FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

Grégoire Petit[1,2], Adrian Popescu[1], Hugo Schindler[1], David Picard[2], Bertrand Delezoide[3]
[1]Université Paris-Saclay, CEA, LIST, F-91120, Palaiseau, France
[2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
[3]Amanda, 34 Avenue Des Champs Elysées, F-75008, Paris, France
{gregoire.petit, adrian.popescu}@cea.fr,hugo-schindler@orange.fr
david.picard@enpc.fr,bertrand.delezoide@amanda.com

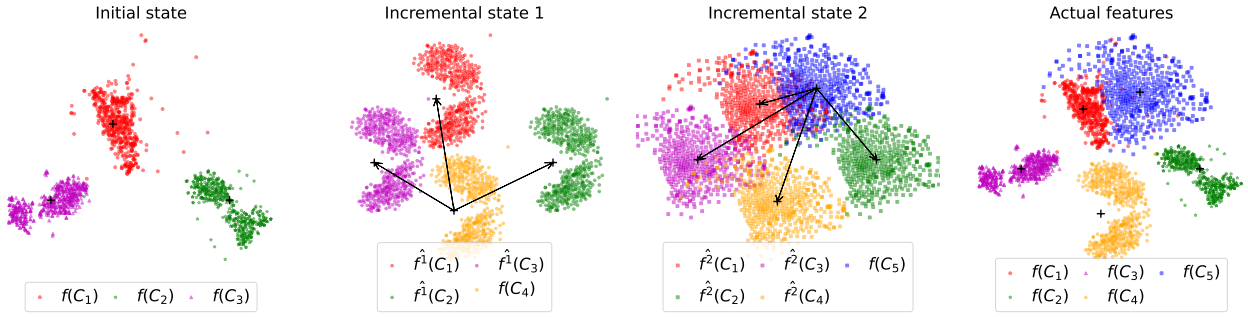
Abstract

Exemplar-free class-incremental learning is very challenging due to the negative effect of catastrophic forgetting. A balance between stability and plasticity of the incremental process is needed in order to obtain good accuracy for past as well as new classes. Existing exemplar-free class-incremental methods focus either on successive fine tuning of the model, thus favoring plasticity, or on using a feature extractor fixed after the initial incremental state, thus favoring stability. We introduce a method which combines a fixed feature extractor and a pseudo-features generator to improve the stability-plasticity balance. The generator uses a simple yet effective geometric translation of new class features to create representations of past classes, made of pseudo-features. The translation of features only requires the storage of the centroid representations of past classes to produce their pseudo-features. Actual features of new classes and pseudo-features of past classes are fed into a linear classifier which is trained incrementally to discriminate between all classes. The incremental process is much faster with the proposed method compared to mainstream ones which update the entire deep model. Experiments are performed with three challenging datasets, and different incremental settings. A comparison with ten existing methods shows that our method outperforms the others in most cases. FeTrIL code is available at `https://github.com/GregoirePetit/FeTrIL`.

1. Introduction

Deep learning [8] has dramatically improved the quality of automatic visual recognition, both in terms of accuracy and scale. Current models discriminate between thousands of classes with an accuracy often close to that of human recognition, assuming that sufficient training examples are provided. Unlike humans, algorithms reach optimal performance only if trained with all data at once whenever new classes are learned. This is an important limitation because data often occur in sequences [17] and their storage is costly. Also, iterative retraining to integrate new data is computationally costly and difficult in time- or computation-constrained applications [9, 32]. Incremental learning [36] was introduced to reduce the memory and computational costs of machine learning algorithms. The main problem faced by class-incremental learning (CIL) methods is catastrophic forgetting [14, 25], the tendency of neural nets to underfit past classes when ingesting new data. Many recent solutions [4, 13, 33, 44, 46], based on deep nets, use replay from a bounded memory of the past to reduce forgetting. However, replay-based methods make a strong assumption because past data are often unavailable [41]. Also, the footprint of the image memory can be problematic for memory-constrained devices [32]. Exemplar-free class-incremental learning (EFCIL) methods recently gained momentum [45, 38, 47, 48]. Most of them use distillation [12] to preserve past knowledge, and generally favor plasticity. New classes are well predicted since models are learned with all new data and only a representation of past data [24, 31, 49]. A few EFCIL methods [1, 6] are inspired by transfer learning [37, 39]. They learn a feature extractor in the initial state, and use it as such later to train new classifiers. In this case, stability is favored over plasticity since the model is frozen [24].

We introduce FeTrIL, a new EFCIL method which combines a frozen feature extractor and a pseudo-feature generator to improve incremental performance. New classes are represented by their image features obtained from the feature extractor. Past classes are represented by pseudo-features which are derived from features of new classes by using a geometric translation process. This translation moves features toward a region of the features space which is relevant for past classes. The proposed pseudo-feature generation is adapted for EFCIL since it is simple, fast and only requires the storage of the centroids for past classes.



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

Figure 1. Illustration of the proposed pseudo-feature generation procedure. This toy example includes an initial state (3 classes) and two IL states (1 new class per state) in subfigures (a), (b) and (c). Subfigure (d) provides the actual features of all classes that would be available for a classical learning. The illustration uses a 2D projection of actual features. Pseudo-features of past classes are generated by geometric translation of features of the new class added in each state with the difference between the centroids of the target past class and of the new class. While imperfect, the pseudo-feature generator produces a usable representation of past classes. *Best viewed in color.*

FeTrIL is illustrated with a toy example in Figure 1. We run experiments with a standard EFCIL setting [13, 47, 48], which consists of a larger initial state, followed by smaller states which include the same number of classes. Results show that the proposed approach has better behavior compared to ten existing methods, including very recent ones.

2. Related Work

CIL algorithms are needed when data arrive sequentially and/or computational constraints are important [9, 17, 24, 29]. Their objective is to ensure a good balance between plasticity, i.e. integration of new information, and stability, i.e. preservation of knowledge about past classes [27]. This is challenging because the lack of past data leads to catastrophic forgetting, i.e. the tendency of neural networks to focus on newly learned data at the expense of past knowledge [25]. Recent reviews of CIL [2, 24] show that a majority of methods replay samples of past classes to mitigate forgetting [4, 13, 33, 46]. One advantage here is that the network architecture remains constant throughout the incremental process. However, these methods have two major drawbacks: (1) First, the assumption that past samples are available is strong since in many cases past data cannot be stored due, for instance, to privacy restrictions [41] and (2) the memory footprint of the stored images is high.

Here, we investigate exemplar-free CIL, with focus on methods which keep the network size constant. This setting is very challenging since it imposes strong constraints on both memory and computational costs. A majority of existing methods use regularization to update the deep model for each incremental step [24], and adapt distillation [12] to preserve past knowledge by penalizing variations for past classes during model updates. Note that, while some of the distillation-based methods were introduced in an exemplar-based CIL (EBCIL) setting, many of them are also applicable to EFCIL. This approach to CIL was popularized by iCaRL [33], itself inspired by learning without forgetting (LwF) [19]. Distillation was later refined and complemented with other components to improve the plasticity-stability compromise. LUCIR [13] applies distillation on features instead of raw classification scores to preserve the geometry of past classes, and an inter-class separation to maximize the distances between past and new classes. The problem was partially addressed by adding specific class separability components in [7, 13]. Distillation-based methods need to store the current and the preceding model for incremental updates. Their memory footprint is larger compared to methods which do not use distillation [24].

Another important problem in CIL is the semantic drift between incremental states. Auxiliary classifiers were introduced in [20] to reduce the effect of forgetting. ABD [38] uses image inversion to produce pseudo-samples of past classes. The method is interesting but image inversion is difficult for complex datasets. Another interesting solution is proposed in [45], where the features drift between incremental steps is estimated from that of new classes. Recent EFCIL approaches [47, 48, 49] use past class prototypes in conjunction with distillation to improve performance. Prototype augmentation is proposed in PASS [48] to improve the discrimination of classes learned in different incremental states. Feature generation for past classes is introduced in IL2A [47] by leveraging information about the class distribution. This approach is difficult to scale-up because a covariance matrix needs to be stored for each class. A prototype selection mechanism is introduced in SSRE [49] to better discriminate past from new classes. FeTrIL shares the idea of using class prototypes with [45, 47, 48, 49]. An important difference is that we freeze the model after the initial state, while the other methods deploy more sophisticated mechanisms to integrate prototypes in a knowledge distillation process. Past comparative studies [2, 24] found that, while appealing in theory, distillation-based methods un-

derperform in EFCIL, particularly for large-scale datasets. Second, since the representation space is fixed, a simple geometric translation of actual features of new classes is sufficient to produce usable pseudo-features. In contrast, IL2A [47], the work which is closest to ours, needs to store a covariance matrix per class to obtain optimal performance. Third, the use of a fixed extractor simplifies the training process since only the final linear layer is trained, compared to a fine tuning of the backbone model required by recent methods which use prototypes and feature generation.

Another line of work takes inspiration from transfer learning [28, 37] to tackle EFCIL. A feature extractor is trained in the initial non-incremental state and fixed afterwards. Then, an external classification layer is updated in each incremental state to integrate new classes. The nearest class mean (NCM) [26] was used in [33], linear SVMs [30] were used in [1] and extreme value machines [34] were recently tested by [6]. The advantages of transfer-learning methods are their simplicity, since only the classification layer is updated, and their lower memory requirement, since they need a single deep model to function. These methods give competitive performance compared to distillation-based ones in EFCIL, particularly at scale [2]. However, features are not updated, and they are sensitive to large domain shifts between incremental tasks [17]. Equally, existing transfer-learning inspired works do not sufficiently address inter-class separability, which is in focus here.

Class prototypes creation was studied in other learning settings than CIL. A very interesting method focused on few-shot learning was proposed in [5]. A distance-based classifier which uses an approximation of the Mahalanobis distance is proposed. The means and variances of new classes are predicted using two supplementary neural networks. While adapted for few-shot learning, such an approach is not fully adapted in CIL. First, the supplementary neural networks require a large number of supplementary parameters. This is a disadvantage here, since CIL methods are needed in computationally-constrained environments. Second, we do not focus on few-shot learning and the means of past classes are well-placed in the representation space.

3. Proposed Method

The objective of CIL is to learn a total of N classes which appear sequentially during training. This process includes an initial state (0) and T incremental ones. New classes need to be recognized alongside past classes which were learned in previous states. We focus on the exemplar-free CIL setting [33, 38, 45, 49], which assumes that no past images can be stored. This scenario is more challenging than exemplar-based CIL since catastrophic forgetting needs to be tackled without resorting to replay [24]. There is no intersection between the classes learned in different incremental states. Unlike task IL [40], the boundaries between different states are not known at test time.

The global functioning of FeTrIL is illustrated in Figure 2. It uses a feature extractor, a pseudo-feature generator based on geometric translation, and an external classification layer in order to address EFCIL. Inspired by transfer-learning based CIL [1, 33], the feature extractor $\mathcal{F}$ is frozen after the initial state. This ensures a stable representation space through the entire CIL process. Given that images of past classes cannot be stored in EFCIL, a generator $\mathcal{G}$ is used to produce pseudo-features of past classes ($\hat{f}^t(C_p)$). $\mathcal{G}$ takes features of new classes ($f(C_n)$) and prototypes of past and new classes ($\mu(C_p)$, $\mu(C_n)$) as inputs. A linear classifier L combines features and pseudo-features to jointly train classifiers for all seen classes (past and new). The pseudo-features generation is crucial since it enables class discrimination across all incremental states. The hypotheses made here are that: (1) while imperfect, the pseudo-features still produce effective representations of past classes, and (2) using a frozen extractor in combination with a generator in EFCIL is preferable to mainstream distillation-based methods [45, 47, 48, 49]. These hypotheses are tested through the extensive experiments in Section 4. We present the main components of FeTrIL in the next subsections.

3.1. Generation of pseudo-features

The pseudo-feature generator, illustrated in Figure 1, produces effective representations of past classes. Existing approaches which generate past data rely on methods such as generative adversarial networks [10], image inversion [38], or covariance-based past class models [47]. We propose a much simpler alternative which is defined as:

$$\hat{f}^t(c_p) = f(c_n) + \mu(C_p) - \mu(C_n) \quad (1)$$

with: C_p - target past class for which pseudo-features are needed; C_n - new class for which images b are available; $f(c_n)$ - features of a sample c_n of class C_n extracted with $\mathcal{F}$; $\mu(C_p), \mu(C_n)$ - mean features of classes C_p and C_n extracted with $\mathcal{F}$; $\hat{f}^t(c_p)$ - pseudo-feature vector of a pseudo-sample c_p of class C_p produced in the t^{th} incremental state.

Eq. 1 translates the value of each dimension with the difference between the values of the corresponding dimension of $\mu(C_p)$ and $\mu(C_n)$. It creates a pseudo-feature vector situated in the region of the representation space associated to target class C_p based on actual features of a new class $f(C_n)$. The computational cost of generation is very small since it only involves additions and subtractions. $\mu(C_p)$ is needed to drive the geometric translation toward a region of the representation space which is relevant for C_p. Centroids are computed when classes occur for the first time and then stored. Their reuse is possible because $\mathcal{F}$ is fixed after the initial step and its associated features do not evolve.

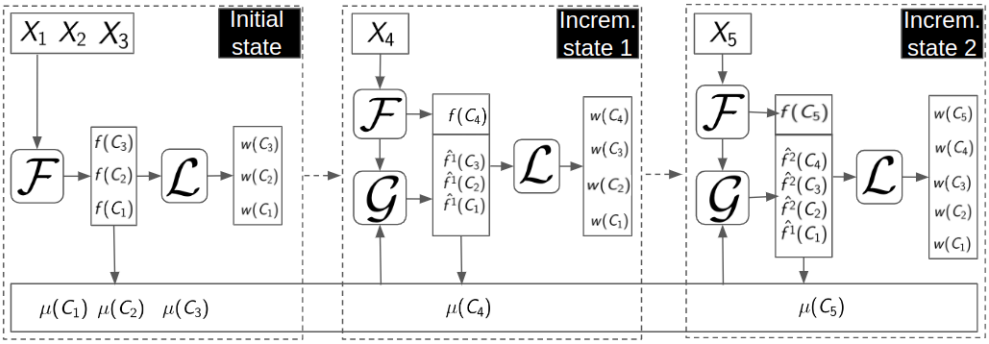


Figure 2. FeTrIL overview for a toy example with an initial state (3 classes) and two incremental states (1 class per state). The feature extractor $\mathcal{F}$ is trained in the initial state, using sets of data X_1, X_2, X_3, and then frozen afterwards. The generator $\mathcal{G}$ uses features $f(C_n)$ of the new class extracted with $\mathcal{F}$ and prototypes of past classes $\mu(C_p)$ to generate pseudo-features of past classes $\hat{f}^t(C_p)$ in the t^{th} state. Prototypes ($\mu(C_i)$) are the centroids of all classes (past and new). They are learned when classes are first seen and then stored throughout the IL process. A linear classifier $\mathcal{L}$ is used to learn classification weights $w(C_i)$ for all seen classes (past and new).

3.2. Selection of pseudo-features

Eq. 1 translates the features for a single sample. If each class is represented by s samples, the generation process needs to be repeated s times. The overview of FeTrIL (Figure 2) and of the pseudo-feature generation (Figure 1) use a minimal example which adds a single class per IL state. When CIL states include several classes C_n, the s pseudo-features of each class C_p can be obtained using different strategies, depending on how features of new classes are used. We deploy the following strategies:

- FeTrIL^k: s features are transferred from the k^{th} similar new class of each past class C_p. Similarities between the target C_p and the C_n available in the current state is computed using the cosine similarity between the centroids of each pair of classes. Experiments are run with different values of k to assess if a variable class similarity has a significant effect on EFCIL performance. Since translation is based on a single new class, the distribution of pseudo-features will be similar to that of features of C_n, but in the region of the representation space around $\mu(C_p)$.
- FeTrIL^{rand}: s features are randomly selected from all new classes. This strategy assesses whether a more diversified source of features from different C_n produces an effective representation of class C_p.
- FeTrIL^{herd}: s features are selected from any new class based on a herding algorithm [43]. It assumes that sampling should include features which produce a good approximation of the past class. Herding was introduced in exemplar-based CIL in order to obtain an accurate approximation of each class by using only a few samples [33] and its usefulness was later confirmed [2, 13, 44]. It is adapted here to obtain a good approximation of the sample distribution of C_p with s pseudo-features.

The comparison of these different strategies will allow us to determine whether the geometric translation of features is prevalent, or if a particular configuration of the features around the centroid of the target past class is needed.

3.3. Linear classification layer training

We assume that the CIL process is in the t^{th} CIL state, which includes P past classes and N new classes. The combination of the feature generator (Subsection 3.1) and selection (Subsection 3.2) provides a set $\hat{f}^t(C_p)$ of s pseudo-features for each class C_p. The objective is to train a linear classifier for all $P + N$ seen classes which takes pseudo features of past classes and actual features of new classes as inputs. This linear layer is defined as:

$$\mathcal{W}^t = \{w^t(C_1), ..., w^t(C_P), w^t(C_{P+1}), ..., w^t(C_{P+N})\} \quad (2)$$

with: w^t - the weight of known classes in the t^{th} CIL state.

$\mathcal{W}^t$ can be implemented using different classifiers, and we instantiate two versions in Section 4: (1) FeTrIL using LinearSVCs [30] as external classifiers, and (2) FeTrIL$_{fc}$ using a fully-connected layer to enable end-to-end training.

4. Evaluation

We evaluate FeTrIL by using a comprehensive EFCIL evaluation scenario [47, 48, 49]. This setting includes four datasets and CIL states of different size.

Datasets. We use four public datasets: (1) CIFAR-100 [16] - 100 classes, 32x32 pixels images, 500 and 100 images/class for training and test; (2) TinyImageNet [18] - 200 leaf clases from ImageNet, 64x64 pixels images, 500 and 50 for training and test; (3) ImageNet-Subset - 100 classes subset of ImageNet LSVRC dataset [35], 1300 and 50 for training and test; (4) ILSVRC - full dataset from [35].

Incremental setting. We use a classical EFCIL protocol from [47, 48, 49]. The number of classes in the initial state is larger, and the rest of the classes are evenly distributed between incremental states. CIFAR-100 and ImageNet-Subset are tested with: (1) 50 initial classes and 5 IL states of 10 classes, (2) 50 initial classes and 10 IL states of 5

classes, (3) 40 initial classes and 20 states of 3 classes, and (4) 40 initial classes and 60 states of 1 class. Compared to [47, 48, 49], configurations (1) and (3) for ImageNet-Subset are added for more consistent evaluation. TinyImageNet is tested with 100 initial classes and the other classes distributed as follows: (1) 5 states of 20 classes, (2) 10 states of 10 classes, (3) 20 states of 5 classes, and (4) 100 states of 1 class. Configuration (4) is interesting since it enables one class increments. It cannot be deployed for any of the compared EFCIL methods since they require at least two classes per increment to update models. ILSVRC is tested with 500 initial classes, and the other 500 split evenly among $T \in \{5, 10, 20\}$ states. This enables a comprehensive comparison of the methods in varied EFCIL configurations. Naturally, task IDs are not available at test time.

Compared methods. We use the following EFCIL methods in evaluation: EWC [15], LwF-MC [33], DeeSIL [1], LUCIR [13], MUC [20], SDC [45], PASS [48], ABD [38], IL2A [47], SSRE [49]. As we discussed in Section 2, these methods cover a large variety of EFCIL approaches. The inclusion of recent works [47, 48, 49] is important to situate our contribution with respect to current EFCIL trends. While focus is on EFCIL, we follow [49] and include a comparisonwith EBCIL methods. We test our method against the recent AANets approach [21], and against the EBCIL methods to which AANETS was added (LUCIR [13], Mnemonics [22], PODNet [7]). Whenever available, results of compared methods marked with * are reproduced either from their initial paper or from [49] for EFCIL or from [21] for EBCIL. The other results are recomputed using the original configurations of the methods.

Implementation details. Following [33, 47, 48, 49], we use ResNet-18 [11] in all experiments. FeTrIL initial training is done uniquely with images of initial classes to ensure comparability with existing methods. The feature extractor is trained in the initial state and then frozen for the reminder of the IL process. We implement a supervised training with cross-entropy loss, SGD optimization, a batch size of 128, for a total of 160 epochs. The initial learning rate is 0.1, and it is decayed by 0.1 after every 50 epochs. To ensure comparability, classes are assigned to IL states using the same random seed as in the compared methods [13, 48, 47, 49].

We provide implementation details for the final layer (Eq. 2) introduced in Subsection 3.3. The hyperparameters of the classification layers were optimized on a pool of 50 classes selected randomly from ImageNet, but disjoint from ILSVRC or ImageNet-Subset. L2-normalization is applied before the linear layer. The LinearSVC layer included in FeTrIL^1 uses 1.0 and 0.0001 for regularization and the tolerance parameters. The number of samples is higher than the dimensionality of the features, and we solve the primal rather than the dual optimization problem. The classifiers are then trained using a standard one against the rest procedure. In Subsection 4.2, we also test a one-vs-many strategy to accelerate incremental updates. The second variant, FeTrIL$^1_{fc}$, using a fully-connected layer as final layer, and implements an end-to-end training strategy. FeTrIL$^1_{fc}$is trained for 50 epochs with an initial learning rate of 0.1, 0.1 decay, and 10 epochs patience.

Evaluation metric. The average incremental accuracy, widely used in CIL [24, 33], is the main evaluation measure. For comparability with [47, 48, 49], it is computed as the average accuracy of all states, including the initial one. We equally provide per-state accuracy curves to have a more detailed view of the accuracy evolution during the CIL process. Following [49], we run each configuration of FeTrIL three times and report the averaged results.

4.1. Results

Comparison to existing EFCIL methods. The results from Table 1 show that FeTrIL^1 outperforms all compared methods in 11 tested configurations out of 12. It is also close to the best in the remaining one. The second best results are obtained with the very recent SSRE method [49]. FeTrIL^1 and SSRE accuracies are close to each other for CIFAR-100, with relative differences between 0.4 and -0.2. The performance gain brought by FeTrIL is of over 4 and 3 top-1 accuracy points for TinyImageNet and ImageNet-Subset, respectively. PASS [48] and IL2A [47], two other recent EFCIL methods, have lower average performance. We note that EFCIL performance boost was recently reported, with methods such as PASS, IL2A, SSRE. These methods combine knowledge distillation and sophisticated mechanisms for dealing with the stability-plasticity dilemma. In contrast, our method uses a fixed feature extractor and a lightweight pseudo-feature generator. FeTrIL only optimizes a linear classification layer, while compared recent methods use backpropagation of the entire model, and need much more computational resources and time to perform the IL process. A more in-depth discussion of complexity is proposed in Subsection 4.2. Performance of the ILSVRC dataset is also very interesting. Direct comparison to PASS or SSRE is impossible since these methods were not tested at scale. However, we can safely assume that FeTrIL^1 is better given PASS and SSRE accuracy for the simpler ImageNet-Subset. ILSVRC results show that the simple method proposed here is effective for a high range of classes. Interestingly, ILSVRC performance is stabler compared to smaller datasets since the pool of new classes available for pseudo-features generation is larger.

Comparison to a transfer-learning baseline. DeeSIL [1] is a simple application of transfer learning to EFCIL. It has no class separability mechanism across different incremental states since classifiers are learned within each state. The need for global separability, included in FeTrIL, is shown by the comparison of short and long

CIL Method	CIFAR-100				TinyImageNet				ImageNet-Subset				ImageNet		
	T=5	T=10	T=20	T=60	T=5	T=10	T=20	T=100	T=5	T=10	T=20	T=60	T=5	T=10	T=20
EWC* [15] (PNAS'17)	24.5	21.2	15.9	x	18.8	15.8	12.4	x	-	20.4	-	x	-	-	-
LwF-MC* [33] (CVPR'17)	45.9	27.4	20.1	x	29.1	23.1	17.4	x	-	31.2	-	x	-	-	-
DeeSIL [1] (ECCVW'18)	60.0	50.6	38.1	x	49.8	43.9	34.1	x	67.9	60.1	50.5	x	61.9	54.6	45.8
LUCIR (CVPR'19)	51.2	41.1	25.2	x	41.7	28.1	18.9	x	56.8	41.4	28.5	x	47.4	37.2	26.6
MUC* [20] (ECCV'20)	49.4	30.2	21.3	x	32.6	26.6	21.9	x	-	35.1	-	x	-	-	-
SDC* [45] (CVPR'20)	56.8	57.0	58.9	x	-	-	-	x	-	61.2	-	x	-	-	-
ABD* [38] (ICCV'21)	63.8	62.5	57.4	x	-	-	-	x	-	-	-	x	-	-	-
PASS* [48] (CVPR'21)	63.5	61.8	58.1	x	49.6	47.3	42.1	x	64.4	61.8	51.3	x	-	-	-
IL2A* [47] (NeurIPS'21)	66.0	60.3	57.9	x	47.3	44.7	40.0	x	-	-	-	x	-	-	-
SSRE* [49] (CVPR'22)	65.9	65.0	**61.7**	x	50.4	48.9	48.2	x	-	67.7	-	x	-	-	-
FeTrIL$^1_{fc}$	64.7	63.4	57.4	50.8	52.9	51.7	49.7	41.9	69.6	68.9	62.5	58.9	65.6	64.4	63.4
FeTrIL^1	**66.3**	**65.2**	61.5	**59.8**	**54.8**	**53.1**	**52.2**	**50.2**	**72.2**	**71.2**	**67.1**	**65.4**	**66.1**	**65.0**	**63.8**

Table 1. Average top-1 incremental accuracy in EFCIL with different numbers of incremental steps. FeTrIL^1 results are reported with pseudo-features translated from the most similar new class. "-" cells indicate that results were not available (see supp. material for details). "x" cells indicate that the configuration is impossible for that method. **Best results - in bold**, second best - underlined.

CIL processes. DeeSIL [1] performance is good for $T = 5$ because each class is trained against enough other classes, but drops significantly for $T = 20$, when there are few new classes. The important performance gain brought by FeTrIL highlights the importance of class separability.

Behavior for minimal incremental updates. Compared EFCIL methods can only be updated with a minimum of two classes per CIL state since they use discriminative classifiers, which require both positive and negative samples. In practice, it is interesting to enable updates once each new class is available. This is possible with FeTrIL because pseudo-features can all originate from a single new class. Results in the right columns of CIFAR-100, TinyImageNet and ImageNet-Subset from Table 1 show that the accuracy obtained in with one class increments is close to that observed for $T = 20$. This highlights the robustness of FeTrIL with respect to frequent updates.

Influence of the final classification layer. FeTrIL^1 compares favorably with FeTrIL$^1_{fc}$. LinearSVC gives better performance than a fully-connected layer, particularly for a large number of incremental steps. However, FeTrIL$^1_{fc}$ is also competitive, and outperforms existing methods in a majority of configurations.

Detailed view of accuracy. We illustrate the evolution of accuracy across incremental states in Figure 3 to complement the averaged results from Table 1. These detailed results confirm the good behavior of the proposed method. The evolution of accuracy for FeTrIL and SSRE is very similar for CIFAR-100, FeTrIL method is better throughout the process for TinyImageNet, and also better than SSRE for the first incremental states for ImageNet-Subset. The performance gain with respect to the other compared methods is much larger for all incremental states.

Comparison to exemplar-based CIL methods. This comparison is interesting because EFCIL is a much more challenging task than EBCIL [2, 24], and an important performance gap between the two was observed. This is intuitive since the storage of images of past classes in EBCIL mitigates catastrophic forgetting. Following [13, 21], a memory of 20 images per class is allowed for all EBCIL methods tested here. FeTrIL is better than all three base methods to which AANets is applied for CIFAR-100. For ImageNet-Subset, FeTrIL accuracy is better than LUCIR's, slightly behind that of Mnemonics [22] and approximately 3.5 points lower than that of PODNet [7]. The performance of FeTrIL remains close that of EBCIL methods in a majority of cases even after the introduction of AANets. The results from Table 2 indicate that, while still present, the gap between EFCIL and EBCIL methods is narrowing.

CIL Method	CIFAR-100		ImageNet-Subset	
	T = 5	T = 10	T = 5	T = 10
LUCIR [13] (CVPR'19)	63.2	61.1	70.8	68.3
+AAnets (CVPR'21)	66.7	65.3	72.6	69.2
Mnemonics [23] (CVPR'20)	63.3	62.3	72.6	71.4
+AAnets (CVPR'21)	67.6	65.7	72.9	71.9
PODNet [7] (ECCV'20)	64.8	63.2	75.5	74.3
+AAnets (CVPR'21)	66.3	64.3	77.0	75.6
FeTrIL^1	66.3	65.2	71.9	70.8

Table 2. Comparison of FeTrIL with the recent AANets method [21], applied on top of EBCIL baselines which store 20 exemplars of past classes to mitigate catastrophic forgetting.

4.2. Method analysis

We present an analysis of: (1) the selection strategies, (2) the memory footprint of the methods, (3) the complexity of model updates, and (4) the stability-plasticity balance.

Pseudo-feature selection comparison. FeTrIL can use any past-new classes combination for translation. In Table 3, we compare the selection strategies from Subsection 3.2. Accuracy varies in a relatively small range for all strategies, indicating that FeTrIL is robust to the way features of new classes are selected, and it can be successfully implemented with any of the strategies. FeTrIL^1 is better than the other selection methods and this motivates its use in the main experiments. Class similarity matters, but results with FeTrIL^{10} remain interesting. FeTrIL^{herd} also has interesting accuracy, but is slightly behind that of FeTrIL^1.

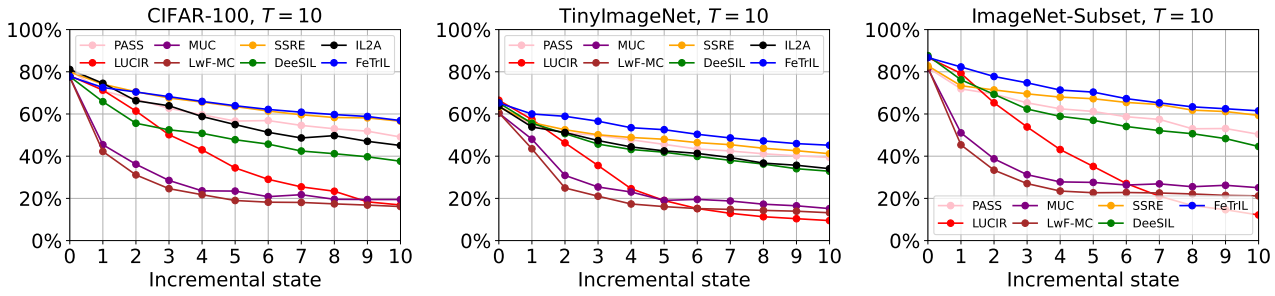


Figure 3. Evolution of top-1 accuracy for an incremental process with $T = 10$ IL states. *Best viewed in color.*

	CIFAR-100	TinyImageNet	ImageNet-Subset
	$T = 5$		
FeTrIL^{1}	66.3	54.8	72.2
FeTrIL^{5}	65.7	53.8	72.2
FeTrIL^{10}	65.1	53.8	71.6
FeTrIL^{herd}	66.2	53.8	72.1
FeTrIL^{rand}	65.1	51.5	70.3

Table 3. Average top-1 CIL accuracy obtained with the variants of pseudo-feature selection from Subsection 3.2 for $T = 5$. We set $k = \{1, 5, 10\}$ for the similarity rank between the past and new classes to test the effect of class similarities. There are 10 (CIFAR-100 and ImageNet-Subset) and 20 (TinyImageNet) new classes per state from which to select features translation.

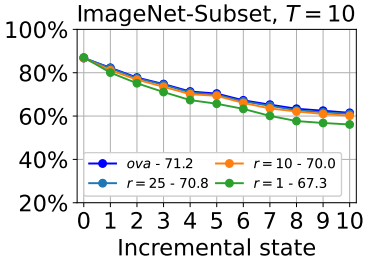


Figure 4. Top-1 incremental accuracy of FeTrIL^{1} for approximate training of the classification layer with different ratios for negative sampling. *ova* denotes a classical one-vs-all training procedure which is used to report the main results from Table 1 and Figure 3.

The results from Table 3 motivate the use of FeTrIL^{1} in the main experiments. Overall, the geometric translation toward the centroid of the past class is by far more important than the new classes features sampling policy. This finding is also supported by the results obtained with a single new class per CIL state (Table 1).

Memory footprint. A low memory footprint is a desirable property of incremental learning algorithms because they are most useful in memory-constrained applications [24, 32, 33], and recommended for embedded devices [9]. All EFCIL methods need to store a representation of past classes to counter catastrophic forgetting. Naturally, this representation should be as compact as possible. Mainstream methods (such as LwF-MC [19], PASS [48], IL2A [48], and SSRE [49]) need to the previous and current deep models during CIL updates for distillation. ResNet-18 [11], the most frequent CIL backbone, has approximately 11.4M parameters. Consequently, distillation-based methods require around 22.8M parameters. Transfer-based methods, such as DeeSIL [1] and FeTrIL, use only the deep model learned in the initial state and frozen afterwards, and only need 11.4M parameters for the model. DeeSIL does not need supplementary parameters during incremental updates. However, this comes at the cost of poor global discrimination of classes, which is reflected in the final performance. FeTrIL stores the class centroids of past classes in order to perform feature translation. Each class needs 512 parameters, which leads to a supplementary 51.2K and 102.4 memory need for 100 and 200 classes, respectively. The class similarities needed for pseudo-feature selection (Subsection 3.2) can be computed sequentially and the added memory cost of this step is negligible. PASS [48], IL2A [47] and SSRE [49] also require the storage of a prototype (mean representation) for each past class and their footprint is equivalent to that of FeTrIL. IL2A [47] additionally stores a covariance matrix per past class (512x512 for ResNet-18) for optimal functioning, which is prohibitive.

Complexity of incremental updates. CIL is useful in resource-constrained environments, and the integration of new classes should be fast [9, 32]. Distillation-based methods retrain the full backbone model at each update. This is is costly because backpropagation complexity depends on the network architecture, the number of samples and the number of epochs [8]. Updates of transfer-based methods are simpler because they update only the final layer. DeeSIL trains linear classifiers using a one-vs-all procedure within each CIL state. The complexity of one training epoch for all classifiers in a CIL state is $O((\frac{n}{T})^2 sd)$ [3], with n - total number of classes in the dataset, d - dimensionality of features and s - samples per class. FeTrIL retrains all linear classifiers, past and new, in each CIL state to improve global separability. Its complexity is $O(n^2 sd)$ in the last incremental state, which includes all classes. However, the one-versus-all training can be replaced with a one-versus-many training with negligible loss of accuracy. A sampling of negative features is performed to respect a predefined ratio r between negatives and positives used to train each

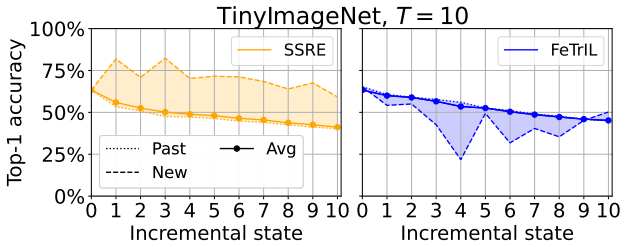


Figure 5. Top-1 incremental accuracy per state for past and new classes for TinyImageNet, with $T = 10$ incremental states for FeTrIL[1] and SSRE, the best compared method. An ideal method would provide high accuracy, but also similar performance for past and new classes. The accuracy of past and new classes is globally closer for FeTrIL[1] , which indicates that our method provides a better stability-plasticity balance than SSRE. Overall accuracy is better for FeTrIL[1] in Figure 3 because the contribution of new classes in each state diminishes during the CIL process.

classifier. This approximation has $O(rnsd)$ complexity. It is interesting since $r < n$, and is more and more useful as n grows during the IL process since r remains constant.

In Figure 4, we present results with different r values for ImageNet-Subset, $T = 10$. Accuracy drops when negative sampling is performed, but it is close to that of one-vs-all training when $r = 25$ and $r = 10$. Performance drops more significantly for $r = 1$, when each linear classifier is learned with an aggressive sampling of negatives. Similar results for CIFAR-100 and TinyImageNet are provided in the suppl. material. Globally, Figure 4 indicates that FeTrIL increments can be accelerated with little accuracy loss.

We measure the time needed for incremental training of ImageNet-Subset, $T = 10$. The training of the initial model is similar for all models and is thus discarded. FeTrIL training is done on a single thread of an Intel E5-2620v4 CPU, and only takes 1 hour, 4 minutes and 16 seconds. If FeTrIL is run with $r = 10$ ratio between positives and negatives, training time is only 15 minutes and 3 seconds. In comparison, PASS [48] needs 11 hours, 8 minutes and 19 seconds on an NVIDIA V100 GPU, with 4 workers for data loading. While clearly favorable to FeTrIL, the comparison is biased in favor of PASS since this method uses an entire GPU, in comparison to a single CPU thread for FeTrIL. Further speed gains are possible for our method by using a GPU implementation of the linear layer. Our method would run much faster with a GPU implementation of the linear layer. Note that the running time of the other methods, such as LUCIR [13] and SSRE [49], which perform backpropagation is similar to that of PASS [48].

Stability-plasticity balance. CIL should ideally ensure a similar accuracy level for past and new classes [24, 49]. Figure 5 shows that the two methods have complementary behavior, which results from the way deep backbones are used. SSRE is biased toward new classes since the model is fine tuned in each incremental state. FeTrIL favors past classes because the deep model is learned with the initial classes (a subset of past classes) and then frozen. The accuracy gap between past and new classes is smaller for FeTrIL compared to SSRE, except for state 4. There, low performance on new classes is probably explained by a strong domain shift compared to the initial state. Globally, the proposed method improves the stability-plasticity balance.

5. Conclusion

We introduce FeTrIL, a new method which addresses exemplar-free class-incremental learning. The proposed combination of a frozen feature extractor and of a pseudo-feature generator improves results compared to recent EF-CIL methods. The generation of pseudo-features is simple, since it consists in a geometric translation, yet effective. Our proposal is advantageous from memory and speed perspectives compared to mainstream methods [13, 33, 38, 42, 45, 47, 48, 49]. This is particularly important for edge devices [9, 32], whose storage and computation capacities are limited. FeTrIL performance is also close to that of exemplar-based methods, which need to store samples of past classes to mitigate catastrophic forgetting. While a gap between exemplar-based and exemplar-free setting subsists, it becomes significantly narrower. The results reported here resonate with past works which show that simple methods can be highly effective in CIL [2, 24, 31]. They question the usefulness of the knowledge distillation component, used by a majority of existing methods. The FeTrIL code will be made public to enable reproducibility.

The main limitations of the proposed method motivate our future work. First, FeTrIL uses a frozen feature extractor learned on the initial state and tends to favor past classes over new ones. We will investigate ways to combine the pseudo-feature generation mechanism and fine-tuning to further improve global performance, as well as the stability-plasticity balance. Second, FeTrIL produces usable pseudo-features, but past class representations would be better if the pseudo-features would be more similar to the original features of past classes. We will study methods that generate more refined features, for instance by using the distribution of the initial features. Last but not least, the tested selection strategies are all effective. However, they could be further improved by filtering out outliers based on the localization of pseudo-features in the representation space.

Acknowledgements. This work was supported by the European Commission under European Horizon 2020 Programme, grant number 951911 - AI4Media. It was made possible by the use of the FactoryIA supercomputer, financially supported by the Ile-de-France Regional Council.

References

[1] Eden Belouadah and Adrian Popescu. Deesil: Deep-shallow incremental learning. *TaskCV Workshop @ ECCV 2018.*, 2018.

[2] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.

[3] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20, 2007.

[4] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, pages 241–257, 2018.

[5] Debasmit Das and CS George Lee. A two-stage approach to few-shot learning for image recognition. *IEEE Transactions on Image Processing*, 29:3336–3350, 2019.

[6] Akshay Raj Dhamija, Touqeer Ahmad, Jonathan Schwan, Mohsen Jafarzadeh, Chunchun Li, and Terrance E Boult. Self-supervised features improve open-world learning. *arXiv preprint arXiv:2102.07848*, 2021.

[7] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer vision-ECCV 2020-16th European conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, volume 12365, pages 86–102. Springer, 2020.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. *arXiv preprint arXiv:2203.10681*, 2022.

[10] Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 98, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[12] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[13] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 831–839, 2019.

[14] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[15] 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. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[17] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *CoRR*, abs/1909.08383, 2019.

[18] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[19] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, ECCV, 2016.

[20] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *European Conference on Computer Vision*, pages 699–716. Springer, 2020.

[21] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2021.

[22] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 12242–12251. IEEE, 2020.

[23] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2020.

[24] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification, 2021.

[25] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[26] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

[27] M Mermillod, A Bugaiska, and P Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504–504, 2013.

[28] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[29] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113, 2019.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[31] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020.

[32] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. A tinyml platform for on-device continual learning with quantized latent replays. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):789–802, 2021.

[33] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[34] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[36] Jeffrey C Schlimmer and Douglas Fisher. A case study of incremental concept induction. In *AAAI*, volume 86, pages 496–501, 1986.

[37] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[38] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. *arXiv preprint arXiv:2106.09701*, 2021.

[39] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[40] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[41] Ragav Venkatesan, Hemanth Venkateswara, Sethuraman Panchanathan, and Baoxin Li. A strategy for an uncompromising incremental learner. *arXiv preprint arXiv:1705.00744*, 2017.

[42] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. *CoRR*, abs/2103.13558, 2021.

[43] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 1121–1128, 2009.

[44] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 374–382, 2019.

[45] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6980–6989. IEEE, 2020.

[46] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13205–13214. IEEE, 2020.

[47] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34, 2021.

[48] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021.

[49] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305, 2022.