 25 classes as new categories. The results are reported in Table 4, we observe that larger newly-added classes lead to heavier forgetting. For example, when only 25 new classes are used, the Recall@1 drops from 79.41% to 76.65%, compared to the one drops from 79.41% to 74.41% where 100 new classes are added. Note that the reference models are trained jointly on all classes and tested on the original and new classes separately.



(a) (b) (c) (d)

Figure 3: (a)-(b) denote the PR curves tested on the original classes and new classes. (c) depicts the mAP results for different methods as the training proceeds. We only show the results tested on the original classes. (d) training time comparison during each epoch.

Configurations		Original (1-100)			Added new (101-125)			Added new (126-150)			Added new (151-175)			Added new (176-200)		
Recall@K(%)		K=1	K=2	K=4	K=1	K=2	K=4	K=1	K=2	K=4	K=1	K=2	K=4	K=1	K=2	K=4
$\mathcal{A}$(1-100) (initial model)		79.41	85.64	89.63	-	-	-	-	-	-	-	-	-	-	-	-
LwF algorithm [illegible]	+$\mathcal{B}$(101-125)	57.50	68.05	75.68	79.59	85.88	88.95	-	-	-	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)	42.46	54.03	64.66	62.59	74.83	82.31	70.17	79.67	86.00	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)	40.21	51.57	61.27	47.79	63.10	75.68	56.83	67.33	78.17	81.57	87.27	90.79	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)(176-200)	33.31	44.75	55.38	49.83	63.78	75.85	48.00	60.33	72.33	67.17	75.88	82.91	83.70	89.41	92.94
EWC algorithm [illegible]	+$\mathcal{B}$(101-125)	61.23	70.85	80.04	80.95	86.39	90.82	-	-	-	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)	46.65	56.40	67.54	65.48	77.72	84.01	72.33	80.67	86.67	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)	43.60	54.79	64.70	61.50	72.45	80.44	66.50	75.50	82.67	81.08	85.26	87.77	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)(176-200)	36.82	47.54	59.66	57.99	67.01	76.87	50.67	64.67	77.67	64.15	74.87	81.24	82.02	86.39	90.42
L2 loss algorithm [illegible]	+$\mathcal{B}$(101-125)	67.37	76.27	83.31	80.61	85.54	89.46	-	-	-	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)	58.14	68.31	76.78	72.11	80.44	87.41	73.33	82.17	88.67	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)	53.86	62.03	71.91	60.37	71.43	80.27	66.33	76.67	84.67	81.24	87.27	90.95	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)(176-200)	45.85	56.61	67.75	57.65	71.77	80.95	59.33	70.50	79.13	73.70	83.08	88.94	84.20	89.24	92.10
Our method	+$\mathcal{B}$(101-125)	76.65	83.47	88.86	73.13	82.31	88.44	-	-	-	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)	73.77	81.36	87.80	74.32	83.33	89.29	74.50	83.00	87.83	-	-	-	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)	70.47	78.77	85.97	70.41	80.78	88.78	72.00	79.17	86.83	78.89	86.77	90.26	-	-	-
	+$\mathcal{B}$(101-125)(126-150)(151-175)(176-200)	66.40	75.93	83.14	70.07	80.27	86.22	69.00	78.33	85.50	73.87	83.92	88.78	85.21	89.92	93.28
$\mathcal{A}$(1-200) (reference model)		77.33	85.08	89.03	76.87	84.86	90.48	73.00	80.00	87.67	83.25	88.94	92.29	83.70	90.25	93.78

Table 3: Recall@K (%) results on the CUB-Birds dataset when new classes are added sequentially. "Added new (101-125)" indicates the first 25 classes (101-125) are used as the first part to train the network.

4.3 Multi-step Incremental Learning for FGIR

We split the new classes into 4 groups and added each group sequentially. The training procedures are as follows: the initial model $\mathcal{A}$ is pre-trained on the original classes (1-100), and used as an initial model to train on newly-added classes (101-125) until convergence to produce a new model $\mathcal{B}$(101-125). Afterwards, the newly-trained model $\mathcal{B}$(101-125) is used as an initial model to train on other new classes (126-150) to produce $\mathcal{B}$(101-125)(126-150). This process is repeated until 4 groups of classes are added sequentially.

We compare to three representative methods (we choose EWC rather than ALASSO since EWC obtains higher performance on the CUB-Birds dataset) and report the results in Table 3. The reference performances are achieved by jointly training all the classes, and then tested on each group (including the original classes). Overall, the model suffers from the catastrophic forgetting issue when sequentially training. However, our method achieves a minimal performance degradation. For instance, when 4 groups have been added, the model $\mathcal{B}$(101-125)(126-150)(151-175)(176-200) is tested on the original classes(1-100). The "L2 loss" algorithm Recall@1 drops 79.41%$\rightarrow$67.37%$\rightarrow$58.14%$\rightarrow$53.86%$\rightarrow$45.85%, the average degradation is 8.39%. Our method Recall@1 drops 79.41%$\rightarrow$76.65%$\rightarrow$73.77% $\rightarrow$70.47%$\rightarrow$66.40%. The average performance degrades by 3.25%, which indicates that our method significantly mitigates the forgetting problem. Furthermore, our method has good performance on new classes, which are closer to the reference performance. When the model $\mathcal{B}$(101-125)(126-150)(151-175)(176-200) is tested on new classes (176-200), the results are achieved with Recall@1=85.21%, Recall@2=89.92% and Recall@4=93.28%, respectively, while the reference results are Recall@1=83.70%, Recall@2=90.25% and Recall@4=93.78%.

4.4 Validation with Image Classification

We evaluate the effectiveness of our method on the CIFAR-100 dataset [10] which is the popular benchmark for class-incremental learning in image classification. We split 100 classes into a sequence of 5 tasks, and each task includes 20 classes. In Table 5, the results indicate the average top-1 accuracy of the classes from seen tasks. In the last column, the test set evaluates the classes from all the five tasks. Note that, the 20 classes in the first task (the second column) achieve the same performance, as it has no incremental learning yet. We

Configurations	Original classes			New classes[†]		
Recall@K(%)	K=1	K=2	K=4	K=1	K=2	K=4
$\mathcal{A}$(1-100) (initial model)	79.41	85.64	89.63	-	-	-
+$\mathcal{B}$(101-125) w Our method	76.65	83.47	88.86	73.13	82.31	88.44
+$\mathcal{B}$(101-200) w Our method	74.41	82.57	88.52	73.11	80.84	86.64
$\mathcal{A}$(1-125) (reference model)	77.84	83.94	87.80	79.25	85.54	91.96
$\mathcal{A}$(1-200) (reference model)	77.33	85.08	89.03	76.64	83.53	89.12

Table 4: Recall@K (%) on the CUB-Birds dataset when 25 or 100 new classes are added at once. Correspondingly, [†] indicates the results are tested on different new classes.

Method	Number of new classes				
	20	40	60	80	100
L2 loss	77.3	47.5	40.5	36.6	32.8
EWC	77.3	60.5	50.9	43.3	39.5
LwF	77.3	62.5	52.9	46.2	41.0
Ours	77.3	64.6	55.8	49.2	43.3

Table 5: Average top-1 accuracy of incremental learning for image classification on CIFAR-100 dataset.

Configurations	Original classes			New classes		
Recall@K(%)	K=1	K=2	K=4	K=1	K=2	K=4
$\mathcal{A}$(1-100) (initial model)	79.41	85.64	89.63	-	-	-
+$\mathcal{B}$(101-200) w $L_{ce}+L_{triplet}$	53.90	64.56	73.56	76.18	82.56	87.39
+$\mathcal{B}$(101-200) w $L_{ce}+L_{triplet}+L_{dist}$	54.92	66.40	75.42	75.76	82.69	86.93
+$\mathcal{B}$(101-200) w $L_{ce}+L_{triplet}+L_{mmd}$	73.36	81.25	87.43	73.40	81.60	86.64
+$\mathcal{B}$(101-200) w $L_{ce}+L_{triplet}+L_{dist}+L_{mmd}$	74.41	82.57	88.52	73.11	80.84	86.64
$\mathcal{A}$(1-200) (reference model)	77.33	85.08	89.03	76.64	83.53	89.12

Table 6: Ablation study for different components of loss function

observe that our method outperforms other methods across the tasks. It suggests our method generalizes well to various applications. Notably, our improvement for image retrieval is more significant than that for image classification. The reason is that the proposed MMD loss is imposed on the feature representation, which largely benefits the metric learning for image retrieval. This also explains why our method is focused mainly on image retrieval.

4.5 Training Time Comparison

We compare the average training time on the CUB-Birds dataset when 100 new classes are added at once. The results are shown in Figure 3(d). Note that all models in five methods are starting from the same initial model trained on the original 100 classes as initialization. The reference time is from joint training where the initial model is trained on all classes. The other four methods are incrementally learning the new classes only. We observe that our method saves more time by 50% as expected. EWC and ALASSO algorithms take more time than reference because the gradients computation during back-propagation process is time-consuming.

4.6 Components Analysis

Ablation Study. We have done an ablation study on the CUB-Birds dataset when multiple classes are added at once. Note that the component "$L_{ce}+L_{triplet}$" comprises our baseline performance, thus we analyze the different loss items in Eq. 6. We can observe the influence of difference components for the original and new classes. The results are shown in Table 6.

Hyper-parameters Sensitivity Analysis. We explore the sensitivity of hyper-parameters α, β in Eq. 6, which affect significantly the trade-off performance. We conduct this experiment on the CUB-Birds dataset. As shown in Table 7, we find that the incrementally-trained model is more sensitive to β than α. For instance, when α is set as 0.1, but β changes from 0.1 to 1, model $\mathcal{B}$ performs better on the new classes and significantly retains its previous performance. However, this obvious trend cannot be observed when β is set as 0.1, but α

changes from 0.1 to 1 where the model $\mathcal{B}$ performs almost the same on the original and new classes. Finally, if α=β=1, the incrementally-trained model $\mathcal{B}$ keeps a better trade-off performance between the original and the new classes.

Configurations	Original classes			New classes		
Recall@K (%)	K=1	K=2	K=4	K=1	K=2	K=4
$\mathcal{A}$(1-100) (initial model)	79.41	85.64	89.63	-	-	-
+$\mathcal{B}$(101-200) (α=0.1,β=0.1)	56.53	66.31	75.59	77.52	83.82	88.15
+$\mathcal{B}$(101-200) (α=0.1,β=1)	73.31	82.00	87.14	72.77	80.92	87.14
+$\mathcal{B}$(101-200) (α=0.1,β=10)	79.58	85.76	90.47	49.50	61.51	70.59
+$\mathcal{B}$(101-200) (α=1,β=0.1)	55.81	67.25	75.59	77.02	83.91	87.90
+$\mathcal{B}$(101-200) (α=1,β=1)	$\underline{74.41}$	$\underline{82.57}$	$\underline{88.52}$	$\underline{73.11}$	$\underline{80.84}$	$\underline{86.64}$
+$\mathcal{B}$(101-200) (α=1,β=10)	79.41	86.31	90.51	48.82	61.09	71.05
$\mathcal{A}$(1-200) (reference model)	77.33	85.08	89.03	76.64	83.53	89.12

Table 7: Sensitivity analysis of the hyper-parameters α, β. The better trade-off performance of the hyper-parameters are underlined.

5 Conclusion

In this paper, for the first time, we have exploited incremental learning for fine-grained image retrieval in several scenarios for increasing numbers of image categories when only images of new classes are used. To overcome the catastrophic forgetting, we adopted the distillation loss function to constrain the classifier in the original network and the incremental classifier in the adaptive network. Moreover, we introduced a regularization function, based on Maximum Mean Discrepancy (MMD), to minimize the discrepancy between features of newly added classes from the original and the adaptive network. Comprehensive and empirical experiments on two fine-grained datasets show the effectiveness of our method that is superior over existing methods. In the future, it is promising to investigate incremental learning between different fine-grained datasets for image retrieval.

6 Acknowledgment

This work is supported by LIACS MediaLab at Leiden University, China Scholarship Council (CSC No. 201703170183), and the FWO project "Structure from Semantic" (grant No. G086617N). We would like to thank NVIDIA for the donation of GPU cards.

References

[1] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. JMLR: Workshop and Conference Proceedings, 2016.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[3] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *NIPS*, pages 1205–1213, 2012.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, pages 437–452, 2018.

[6] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, 2010.

[7] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop*, volume 2, 2011.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] 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. *In PNAS*, 114(13):3521–3526, 2017.

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

[11] Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. *arXiv preprint arXiv:1802.01483*, 2018.

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017.

[13] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NIPS*, pages 136–144, 2016.

[14] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, pages 6467–6476, 2017.

[15] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *ICCV Workshops*, 2019.

[16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016.

[17] Dongmin Park, Seokil Hong, Bohyung Han, and Kyoung Mu Lee. Continual learning by asymmetric loss approximation with single-side overestimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3335–3344, 2019.

[18] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, pages 2990–2999, 2017.

[19] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ECCV*, pages 3400–3409, 2017.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[21] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[22] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[23] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *CVPR*, pages 5022–5030, 2019.

[24] Dayan Wu, Qi Dai, Jing Liu, Bo Li, and Weiping Wang. Deep incremental hashing network for efficient image retrieval. In *CVPR*, pages 9069–9077, 2019.

[25] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019.

[26] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *ICCV*, pages 6619–6628, 2019.

[27] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *CVPR*, pages 2272–2281, 2017.

[28] Xin Yao, Tianchi Huang, Chenglei Wu, Rui-Xiao Zhang, and Lifeng Sun. Adversarial feature alignment: Avoid catastrophic forgetting in incremental task lifelong learning. *Neural computation*, 31(11):2266–2291, 2019.

[29] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *ICCV*, pages 2759–2768, 2019.

[30] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S Davis. M2kd: Multi-model and multi-level knowledge distillation for incremental learning. *arXiv preprint arXiv:1904.01769*, 2019.