Semantic Drift Compensation for Class-Incremental Learning

Lu Yu[1,2], Bartłomiej Twardowski[2], Xialei Liu[2], Luis Herranz[2], Kai Wang[2],
Yongmei Cheng[1], Shangling Jui[3], Joost van de Weijer[2]
[1] School of Automation, Northwestern Polytechnical University, Xi'an, China
[2] Computer Vision Center, Universitat Autonoma de Barcelona, Barcelona, Spain
[3] Huawei Kirin Solution, Shanghai, China
{luyu,btwardowski,xialei,lherranz,kwang,joost}@cvc.uab.es,
chengym@nwpu.edu.cn, jui.shangling@huawei.com

Abstract

Class-incremental learning of deep networks sequentially increases the number of classes to be classified. During training, the network has only access to data of one task at a time, where each task contains several classes. In this setting, networks suffer from catastrophic forgetting which refers to the drastic drop in performance on previous tasks. The vast majority of methods have studied this scenario for classification networks, where for each new task the classification layer of the network must be augmented with additional weights to make room for the newly added classes.

Embedding networks have the advantage that new classes can be naturally included into the network without adding new weights. Therefore, we study incremental learning for embedding networks. In addition, we propose a new method to estimate the drift, called semantic drift, of features and compensate for it without the need of any exemplars. We approximate the drift of previous tasks based on the drift that is experienced by current task data. We perform experiments on fine-grained datasets, CIFAR100 and ImageNet-Subset. We demonstrate that embedding networks suffer significantly less from catastrophic forgetting. We outperform existing methods which do not require exemplars and obtain competitive results compared to methods which store exemplars. Furthermore, we show that our proposed SDC when combined with existing methods to prevent forgetting consistently improves results. [1]

1. Introduction

Future learning machines should be able to adapt to an ever-changing world. They should continuously learn new tasks without forgetting previously learned tasks. Other than the generally applied setup, where training data for all tasks is simultaneously available, in continual learning tasks are learned in a consecutive manner. At each moment the algorithm has only access to the data of a single task. For deep neural networks, one could finetune the network on the data of the latest task. However, in the absence of training data of previous tasks, the network suffers from *catastrophic forgetting* [26]. This refers to a drastic drop in performance on previous tasks. Continual learning studies strategies to mitigate the impact of catastrophic forgetting [17, 19, 30].

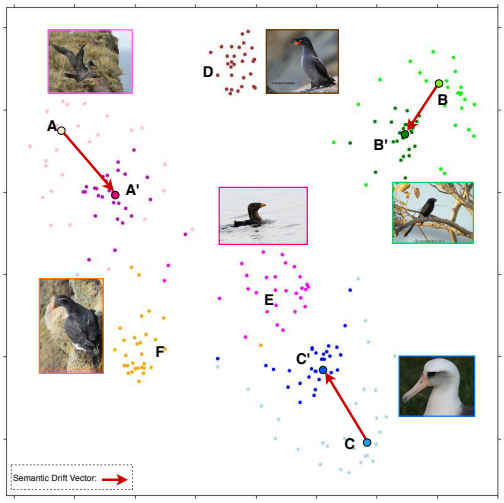

Figure 1: T-SNE visualization of embedding space after finetuning. A, B, C indicate prototypes of task 1 after training task 1; A',B',C' and D,E,F respectively for task 1 and 2 after training for task 2. The semantic drift (indicating forgetting) from task 1 is given by red vectors. Our method estimates this and compensates the prototypes accordingly.

Continual learning has explored a variety of strategies to prevent networks from forgetting previously learned tasks. Li et al. [19] propose a method called learning without forgetting (LwF). They use the same data to supervise learning

[1] Code available at https://github.com/yulu0724/SDC-IL.

of the new tasks and to provide unsupervised output guidance on the old tasks to prevent forgetting. Elastic weight consolidation (EWC) [17] estimates the Fisher matrix to weight a regularization term favouring changes to neurons which were found to be less important in previous tasks, and which prevents the relevant neurons from adapting to the new task. Further research on continual learning includes regularization terms [1, 20], sub-network selection by mask learning [22, 23, 35], and the use of exemplars [21, 30].

Many of the early works in continual learning considered a task-incremental learning (task-IL) scenario [38], in which the network has access to the task-ID at inference time [1, 17, 19, 25, 35]. Recently, more works consider the more difficult class-incremental learning (class-IL) [2, 10, 15, 20], where no task-ID is available at inference. The main additional challenge for class-IL is the class imbalance between old versus new tasks. This is addressed by storing data of previous tasks [5, 15, 44]. In this paper, we propose a new method for class-incremental learning. We consider the difficult scenario where no data of previous tasks can be stored. The importance of continual learning algorithms which do not require any storage is growing in a world where data privacy and security are fundamental for many users, and are controlled by government legislation.

The discussed previous works all study continual learning in classification networks. For these networks, new weights have to be added to accommodate for the newly added classes. Instead, we perform class-incremental learning for embedding networks which naturally allow for the inclusion of new classes, and do not require network changes for new classes. Embedding networks map data to embedding spaces in which distances correspond to semantic dissimilarities between data points [8]. They are typically used for image retrieval [40], face recognition [33],etc. However, they can also be used for classification when combined with, for example, a nearest class mean classifier [27].

In this paper, we show that embedding networks suffer significantly less from catastrophic forgetting than classification networks. We also propose a new method called *semantic drift compensation*. Instead of preventing drift, which most existing methods do, our method estimates the drift of previous tasks during the training of new tasks (see Fig. 1). We show that an estimate of the semantic drift in previous tasks can be used to compensate for it, thereby improving performance. We evaluate embedding networks for image classification by using the nearest class mean (NCM) classifier [27]. We will refer to the class embedding mean with the term *prototype*. We will show how the drift of prototypes learned in previous tasks can be approximated while only having access to data of the current task. Furthermore, the proposed method can easily be combined with existing methods that prevent forgetting, such as EWC [17], LwF [19], or MAS [1], to further improve results.

2. Related Work

Continual Learning. Regularization-based methods optimize network parameters on the current task while preventing the drift of already consolidated weights. Learning without forgetting (LwF) [19] adapts a learned model to new tasks while retaining the knowledge gained earlier with a regularization term on probabilities. EWC [17] and a variant R-EWC [20] include a regularization term on the weights that forces parameters of the current network to remain close to the parameters trained for the previous tasks. Zenke et al. [47] propose to compute the consolidation strength of synapses (represented by the network weights) in an online manner, and extends them with a memory to accumulate task-relevant information. Aljundi et al. [1] compute the weight importance in a unsupervised manner.

Rehearsal-based methods store a small subset of training data from previous tasks in order to prevent catastrophic forgetting. These exemplars are combined (i.e. rehearsed) with the current task data so that the network parameters are jointly optimized. Some existing works use a distillation loss to prevent forgetting [6, 21, 30, 15]. In [44], bias correction is proposed to solve the problem of the data imbalance between the old and new classes especially for large scale datasets. Another alternative is to learn a generative model of previous tasks, and generate synthetic samples (i.e. pseudo-rehearsal) that are combined as usual [43, 36].

We consider continual learning in the challenging class-incremental learning (class-IL) scenario. Some aforementioned methods can be applied directly to class-IL by adjusting the network architecture, e.g. [2, 32]. However, they do not scale with the number of classes—as the network requires constant expansion. In [10] the presented approach uses an attention distillation loss to penalize the changes on attention maps, which helps to retain information of the previous classes, whenever new classes are added. Three strategies to prevent forgetting: cross normalization, less-forgetting constrain and inter-class separation with saved exemplars from previous tasks are introduced in [15]. A method applicable for a class-incremental problem was presented in [20], where EWC is combined with a reparametrization of the network in form of a factorized rotation what results in a better performance on previous tasks. The authors of work [44] discussed the problems of large number of classes and visual similarities between new versus old classes. They proposed a bias correction of network's outputs for new classes by a linear models. A distillation loss was used in [5] with an additional exemplars set in order to prevent forgetting for class-incremental learning. Finally, [3] exploits a dual memory to reduce the negative effect of catastrophic forgetting in image recognition. A model integrates knowledge distillation and retrospection along with the path selection strategy is proposed

to overcome catastrophic forgetting in [29].

Our method differs with previous work in two principal aspects. To train new tasks, we use an embedding network with a metric learning loss rather than a classification loss. And secondly, all of the methods discussed above focus on preventing forgetting during the learning of new tasks. Our method does not focus on preventing forgetting, but instead proposes to estimate the drift of features that happens due to the learning of new tasks. Having an approximation of the drift, we can compensate the prototypes of previous tasks.

Deep Metric Learning. Siamese networks [8] were first proposed to learn embeddings for face verification. Initially, they used contrastive loss, which ensures that pairs from the same category are close and pairs from different categories are far. Triplet networks [13, 39] were proposed to address the limitations of contrastive loss. The inputs are an anchor image, a positive and a negative image. The aim of a triplet network is to learn embeddings for which the distance between the similar pairs is smaller than the distance between the dissimilar pairs. A direct extension to this method is a quadruplet network [7], where the number of points is extended to four and three pairs are used at once in a loss function. Further improvements to the embedding learning include: constraining the angle at the negative point of triplet triangles [40] and exploiting all selected pairs information in a multi-similarity loss function proposed in [41].

The advantages of embedding networks, when compared to classification networks, is an ongoing discussion. Recent works have pointed out serious shortcomings of classification networks, mostly attributed to the cross-entropy loss (which is based on a softmax operation). Embedding networks were found to be more robust to the exposure of adversarial examples, and better in the detection of out-of-distribution examples [24, 31]. Furthermore, deep embeddings were reported to be superior to classification networks for transfer learning [34] and preliminary results suggest that they might be less prone to catastrophic forgetting [45].

3. Continual Learning for Embeddings

We consider a class-incremental learning setup where a network learns several tasks, each task containing a number of new classes. During the training of task t we only have access to data D^t which contains pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is an image of class $y_i \in C^t$. For each task we consider that there is data of a limited set of classes $C^t = \{c^t_1, c^t_2 ..., c^t_{m^t}\}$, where m^t is the number of classes in task t. We consider the generally studied case where there is no overlap between the classes of different tasks: $C^t \cap C^s = \varnothing$ for $t \neq s$. After training all n tasks we evaluate the learned embedding on all classes $C = \bigcap_i C^i$. As other class-incremental methods, we consider a *task-agnostic* setting where the algorithm has no access to the task label at test time.

3.1. Embedding Networks

We start by explaining the training of an embedding network for a single task. Embedding networks map data into a low-dimensional output where distance represents the semantic dissimilarity between the images [4, 8]. They simultaneously perform feature extraction as well as metric learning. In the learned embedding space it is possible to apply a simple metric, such as L2-distance, to determine the similarity between the original images.

Chopra et al. [8] proposed to use Siamese networks with the contrastive loss as an objective function. This loss needs related and unrelated pairs of images, and ensures that the distance between related pairs will be low, and the distance between unrelated pairs larger than a margin. For some embeddings it was found that the contrastive loss is hard to train, and other losses have been proposed. The triplet loss is proposed by Hoffer et al. [13] based on the work of Wang et al. [39]. The objective function forces the negative instance to be further away from the anchor than the positive ones (plus a margin m). The triplet loss is given by:

$$\mathcal{L}_T = \max(0, d_+ - d_- + m), \tag{1}$$

where d_+ and d_- are the Euclidean distances between the embeddings of the anchor $\mathbf{z}_a$ and the positive instance $\mathbf{z}_p$ and the negative instance $\mathbf{z}_n$ respectively. Here $\mathbf{z}_i = F(\mathbf{x}_i)$ is the output embedding for image $\mathbf{x}_i$.

Having trained an embedding network we can use the embedding space for classification. We will use nearest class mean (NCM) classifier which is defined as:

$$c^*_j = \operatorname*{argmin}_{c \in C} \operatorname{dist}(\mathbf{z}_j, \boldsymbol{\mu}_c) \tag{2}$$

$$\boldsymbol{\mu}_c = \frac{1}{n_c} \sum_i [y_i = c]\, \mathbf{z}_i \tag{3}$$

where n_c is the number of training images for class c and $[P] = 1$ if P is true, and 0 otherwise. We will refer to $\boldsymbol{\mu}_c$ as the *prototype* of class c. The terminology of prototypes was also used in several works [37, 45] to refer to class representative points in an embedding space.

3.2. Softmax Classifier versus Embedding Learning

The conventional approach to image classification is a softmax classifier trained with a cross-entropy loss. Due to its success, it has been the natural starting point for methods studying continual learning for image classification. However, there are several fundamental drawbacks of the softmax classifier which might limit its application to continual learning. First, network outputs are tightly coupled with prediction classes. Whenever new object classes are added, structural changes to the architecture are required, i.e. new neurons added to accommodate for the new classes. In

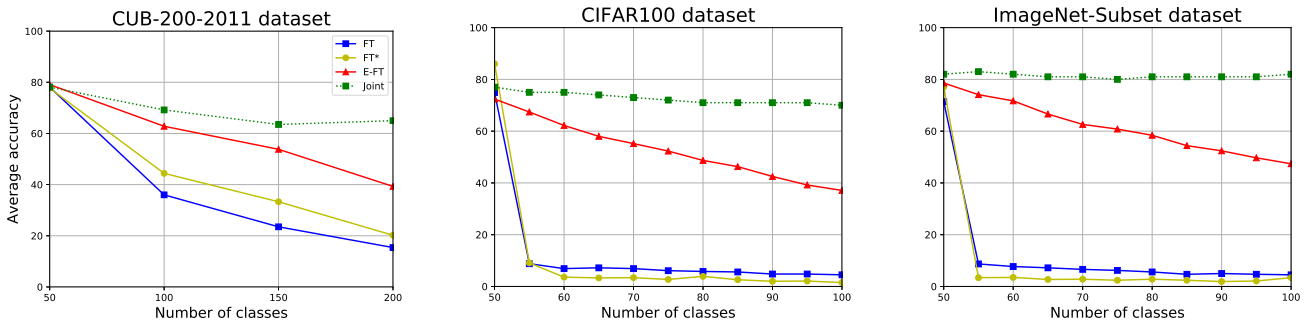

Figure 2: Average incremental accuracy for finetuning trained and evaluated with softmax (FT) , finetuning trained with softmax but classified with NCM (FT*), and trained with metric learning loss and classified with NCM (E-FT). In addition, we report joint training (Joint). The results show that continual learning with embedding networks suffers from significantly less forgetting.

a class-incremental setting, this results in creating a new output layer (head) for each task [5]. Second, in order to acquire a final prediction from a multi-head network, outputs need to be aggregated.Third, the updated model is expected to be biased in the predictions that favor new classes [15, 44]. All of the aforementioned issues can be mitigated to some extent. However, an adaptation process of softmax-based classifiers for class-incremental learning is challenging and harder for longer task sequences. Using embedding networks for continual learning has advantages. New classes can be naturally added without any architectural changes. While learning new tasks, the network gets fine-tuned to the new data distribution. However, metric learning methods do not require information about classes directly. This is used only to make a proper preparation of the input data, i.e. pairs of positive and negative examples, and thus, the architecture stays intact.

In order to compare classification and embedding networks in a continual learning setting, we compare both when applying finetuning to adjust to new tasks; a setting which is known to lead to catastrophic forgetting for classification networks. The softmax classifier uses new heads for the incremental classification. During testing we compute the probability of each head and take the maximum as the true prediction (called FT). As a second option, we consider performing NCM on the average-pooled output of block 5 of the trained ResNet network, which has the same dimensionality as our embedding network (denoted by FT*). This technique was also used in iCaRL [30]. The embedding network (trained with triplet loss [13]) represents classes with a prototype and performs NCM for classification, and is denoted by E-FT. After convergence on a task is reached, we continue training on a new task with standard stochastic gradient descent, and repeat this until all tasks are learned.

The results of the comparison on three datasets are presented in Fig. 2. Let $a_{k,j} \in [0,1]$ be the accuracy of the j-th task ($j \leq k$) after training the network sequentially for k tasks. Then, the average incremental accuracy at task k is defined as $A_k = \frac{1}{k}\sum_{j=1}^{k} a_{k,j}$ [2]. We observe that the softmax leads to better classification performance for the first task, as mentioned in [13, 14]. However, the average incremental accuracy of softmax drops significantly when continually adding tasks. This is especially evident for the CIFAR100 and ImageNet-Subset datasets. Using the softmax and classification with NCM on embedding (FT*) suffers similarly from catastrophic forgetting. The finetuning with metric loss (E-FT) significantly better results are obtained, with absolute gains of 23.9% on CUB-200-2011, with 32.6% on CIFAR100, and 42.9% on ImageNet-Subset. In conclusion, a well-established method for a single-task learning (i.e. softmax) is not optimal when it is used in a continual learning setting. Classification with NMC and embeddings learned with metric losses suffer significantly less from catastrophic forgetting as we observed. Therefore, we propose to use them for continual learning. In the following, we will develop additional methods to further improve continual learning for embedding networks.

3.3. Regularizing Embedding Networks

The problem of catastrophic forgetting in continual learning has been extensively studied for classification networks [6, 17, 19, 20, 30, 36, 43]. To our knowledge, there is no prior work to prevent forgetting the knowledge from the previous tasks on embedding networks. In the following, we adapt several existing techniques to embeddings. We will indicate the variant for embeddings the following notation convention: we append an E (for embedding) to the name of the original method designed for a classification network, e,g, E-LwF would be LwF (Learning without Forgetting) adapted for an embedding network.

Finetuning (E-FT) Described in Sec. 3.2 and used as a baseline. For all experiments triplet loss [13] is used.

Alignment Loss (E-LwF) [19] This method was proposed on classification networks. It aims to match the softmax

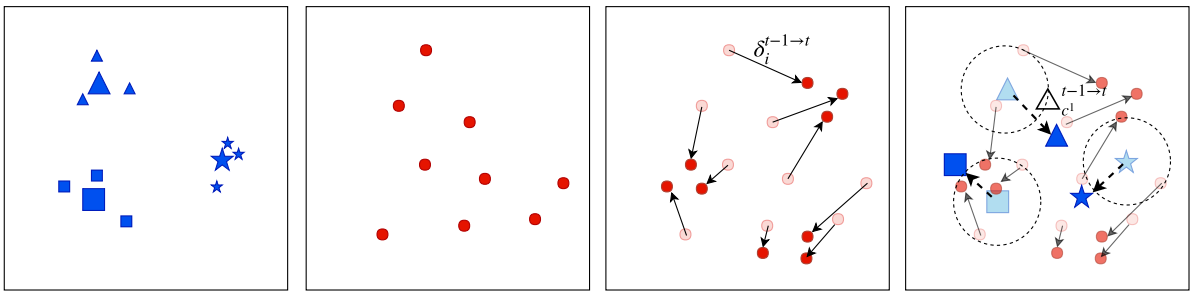

Figure 3: Illustration of semantic drift compensation. (a) Data and prototypes of three classes of task 1 after training task 1. (b) Data of task 2 after training task 1. (c) Drift of data of task 2 while training task 2. This results in a sparse vector field of drift vectors. (d) This vector field is used to approximate the drift of the prototypes of task 1.

output of the network of previous models on current data. Instead, on embedding networks, we constrain the parameters drift by minimizing the distance between the output embeddings of image $\mathbf{x}_i$ during training the current task ($\mathbf{z}_i^t$) with respect to its embedding in the previous task ($\mathbf{z}_i^{t-1}$), similar as [46]. This leads to the following loss:

$$\mathcal{L}_{LwF} = \left\| \mathbf{z}_i^t - \mathbf{z}_i^{t-1} \right\|, \tag{4}$$

where $\|.\|$ refers to the Frobenius norm.

E-EWC [17] This method was proposed on classification networks to keep the network parameters close to the optimal parameters for the previous task while training the current task. This can also be leveraged on embedding networks. The function that we minimize in EWC is:

$$\mathcal{L}_{EWC} = \sum_p \frac{1}{2} \mathcal{F}_p^{t-1} (\theta_p^t - \theta_p^{t-1})^2, \tag{5}$$

where $\mathcal{F}^{t-1}$ is the Fisher information matrix computed after the previous task $t-1$ was learned, and the summation goes over all parameters θ_p of the network.

E-MAS [1] This method was proposed to accumulate an importance measure for each parameter of the network based on how sensitive the predicted output function is to a change in this parameter, which can be directly applied to embeddings. The function that we minimize in MAS is:

$$\mathcal{L}_{MAS} = \sum_p \frac{1}{2} \Omega_p (\theta_p^t - \theta_p^{t-1})^2, \tag{6}$$

where Ω_p is estimated by the sensitivity of the squared l_2 norm of the function output to their changes.

These losses can be added to the metric learning loss to prevent forgetting while training embeddings continually:

$$\mathcal{L} = \mathcal{L}_{ML} + \gamma \mathcal{L}_C, \tag{7}$$

where $C \in \{LwF, EWC, MAS\}$, γ is trade-off between the metric learning loss and the other losses.

4. Semantic Drift Compensation

Embeddings suffer from drift when learned in a sequential manner. When data from previous tasks is not available, using the original prototype in the NCM usually results in a performance drop. We aim at reducing the error that drift causes and propose a drift compensation to update previously computed prototypes. The main idea is to estimate the unknown drift according to the known drift of the current data during the training of the current task.

4.1. Computation of the Semantic Drift

In Sec. 3.1, we discussed how prototypes of the classes can be computed for a single task. Here we extend this theory to the continual learning setting. We refer to the prototype mean as $\boldsymbol{\mu}_{c^s}^t$ which is the mean for class c^s after learning task t computed with Eq. 3. Class c^s is learned during task s (we removed the sub-index i from c_i^s for conciseness). When $t > s$ we have no access to data of task s and we cannot compute the true prototype mean (by applying Eq. 3 again). We call the difference between the true class mean and the estimate of the class mean the *semantic drift*:

$$\triangle_{c^s}^{s\to t} = \boldsymbol{\mu}_{c^s}^t - \boldsymbol{\mu}_{c^s}^s, \tag{8}$$

Since we cannot compute $\boldsymbol{\mu}_{c^s}^t$ directly we have to find alternative ways to approximate the semantic drift $\triangle_{c^s}^{s\to t}$. We start by proposing a method to compute $\triangle_{c^s}^{t-1\to t}$ from which we can then derive $\triangle_{c^s}^{s\to t}$.

When training task t we do not have access to the data of task s and therefore we cannot observe how the embeddings $\mathbf{z}_i$, for which $y_i \in C^s$, drift during training of task t. However, we can measure the drift of the current data during the training of task t.

$$\boldsymbol{\delta}_i^{t-1\to t} = \mathbf{z}_i^t - \mathbf{z}_i^{t-1}, \; y_i \in C^t, \tag{9}$$

here we use the notation $\mathbf{z}_i^t$ to refer to the embedding of point i after training task t. At the start of training task t we have access to $\mathbf{z}_i^{t-1}$ which is the embedding of data point i after training task $t-1$.

We propose to approximate the semantic drift $\triangle_{c^s}^{t-1\to t}$ from the sparse vector field $\boldsymbol{\delta}_i^{t-1\to t}$. We do this by interpolating this vector field at the prototype location $\boldsymbol{\mu}_{c^s}^{t-1}$ using:

$$\hat{\triangle}_{c^s}^{t-1\to t} = \frac{\sum_i [y_i \in C^t]\, w_i \boldsymbol{\delta}_i^{t-1\to t}}{\sum_i [y_i \in C^t]\, w_i}, \tag{10}$$

with

$$w_i = e^{-\frac{\left\|\mathbf{z}_i^{t-1} - \boldsymbol{\mu}_{c^s}^{t-1}\right\|^2}{2\sigma^2}}, \tag{11}$$

where σ is the standard deviation of the Gaussian kernel.

In summary, as shown in Fig. 3, for all data points in task t we can monitor the semantic drift during the training of task t. This results in a set of drift vectors $\boldsymbol{\delta}_i^{t-1\to t}$ which are used to compute the semantic drift of all previously learned prototypes $\hat{\boldsymbol{\mu}}_{c^s}^{t-1}$. This is done by assigning a weight to the drift vectors according to their distance to the prototypes, and computing the prototype drift as a weighted mean of the nearby drift vectors (with Eq. 10).

We can apply the semantic drift compensation (SDC):

$$\hat{\boldsymbol{\mu}}_{c^s}^{t} = \boldsymbol{\mu}_{c^s}^{s} + \hat{\triangle}_{c^s}^{s\to s+1} + ... + \hat{\triangle}_{c^s}^{t-1\to t} \tag{12}$$

where total compensation is the sum of the compensations which where measured during all previous steps. Normally a recursive scheme would be applied where you update all previously learned prototypes at each new task:

$$\hat{\boldsymbol{\mu}}_{c^s}^{t} = \hat{\boldsymbol{\mu}}_{c^s}^{t-1} + \hat{\triangle}_{c^s}^{t-1\to t}. \tag{13}$$

4.2. Regularized Semantic Drift Compensation

Many approaches to continual learning have focused on preventing the network from using parameters which were found to be relevant for previous tasks [1, 17, 19]. Our method is based on an entirely different approach where we accept the fact that if we share parameters between the tasks, and we want all tasks to be able to improve (i.e. back-propagate) to all these parameters, this will result in a drift for the previously learned tasks. Approximating this drift allows us then to compensate for it. Since our approach applies a different methodology to prevent forgetting, it is interesting to see if it is complementary to these other methods. We therefore propose to combine existing methods (E-LwF, E-EWC and E-MAS) with semantic drift compensation and will evaluate this in the experimental results.

To provide an illustration of SDC, we conduct experiments on MNIST with a 2-dimensional embedding. We divide the ten classes into two disjoint tasks randomly. In Fig. 4 we show examples of the drift vectors which are estimated by SDC in the case of E-FT and E-EWC[2]. We can see that the approximated drift vectors improve the locations of the prototypes to be closer to the correct positions. As a result, the accuracy of the overall method remains higher while training new tasks.

[2]Examples of the other two methods are in the supplemental material as well as all implementation details and results in tabular form.

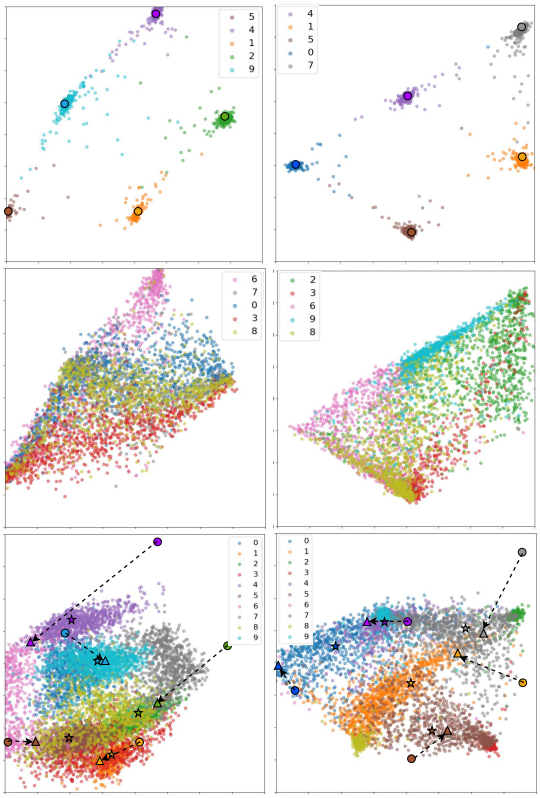

Figure 4: **Visualization of SDC with E-FT (left) and E-EWC (right).** Top figures represent the embedding of 5 classes of task 1 after training task 1; middle ones represent the embedding of another 5 classes of task 2 after training task 1; bottom ones show the embeddings of both tasks after training task 2. The saved prototypes of the previous task (indicated by circle) are corrected by SDC to new positions (indicated by triangle). Note that the corrected prototypes are closer to the real mean (indicated by star). The dotted arrows are the SDC vectors.

5. Experiments

In this section, we follow the protocol for evaluating incremental learning [1, 20, 30]. For the multi-class datasets, the classes are arranged in a fixed random order. Each method is trained in a class-incremental way on the available data and evaluated on the test set. For the evaluation metric we report: *average incremental accuracy* [2] which is the average accuracy of only those classes that have already been trained. We also report *average forgetting* [6] on CIFAR100 and ImageNet-Subset dataset.

Datasets. We have used the following datasets: CUB-200-2011 [42], Flowers-102 [28], Caltech-101 [11], CIFAR100 [18], and ImageNet-Subset containing 100 randomly chosen classes from ImageNet [9]. All are divided by classes into tasks randomly. CUB-200-2011 has 200 classes of birds with $11,788$ images in total. Flowers-102 consists

Table 1: Average incremental accuracy for fine-grained datasets.

	CUB-200-2011						Flowers-102					
	T1	T2	T3	T4	T5	T6	T1	T2	T3	T4	T5	T6
E-Pre	78.5	69.1	62.1	58.1	54.7	52.1	90.9	77.5	77.7	76.1	75.2	73.6
E-Fix	84.1	70.6	61.7	56.9	53.5	50.3	98.2	83.6	82.8	80.1	78.4	76.9
FT	79.7	34.7	23.3	17.5	12.6	11.4	99.1	43.9	32.2	24.2	18.8	15.3
E-FT	84.1	73.6	62.5	54.2	43.0	37.4	98.2	76.0	59.3	50.2	42.4	29.1
E-FT+SDC	84.1	**75.5**	**69.5**	**63.6**	**57.5**	**49.3**	98.2	**85.5**	**74.1**	**61.9**	**49.8**	**35.3**
LwF	79.7	54.8	40.8	33.7	27.0	23.6	99.1	69.7	67.4	60.0	49.9	46.6
E-LwF	84.1	74.0	64.8	60.0	55.5	51.4	98.2	85.3	81.6	77.2	69.3	63.5
E-LwF+SDC	84.1	**74.4**	**65.9**	**61.3**	**57.3**	**52.7**	98.2	**86.1**	**82.2**	**79.6**	**74.7**	**69.7**
EWC	79.7	43.4	26.6	20.0	15.5	12.6	99.1	65.2	40.9	33.8	23.7	22.1
E-EWC	84.1	73.6	65.0	61.6	55.0	54.2	98.2	86.2	84.9	82.9	80.9	79.6
E-EWC+SDC	84.1	**74.8**	**67.4**	**62.8**	**58.2**	**56.4**	98.2	**87.6**	**86.9**	**86.0**	**84.2**	**83.9**
MAS	79.7	49.4	37.8	31.4	25.0	22.3	99.1	71.1	61.3	57.9	52.1	44.8
E-MAS	84.1	**72.5**	65.1	60.4	54.7	51.9	98.2	82.9	79.1	76.6	73.9	70.9
E-MAS+SDC	84.1	71.9	**65.3**	**61.1**	**57.3**	**54.4**	98.2	**83.1**	**80.7**	**78.8**	**76.8**	**76.0**

of 102 flower categories of which we randomly choose 100 with 8189 images in total. CIFAR100 contains 600 images for each class. ImageNet-Subset has 129, 156 images in total. Caltech-101 composes of images of objects belonging to 101 widely varied categories.

Implementation Details. All models are implemented with Pytorch. Adam [16] is used for the optimization. ResNet-18 [12] is adopted as the backbone network pre-trained from ImageNet for CUB-200-2011[3] and Flowers-102. For CIFAR100 and ImageNet-Subset version of ResNet-32 and ResNet-18 were used respectively, as in [15], but without pre-training. A triplet loss [13] is used in all reported experiments[4]. The training images (all resized to 256×256, except for CIFAR100 to 32×32) are randomly cropped and flipped. We use a mini-batch size of 32. We train our models with learning rate $1\mathrm{e}{-5}$ for 50 epochs on CUB-200-2011, $1\mathrm{e}{-4}$ for 20 on Flowers-102, and $1\mathrm{e}{-6}$ for 50 on CIFAR100 and ImageNet-Subset. The final embeddings of 512 dimensions are normalized. The trade-off between the E-LwF, E-EWC, E-MAS and triplet loss is 1, $1e7$ and $1e6$ respectively. We choose a fixed $\sigma = 0.3$ to compute the weights of the SDC vectors for all datasets, except for CIFAR100 we choose $\sigma = 0.2$.

5.1. Classification with Embedding Networks

To evaluate the effectiveness of our method, we conduct experiments on two fine-grained datasets: CUB-200-2011 and Flowers-102[5] on the six-task scenario. Results are shown in Table 1. Here we analyze the average results after training the last task (T6).

When comparing the various methods to prevent forgetting trained with softmax (LwF/EWC/MAS) to those applied on embedding network (E-LwF/E-EWC/E-MAS), we observe an enormous gain in performance, showing that embedding networks are less prone to catastrophic forgetting. We also add results for NME on the pre-trained ImageNet model (E-Pre) and the model fixed after training the first task (E-Fix). We can see the best overall accuracy with SDC on two datasets all outperform these two baselines. Furthermore, it can be seen that E-LwF, E-EWC and E-MAS outperform E-FT on both datasets. For example E-EWC obtains a gain of 16.8% on birds and of 50.5% on flowers. The performance of all three methods to prevent forgetting is comparable. Next, we can observe that SDC improves the results of all methods even further, especially for E-FT with 11.9% on birds and 6.2% on flowers. Finally, it is interesting to observe that simple finetuning on embedding networks (E-FT) obtains superior results than LwF, EWC and MAS on birds. When further combined with semantic drift compensation, it further improves over these methods.

To further analyze if SDC prevents the drift of prototypes, we measure the average distance between the real class-mean and the prototypes (before and after application of SDC). The results are provided in Fig. 5. We observe that SDC reduces the drift of the prototypes.

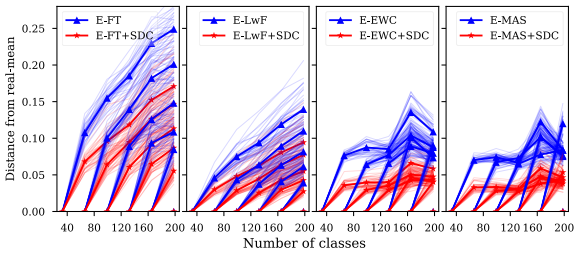

Figure 5: Impact of SDC on the distance between real-mean and prototypes for CUB-200-2011 dataset over tasks. Each line represents a single class. Bold lines represent the mean value of all classes. The graph confirms that SDC correctly compensates for part of the drift of the prototypes.

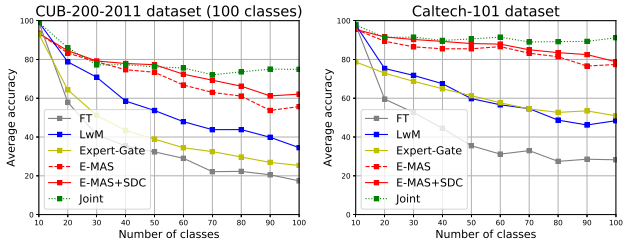

Figure 6: Average incremental accuracy. Comparison of ten-task on CUB-200-2011 (100 classes) and Caltech-101.

[3]Results on CUB-200-2011 pretrained from ImageNet without birds do not change much, as shown in the supplemental material.

[4]The results of using Multi-similarity [41] and Angular [40] loss functions are in the supplementary material. Multi-similarity loss improves performance on the first task, but obtain similar results for longer sequences.

[5]Results on Cars-196 are shown in the supplemental material.

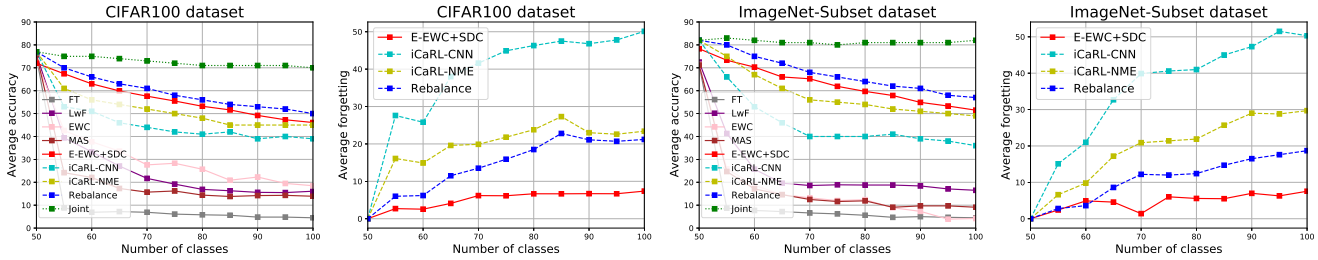

Figure 7: Comparison of average incremental accuracy and average forgetting with eleven-task setting on CIFAR100 and ImageNet-Subset dataset. Solid lines present **non-exemplar** based methods, dash lines present **exemplar** based methods.

5.2. Comparison to State-of-the-Art Methods

Ten-task IL on CUB-200 and Caltech-101 To evaluate SDC for longer sequences and compare to Learning without Memorization (LwM), we follow the setting from [10] and conducted experiments on CUB-200 (100 classes) and Caltech-101, where classes are divided randomly into ten equal tasks. Fig. 6 shows the comparison with FT(softmax), LwM [10], Expert Gate [2], the upper bound of joint training, and our best overall methods E-MAS and E-MAS+SDC. We obtain a clear superiority on both datasets. Interestingly, E-MAS already obtains 21.2% and 29.0% higher than the recent LwM method respectively on these two datasets after training 10 tasks. Applying our SDC method further improves the gain further with 6.4% on CUB-200-2011 and 1.4% on Caltech-101.

Experiments on CIFAR100 and ImageNet-Subset In [15], the eleven-task evaluation protocol for class-incremental learning was used, where the first task consists of half of the available classes and the rest is split in 10 tasks equally. *Average forgetting* is defined in [6] to estimate the forgetting of previous tasks. They quantify forgetting for the $j-$th task $f_j^k = \max_{l\in 1,\dots,k-1}(a_{l,j} - a_{k,j}), \forall j < k$, where $a_n m$ is the accuracy of task n after training task m. The average forgetting at $k-$th task is written as $F_k = \frac{1}{k-1}\sum_{j=1}^{k-1} f_j^k$.

For the CIFAR100 results of average incremental accuracy and average forgetting are presented in Fig.7. Three groups of methods are shown: non-exemplar based (FT, LwF, EWC, MAS, E-MAS+SDC), exemplar based (iCaRL-CNN [30], iCaRL-NME [30], Rebalance [15]), and joint training. From the average incremental accuracy, we can see that our overall best method E-EWC+SDC beats all the other non-exemplar based methods by a large margin, with a minimal gap of 27.6% compared to EWC. It also surpasses two exemplar based methods, namely iCaRL-CNN and iCaRL-NME [30], by 7.1% and 1.1%. To compare the preventing forgetting capability, we show the performance of our method and exemplar based methods in terms of *average forgetting* metric in Fig.7. Our method (in red) suffers from less forgetting than all the exemplar based methods, obtaining a 13.9% gain over the best exemplar method (Rebalance [15]). Experiments on ImageNet-Subset outperform all the non-exemplar based methods and two exemplar based methods as well in Fig.7. The conclusion is consistent with CIFAR100 on average incremental accuracy, 35.0% higher than LwF and 15.5% higher than iCaRL-CNN and 2.5% higher than iCaRL-NME. For average forgetting, our method has 3.5% less forgetting than Rebalance method.

Finally, we also ran fixing the network after finetuning on task one for both datasets. The results are 46.3% for $CIFAR100$ and 50.5% on $ImageNet - Sub$. This shows that currently for these difficult multitask settings, methods without exemplars do not significantly outperform this baseline; even some methods with exemplars such as iCaRL-CNN and iCaRL-NME fail on that. This is partially due to the large number of classes in task 1, if we would focus on the performance on the continually learned tasks (2-end) these methods would still report a clear advantage (also see supplementary materials).

6. Conclusions

The dramatic effect of forgetting when applying finetuning, as observed on classification networks, is much less pronounced for embedding networks. This suggest that the current dominance of softmax based methods in continual learning needs to be revisited and our results advocate the usage of embedding networks instead. Furthermore, we proposed a method to approximate the semantic drift of prototypes during training of new tasks. The method is complementary to several existing methods for incremental learning originally designed for classification networks. Experiments show that our method consistently improves results when combined with existing approaches.

Acknowledgement We acknowledge the support from Huawei Kirin Solution, the Industrial Doctorate Grant 2016 DI 039 of the Generalitat de Catalunya, the EU Project CybSpeed MSCA-RISE-2017-777720, EU's Horizon 2020 programme under the Marie Sklodowska-Curie grant agreement No.6655919 and the Spanish project RTI2018-102285-A-I00, National Key Laboratory on Blind Signal Processing under grant No.61424131903.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2, 5, 6

[2] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pages 3366–3375, 2017. 2, 4, 6, 8

[3] E. Belouadah and A. Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 583–592, 2019. 2

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, pages 737–744, 1994. 3

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11216 LNCS:241–257, 2018. 2, 4

[6] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547, 2018. 2, 4, 6, 8

[7] W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *CVPR*, pages 403–412, 2017. 3

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005. 2, 3

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 6

[10] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. In *CVPR*, pages 5138–5146, 2019. 2, 8

[11] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR workshop*, pages 178–178. IEEE, 2004. 6

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 7

[13] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 3, 4, 7

[14] S. Horiguchi, D. Ikami, and K. Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *IEEE Trans. on PAMI*, 2019. 4

[15] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2, 4, 7, 8

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2014. 7

[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. Nat. Acad. Sci. USA*, page 201611835, 2017. 1, 2, 4, 5, 6

[18] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[19] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Trans. on PAMI*, 40(12):2935–2947, 2018. 1, 2, 4, 6

[20] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *Proc. ICPR*, 2018. 2, 4, 6

[21] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NIPS*, pages 6467–6476, 2017. 2

[22] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, pages 67–82, 2018. 2

[23] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, pages 7765–7773, 2018. 2

[24] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez. Metric learning for novelty and anomaly detection. In *BMVC*, 2018. 3

[25] M. Masana, T. Tuytelaars, and J. van de Weijer. Ternary feature masks: continual learning without any forgetting. *arXiv preprint arXiv:2001.08714*, 2020. 2

[26] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[27] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. on PAMI*, 35(11):2624–2637, 2013. 2

[28] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics and Image Processing*, pages 722–729. IEEE, 2008. 6

[29] J. Rajasegaran, M. Hayat, S. H. Khan, F. S. Khan, and L. Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems*, pages 12648–12658, 2019. 3

[30] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542. IEEE, 2017. 1, 2, 4, 6, 8

[31] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. In *ICLR*, 2016. 3

[32] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 2

[34] T. Scott, K. Ridgeway, and M. C. Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. In *NIPS*, pages 76–85, 2018. 3

[35] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4555–4564, 2018. 2

[36] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NIPS*, pages 2990–2999, 2017. 2, 4

[37] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017. 3

[38] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 2

[39] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, pages 1386–1393, 2014. 3

[40] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *ICCV*, pages 2612–2620. IEEE, 2017. 2, 3, 7

[41] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *CVPR*, pages 5022–5030, 2019. 3, 7

[42] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 6

[43] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu. Memory replay gans: learning to generate images from new categories without forgetting. In *NIPS*, 2018. 2, 4

[44] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *CVPR*, 2019. 2, 4

[45] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu. Robust classification with convolutional prototype learning. In *CVPR*, pages 3474–3482, 2018. 3

[46] L. Yu, V. O. Yazici, X. Liu, J. Van de Weijer, Y. Cheng, and A. Ramisa. Learning metrics from teachers: Compact networks for image embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2907–2916, 2019. 5

[47] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995. JMLR. org, 2017. 2