Generative Feature Replay For Class-Incremental Learning

Xialei Liu[1,*], Chenshen Wu[1,*], Mikel Menta[1], Luis Herranz[1], Bogdan Raducanu[1],
Andrew D. Bagdanov[2], Shangling Jui[3], Joost van de Weijer[1]
[1] Computer Vision Center, Universitat Autonoma de Barcelona, Barcelona, Spain
[2] Media Integration and Communication Center, University of Florence, Florence, Italy
[3] Huawei Kirin Solution, Shanghai, China
{xialei,chenshen,mkmenta,lherranz,bogdan,joost}@cvc.uab.es,
andrew.bagdanov@unifi.it , jui.shangling@huawei.com

Abstract

Humans are capable of learning new tasks without forgetting previous ones, while neural networks fail due to catastrophic forgetting between new and previously-learned tasks. We consider a class-incremental setting which means that the task-ID is unknown at inference time. The imbalance between old and new classes typically results in a bias of the network towards the newest ones. This imbalance problem can either be addressed by storing exemplars from previous tasks, or by using image replay methods. However, the latter can only be applied to toy datasets since image generation for complex datasets is a hard problem.

We propose a solution to the imbalance problem based on generative feature replay which does not require any exemplars. To do this, we split the network into two parts: a feature extractor and a classifier. To prevent forgetting, we combine generative feature replay in the classifier with feature distillation in the feature extractor. Through feature generation, our method reduces the complexity of generative replay and prevents the imbalance problem. Our approach is computationally efficient and scalable to large datasets. Experiments confirm that our approach achieves state-of-the-art results on CIFAR-100 and ImageNet, while requiring only a fraction of the storage needed for exemplar-based continual learning. Code available at https://github.com/xialeiliu/GFR-IL.

1. Introduction

Humans and animals are capable of continually acquiring and updating knowledge throughout their lifetime. The ability to accommodate new knowledge while retaining previously learned knowledge is referred to as *incremental or continual learning*, which is essential to building scalable and reusable artificially intelligent systems. Current deep neural networks have achieved impressive performance on many benchmarks, comparable or even better than humans (e.g. image classification [13]). However, when trained for new tasks, these networks almost completely forget the previous ones due to the problem of *catastrophic forgetting* [31] between the new and previously-learned tasks.

To overcome *catastrophic forgetting* several approaches, inspired in part by biological systems, have been proposed. The first category of approaches use regularizers that limit the plasticity of the network while training

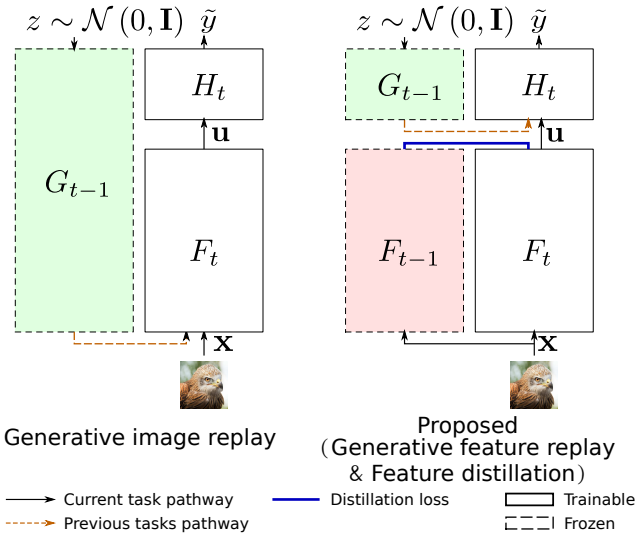

Figure 1. Comparison of generative image replay and the proposed generative feature replay. Instead of replaying images x the proposed method uses a generator G to replay features u. To prevent forgetting in the feature extractor F we apply feature distillation. Feature replay allows us to train classifiers H which do not suffer from the imbalance problem common to class-incremental methods. Furthermore, feature generation is significantly easier than image generation and can be applied to complex datasets.

*Both authors contributed equally.

on new tasks so the network remains stable on previous ones [1, 19, 23, 24, 56]. Another type of approach involves dynamically increasing the capacity of the network to accommodate new tasks [21, 44], often combined with task-dependent masks on the weights [28, 29] or activations [45] to reduce the chance of catastrophic forgetting.

A third category of approaches relies on memory replay, i.e. replaying samples of previous tasks while learning with the samples of the current task. These samples could be real ones ('exemplars'), like in [4, 25, 41] in which we refer to the process as 'rehearsal' or could be synthethic ones obtained through generative mechanisms, in which case we refer to the process as 'pseudo-rehearsal' [43, 46, 49]. Incremental learning methods are typically evaluated and designed for a particular testing scenario [48]. Task-incremental learning considers the case where the task ID is given at inference time [25, 29, 45]. Class-incremental learning considers the more difficult scenario in which the task ID is unknown at testing time [14, 41, 50].

Recently, research attention has shifted from task-incremental to class-incremental learning. The main additional challenge, which class-incremental methods have to address, is balancing the different classifier heads. The imbalance problem occurs because during training of the current task there is none or only limited data available from previous tasks, which biases the classifier towards the most recently learned task. Various solutions to this problem have been proposed. iCarL[41] stores a fixed budget of exemplars from previous tasks in a way that exemplars approximate the mean of classes in the feature space. The nearest-mean classifier is used for inference. Wu et al. [50] found that the last fully-connected layer has a strong bias towards new classes, and corrected the bias with a linear model estimated from exemplars. Hou et al. [14] replace the softmax with a cosine similarity-based loss, which, combined with exemplars, addresses the imbalance problem. All these methods have in common that they require storage of exemplars. However, for many applications – especially due to privacy concerns or storage restrictions – it is not possible to store *any* exemplars from previous tasks.

The only methods which successfully addresses the imbalance problem without requiring any exemplars are methods performing generative replay [46, 49]. These methods train a generator continuously to generate samples of previous tasks, and therefore prevent the imbalance problem. Thus, these methods report excellent results for class-incremental learning. However, they have one major drawback: the generator should accurately generate images from previous task distributions. For small data sets like MNIST and CIFAR-10 this is feasible, however, for larger datasets with more classes and larger images (like CIFAR-100 and ImageNet) these methods yield unsatisfactory results.

In this paper, we propose a novel approach based on generative feature replay to overcome catastrophic forgetting in class-incremental continual learning. Our approach is motivated by the fact that image generation is a complex process when the number of images is limited or the number of classes is high. Therefore, instead of image generation we adopt feature generation which is considerably easier than accurately generating images. We split networks into two parts: a feature extractor and a classifier. To prevent forgetting in the entire network, we combine generative feature replay (in the classifier) with feature distillation on the feature extractor. To summarize, our contributions are:

- We design a hybrid model for class-incremental learning which combines generative feature replay at the classifier level and distillation in the feature extractor.
- We provide visualization and analysis based on Canonical Correlation Analysis (CCA) of how and where networks forget in order to offer better insight.
- We outperform other methods which do not use exemplars by a large margin on the ImageNet and CIFAR-100 datasets. Notably, we also outperform methods using exemplars for most of the evaluated settings. Additionally, we show that our method is computationally efficient and scalable to large datasets.

2. Related Work

2.1. Continual learning

Continual learning can be divided into three main categories as follows (more details in the surveys [7, 36]):

Regularization-based methods. A first family of techniques is based on regularization. They estimate the relevance of each network parameter and penalize those parameters which show significant change when switching from one task to another. The difference between these methods lies on how the penalization is computed. For instance, the EWC approach in [19, 24], weights network parameters using an approximation of the diagonal of the Fisher Information Matrix (FIM). In [56], the importance weights are computed online. They keep track of how much the loss changes due to a change in a specific parameter and accumulate this information during training. A similar approach is followed in [1], but here, instead of considering the changes in the loss, they focus on the changes on activations. This way, parameter relevance is learned in an unsupervised manner. Instead of regularizing weights, [15, 23] align the predictions using the data from the current task.

Architecture-based methods. A second family of methods to prevent catastrophic forgetting produce modifications in a network's morphology by growing a sub-network for each task, either logically or physically [21, 44]. Piggyback [28] and Packnet [29] and learn a separate mask for each task,

while HAT [45] and Ternary Feature Masks [30] learn a mask on the activations instead of for each parameter.

Rehearsal-based methods. The third and last family of methods to prevent catastrophic forgetting are rehearsal-based. Existing approaches use two strategies: either store a small number of training samples from previous tasks or use a generative mechanism to sample synthetic data from previously learned distributions. In the first category, iCaRL [41] stores a subset of real data (called *exemplars*). For a given memory budget, the number of exemplars stored should decrease when the number of classes increases, which inevitably leads to a decrease in performance. A similar approach is pursued in [25], but the gradients of previous tasks are preserved. An improved version of this approach overcomes some efficiency issues [5]. In [14] the authors propose two losses called the less-forget constraint and inter-class separation to prevent forgetting. The less-forget constraint minimizes the cosine distance between the features extracted by the original and new models. The inter-class separation separates the old classes from the new ones with the stored exemplars used as anchors. In [50, 2], a bias correction layer to correct the output of the original fully-connected layer is introduced to address the data imbalance between the old and new categories. In [38], they propose to store activations for replay and a slow-down learning at all the layers below the replay layer.

Methods in the second category do not store any exemplars, but introduce a generative mechanism to sample data from. In [46], memory replay is implemented with an unconditional GAN, where an auxiliary classifier is required in order to determine which class the generated samples belong to. An improved version of this approach was introduced in [49], where they use a class-conditional GAN to generate synthetic data. In contrast, FearNet [17] uses a generative autoencoder for memory replay and [53] generates intermediate features. Using the class statistics from the encoder, synthetic data for previous tasks is generated based on the mean and covariance matrix. The main limitation of this approach is the assumption of a Gaussian distribution of the data and the reliance on pretrained models.

2.2. Generative adversarial networks

Generative adversarial networks (GANs) [11] are able to generate realistic and sharp images conditioned on object categories [12, 39], text [42, 57], another image (image translation) [18, 58] and style transfer [10]. In the context of continual learning, they were successfully been used for memory replay, by generating synthetic samples from previous tasks [49]. Here we are going to analyze the GANs limitations and argue why GANs for feature generation are preferable over image generation.

Adversarial image generation. Although GANs achieved impressive performance recently, in order to generate high-resolution images [3, 16], they are not immune to common GAN problems such as stability (solutions are available at a high computational costs) and the need for a large training set of real images. Additionally, the generation of high-resolution images does not guarantee that they are able to capture a large enough variety of visual concepts with a good discriminative power [6]. Only recently, the authors in [27] proposed to uses high resolution images.

However, they are not yet sufficient to generate high quality images for the downstream tasks, for instance training a deep neural network classifier. In the case of few-shot and zero-shot learning, only few samples or no sample are existing to train the GANs, which results in even more challenges to generate useful images.

Adversarial feature generation. Recently, feature generation has appeared as an alternative to image generation, especially for the cases of few-shot learning, demonstrating superior performance. In [51], they propose a GAN architecture with a classifier on top of the generator, in order to generate features that are better suited for classification. The same idea is further improved in [52], where they combine a better feature generator by combining the strength of a VAE and a GAN. In the current work, we use adversarial feature generation for memory replay in a continual learning framework. As demonstrated in [51, 52], feature generation has achieved superior performance compared to image generation for zero-shot and few-shot learning.

3. Forgetting in feature extractor and classifier

In this section, we take a closer look at how forgetting occurs at different levels in a CNN.

3.1. Class-incremental learning

Classification model and task. We consider classification tasks learned from a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{X}$ is the ith image, $y_i \in \mathcal{C}$ is the corresponding label (from a vocabulary of K classes) and N is the size of the dataset. The classifier network has the form $\tilde{\mathbf{y}} = M(\mathbf{x}; \theta, V) = H(F(\mathbf{x}; \theta); V)$, where we explicitly distinguish between *feature extractor* $F(\mathbf{x}; \theta)$, parametrized by θ, and *classifier* $H(\mathbf{u}; V) = \mathcal{A}(V\mathbf{u})$, where V is a matrix projecting the output of the feature extractor $\mathbf{u}$ to the class scores (in the following we omit parameters θ and V), and $\mathcal{A}$ is the softmax function that normalizes the scores to class probabilities. During training we minimize the cross-entropy loss between true labels and predictions $\mathcal{L}_{\text{CE}}(\mathcal{D}) = -\Sigma_{i=1}^{N} \mathbf{y}_i \cdot \log \tilde{\mathbf{y}}_i$, where $\mathbf{y}_i$ is the one-hot representation of class label $y_i \in \mathcal{C}$.

Continual learning. We consider the continual learning setting where T classification tasks are learned independently and in sequence from the corresponding datasets $\mathcal{D}_1, \ldots, \mathcal{D}_t, \ldots, \mathcal{D}_T$. The resulting model M_t after learning

Figure 2. Canonical Correlation Analysis (CCA) similarity of different continual learning methods performed on equally distributed 4-task scenario on CIFAR-100. The vertical axis shows the evolution over time of the correlation for given task activations. The horizontal axis shows correlation at different layers of the network.

task t has feature extractor F_t and classifier H_t . We assume that the classes in each task are disjoint, i.e. $\mathcal{C}_t \bigcap \mathcal{C}_{t'} = \emptyset$ for all $t' \neq t$. Ideally, after learning task t, the model can perform inference on all tasks $t' \leq t$ (i.e. it remembers current and previous tasks). We consider class-incremental learning in this work, where task-ID is unknown and it requires predictions over all the classes learned so far.

3.2. Forgetting analysis of various methods

Fine-tuning. In Figure 2 (far left) we illustrate the effect of continual learning (via simply fine-tuning the network on new tasks) on features extracted at different layers of the network. Forgetting is measured using Canonical Correlation Analysis (CCA) similarity* between the features extracted for task $t' \leq t$ by model M_t and the optimal model $M_{t'}$ (i.e. trained at time t' with $\mathcal{D}_{t'}$). Earlier features remain fairly correlated, while the correlation decreases progressively with increasing layer depth. This suggests that forgetting in higher-level features is more pronounced, since they become progressively more task-specific, while lower features are more generic.

Learning without forgetting. A popular method to prevent forgetting is *Learning without Forgetting* (LwF) [23], which keeps a copy of the model M_{t-1} before learning the new task and distills its predicted probabilities into the new model M_t (which may otherwise suffer interference from the current task t). In particular, LwF uses a modified cross-entropy loss over each head of previous tasks given by $\mathcal{L}_{\text{LwF}}(\mathcal{X}_t) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{X}_t} \Sigma_{j=1}^{t-1} \tilde{\mathbf{y}}^{t-1,j} \cdot \log \tilde{\mathbf{y}}^{t,j}$.

Note that the probabilities $\tilde{\mathbf{y}}^{t-1,j}$ and $\tilde{\mathbf{y}}^{t,j}$ are always estimated with current input samples $\mathbf{x} \in \mathcal{X}_t$, since data from previous tasks is not available. Since tasks are different, there is a distribution shift in the visual domain (i.e. $\tilde{\mathbf{y}}^{t-1,j}$ if extracted from $\mathbf{x} \in \mathcal{X}_{t-1}$ instead of $\mathbf{x} \in \mathcal{X}_t$), which can reduce the effectiveness of distillation when the domain shift from $\mathcal{X}_{t-1}$ to $\mathcal{X}_t$ is large. Figure 2 shows how LwF helps to increase the CCA similarity for previous tasks at the classifier, effectively alleviating forgetting and maintaining higher accuracy for previous tasks than fine tuning. However, the correlation at middle and lower-level layers in the feature extractor remains similar or lower to the case of fine tuning. This may be caused by the fact that the distillation constraint on the probabilities is too loose to enforce correlation in intermediate features.

Generative image replay. The lack of training images for previous tasks in continual learning has been addressed with a generator of images from previous tasks and using them during the training of current and future tasks [34, 35, 46, 49]. We consider conditional GAN with Projection Discriminator [33], which can control the class of generated images. At time t, the image generator samples images $\hat{\mathbf{x}} = G_{t-1}(c, \mathbf{z})$ where c is the desired class and $\mathbf{z}$ is a random latent vector sampled from a simple distribution (typically a normalized Gaussian). These generated images are combined with current data in an augmented dataset $\mathcal{D}'_t = \{(\hat{\mathbf{x}}_i, y_i)\}_{i=1}^{N_R} \cup \mathcal{D}_t$, where $\hat{\mathbf{x}}_i = G_{t-1}(y_i, \mathbf{z}_i)$ and N_R is the number of replayed images for previous tasks (typically distributed uniformly across tasks and classes).

Generative image replay, while appealing, has numerous limitations in practice. First, real images are high dimensional representations and the image distribution of a partic-

*CCA similarity computes the similarity between distributed representations even when they are not aligned. This is important, since learning new tasks may change how different patterns are distributed in the representation. We use SVCCA [40] which first removes noise using singular value decomposition (SVD).

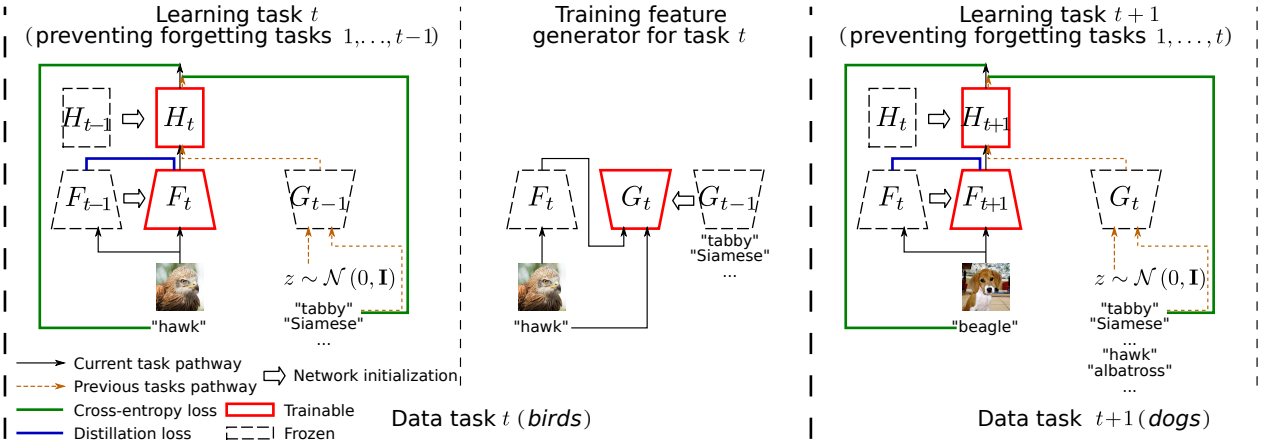

Figure 3. Proposed framework. Distillation and feature generation are used during training to prevent forgetting previous tasks. Once the task is learned, the feature generator is updated with adversarial training and distillation to prevent forgetting in the generator.

ular task lies in a narrow yet very complex manifold. This complexity requires deep generators with many parameters and are computationally expensive, difficult to train, and often highly dependent on initialization [26]. Training these models requires large amounts of images, which is rarely the case in continual learning. Even with enough training images, the quality of the generated images is often unsatisfactory as training data for the classifier, since they may not capture relevant discriminative features. Figure 2 shows the CCA similarity for class-conditional GAN. It shows a similar pattern to LwF and fine tuning with the similarity decreasing especially in intermediate layers.

4. Feature distillation and generative feature replay

In the previous analysis of forgetting in neural networks, we saw that generative image replay yields unsatisfactory results when applied to datasets that are difficult to generate (like CIFAR-100). We also observed that feature distillation prevents forgetting in the feature extractor. Therefore, to obtain the advantage of replay methods, which do not have the imbalance problem arising from multiple classification heads, we propose *feature* replay as an alternative to image replay. We combine feature distillation and feature replay in a hybrid model that is effective and efficient. (see Figure 1 right). Specifically, we use distillation at the output of the feature extractor in order to prevent forgetting in the feature extractor, and use feature replay of the same features to prevent forgetting in the classifier and to circumvent the classifier imbalance problem. Note that feature distillation has also been used in other applications [32, 47, 55].

Our framework consists of three modules: feature extractor, classifier, and feature generator. To prevent forgetting we also keep a copy of the feature extractor, classifier and feature generator from the previous set of tasks. Figure 3 illustrates continual learning in our framework. The classifier H_t and feature extractor F_t for task t are implicitly initialized with H_{t-1} and F_{t-1} (which we duplicate and freeze) and trained using feature replay and feature distillation. When the feature extractor and classifier are trained, we then freeze them and then train the feature generator G_t. A detailed algorithm is given in Algorithm 1.

Algorithm 1 : Class-incremental task learning.

Input: Sequence $\mathcal{D}_1, \ldots, \mathcal{D}_T$, where $\mathcal{D}_t = (\mathcal{X}_t, \mathcal{C}_t)$.
Require: Feature extractor F_0, Classifier H_0, Generator G_0. All trained end-to-end.
for $t = 1, \ldots, T$
  if $t = 1$
    Step 1: Train F_1 and H_1 with $\mathcal{D}_1$.
    Step 2: Train G_1 with $\mathbf{u}_1 = F_1(\mathbf{x}_i), \forall x_i \in \mathcal{D}_1$.
  else
    Step 3: Train F_t and H_t with $\mathcal{D}_t$ and generated features $\hat{\mathbf{u}}_{t'} = G_{t-1}(\mathcal{C}_{t'}, \mathbf{z})$, where $\mathcal{C}_{t'}$ is all previous classes.
    Step 4: Train G_t with $\mathbf{u}_t = F_t(\mathbf{x}_i), \forall x_i \in \mathcal{D}_1$ and $\hat{\mathbf{u}}_{t'} = G_{t-1}(\mathcal{C}_{t'}, \mathbf{z})$
end for

4.1. Feature generator

To prevent forgetting in the classifier we train a feature generator G_t to model the conditional distribution of features $p_{\mathbf{u}}(\mathbf{u}|c)$ as $\hat{\mathbf{u}} = G_t(c, \mathbf{z})$, and sample from it when learning future tasks. We consider two variants: *Gaussian class prototypes*, *conditional GAN with replay alignment*.

Gaussian class prototypes. We represent each class c of a task t as a simple Gaussian distribution $G_t(c, \mathbf{z}) = \mathcal{N}(\mathbf{u}; \mu_t^{(c)}, \mathbf{\Sigma}_t^{(c)})$, where $\mathcal{N}(\cdot; \cdot, \cdot)$ is a Gaussian distribution whose parameters are estimated using

$\{\mathbf{u}_i = F_t(\mathbf{x}_i), \forall(\mathbf{x}_i, y_i) \in \mathcal{D}_t, y_i = c\}$. This variant has the advantage of compactness and efficient sampling.

Conditional GAN with replay alignment. To generate more complex distributions and share parameters across classes and tasks, we propose to generate the feature extractor distribution with GANs. We use the Wasserstein GAN and adapt it to feature generation and continual learning using the following losses (between learning tasks t and $t+1$):

$$\mathcal{L}_{D_t}^{\text{WGAN}}(\mathcal{X}_t) = +\mathbb{E}_{\mathbf{z}\sim p_z, c\in C_t}[D_t(c, G_t(c,\mathbf{z}))] - \mathbb{E}_{\mathbf{u}\sim\mathcal{D}_t}[D_t(c, F_t(\mathbf{x}))] \quad (1)$$

$$\mathcal{L}_{G_t}^{\text{WGAN}}(\mathcal{X}_t) = -\mathbb{E}_{\mathbf{z}\sim p_z, c\in C_t}[D_t(c, G_t(c,\mathbf{z}))]. \quad (2)$$

A replay alignment loss $\mathcal{L}_{G_t}^{\text{RA}}$ is also added:

$$\mathcal{L}_{G_t}^{\text{RA}} = \Sigma_{j=1}^{t-1}\Sigma_{c\in C_j}\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left[\|G_t(c,\mathbf{z}) - G_{t-1}(c,\mathbf{z})\|_2^2\right]. \quad (3)$$

which can be seen as a type of distillation [49]. This replay alignment loss encourages the current generator G_t to replay exactly the same features as G_{t-1} when conditioned on a given previous class c and a given latent vector $\mathbf{z}$. We use a discriminator D_t during the adversarial training, which alternates updates of D_t and G_t (i.e. $\min_{D_t}\mathcal{L}_{D_t}^{\text{WGAN}}(\mathcal{X}_t)$ and $\min_{G_t}\mathcal{L}_{G_t}^{\text{WGAN}}(\mathcal{X}_t) + \mathcal{L}_{G_t}^{\text{RA}}$, respectively).

4.2. Feature extractor with feature distillation

We prevent forgetting in F_t by distilling the features extracted by F_{t-1} via the following L2 loss:

$$\mathcal{L}_{F_t}^{\text{FD}}(\mathcal{X}_t) = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}_t}[\|F_t(\mathbf{x}) - F_{t-1}(\mathbf{x})\|_2]. \quad (4)$$

Note that there are no separate losses for each head (like in [23]) because the feature $\mathbf{u} = F(\mathbf{x})$ is shared among tasks. Also, the loss can be applied on any feature (e.g. tensors). Note in Fig. 2 (center) how the CCA similarity of our approach compared to LwF increases, which indicates that there is less forgetting.

4.3. Algorithm of class-incremental learning

We are interested in a single head architecture that provides well-calibrated, task-agnostic predictions, which naturally arises if all tasks are learned jointly when all data is available. In our case we extend the last linear layer V_{t-1} to V_t by increasing its size to accommodate the new classes C_t. The softmax is also extended to this new size. During training we combine the available real data for the current task (fed to F_t) with generated features for previous tasks $\{(\hat{\mathbf{u}}_i, y_i)\}_{i=1}^{N_R}$. Since we only train a linear layer with features, this process is efficient.

Figure 2 (far right) shows that our method preserves similar representations for previous tasks at all layers, including the classifier. Our combination of distillation and replay maintains higher accuracy across all tasks, effectively addressing the problems of forgetting and task aggregation.

5. Experimental results

We report experiments evaluating the performance of our approach compared to baselines and the state-of-the-art.

Datasets. We evaluate performance on ImageNet [8] and CIFAR-100 [20]. ImageNet-Subset contains the first 100 classes in ImageNet in a fixed, random order. We resize ImageNet images to 256×256, randomly sample 224×224 crops during training, and use the center crop during testing. CIFAR-100 images are padded with 4 pixels, from which 32×32 crops are randomly sampled. The original center crop is used for testing. Random horizontal flipping is used as data augmentation for both datasets.

Training. We use Pytorch as our framework [37]. For CIFAR-100, we modify the ResNet-18 network to use 3×3 kernels for the first convolutional layer and train the model from scratch[†]. We train each classification task for 201 epochs and GANs for 501 epochs. For ImageNet, we use ResNet-18 and also train the model from scratch. We train each classification task for 101 epochs and GANs for 201 epochs. The Adam optimizer is used in all experiments, and the learning rate for classification and GANs are 1e-3 and 1e-4, respectively. The classes for both datasets are arranged in a fixed random order as in [14, 41]. The coefficient of distillation loss is set to 1.

Evaluation. The first evaluation metric is the average overall accuracy as in [14, 41]. It is computed as the average accuracy of all tasks up to the current task. The second evaluation metric is the average forgetting measure as in [4]. It defines forgetting for a specific task as the difference between the maximum accuracy achieved on that task throughout the learning process and the accuracy the model currently achieves for it. The average forgetting is computed by averaging the forgetting for all tasks up to the current one. More evaluation metrics can be found in [9, 22]

5.1. Class-incremental learning experiments

We first compare our approach with other methods on ImageNet-Subset and CIFAR-100. We use half of the classes from each dataset as the first task and split the remaining classes in 5, 10 and 25 tasks with equally distributed classes (as also done in [14]). In figures and tables "Ours Gaussian" indicates our method with Gaussian replay and "Ours" indicates our method with generative feature replay. We compare our approach with several methods: LwF [23], EWC [19], MAS [1], iCaRL [41] and Rebalance [14]. iCaRL-CNN uses a softmax classifier while iCaRL-NME uses the nearest mean classifier. The first three methods are trained without exemplars and iCaRL and Rebalance store 20 samples per class. For the first three methods, we train a multi-head network, where each task has

[†]This network setting was also used for the computation of Figure 2.

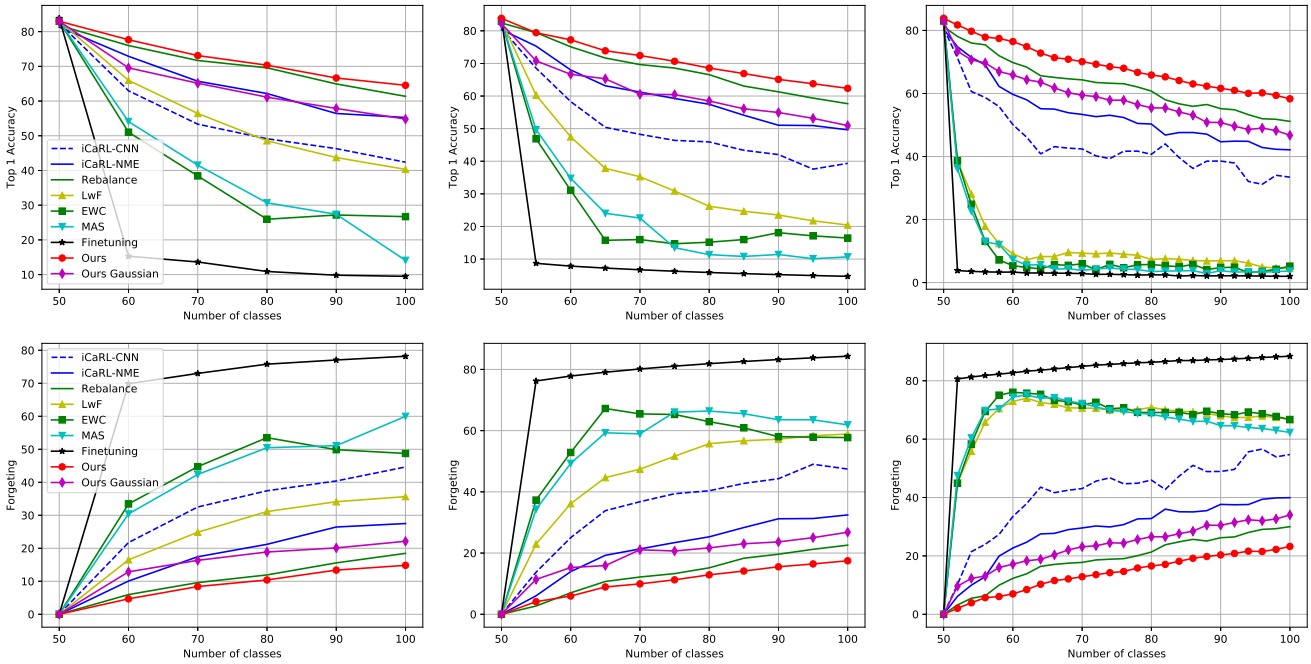

Figure 4. Comparison in the average accuracy (Top) and the average forgetting (Bottom) with various methods on ImageNet-Subset. The first task has the half number of classes, and the remaining classes are divided into 5, 10, 25 tasks respectively. The lines with symbols are methods without using any exemplars, and without symbols are methods with 2000 exemplars. (Joint Training: 77.6)

Table 1. Memory use comparison between exemplar-based methods, generative image replay (MeRGAN), and Ours.

Method	Datasets	Image Size	Exemplar	ResNet-18	GAN
Exemplar-based	CIFAR-100	32x32x3	2000 (6.2 Mb)	42.8 Mb *	–
	ImageNet-100	256x256x3	2000 (375 Mb)	45 Mb	–
	ImageNet-1000	256x256x3	20000 (3.8 Gb)	45 Mb	–
MeRGAN	–	–	–	45 Mb	8.5 Mb
Ours	–	–	–	45 Mb	4.5 Mb

a separate head since they will not work with single-head when there are no exemplars. We simply pick the maximum probability across all heads as the chosen output.

Comparative analysis on ImageNet-Subset. We report the average accuracy and the average forgetting on ImageNet-Subset in Figure 4. It is clear that using exemplars for iCaRL and Rebalance is superior to most methods without exemplars, such as LwF, MAS and EWC. Our method with Gaussian replay performs similarly to iCaRL-NME and much better than iCaRL-NME in the 5 and 10 task setting. Surprisingly, it outperforms both iCaRL-CNN and iCaRL-NME by a large margin in the 25-task setting. By using GANs for replay, our method shows significant improvement compared to Gaussian replay and outperforms the state-of-the-art method Rebalance by a large margin. The gain increases with increasing number of tasks. It achieves the best results in all settings in terms of both average accuracy and forgetting. It is important to note that for our methods we do not need to store any exemplars from previous tasks and generated features are dynamically combined with current data. A comparison with other methods on ImageNet-1000 is in the supplementary material.

Comparative analysis on CIFAR-100. Results for CIFAR-100 are shown in Figure 5. Our method with generative feature generation outperforms iCaRL, LwF, MAS and EWC by a large margin and achieves comparable results as Rebalance in the case of 5 and 10 tasks. We achieve slightly worse results in the 25-task setting compared to Rebalance, which might be because features from low resolution images are not as good as those learned from ImageNet. In contrast, for both iCaRL and Rebalance, 2000 exemplars in total must be stored. It is interesting that our method with Gaussian replay performs quite well compared to iCaRL, but slightly worse than Rebalance.

5.2. Comparison of storage requirements

In Table 1 we compare the memory usage of exemplar-based methods iCaRL [41] and Rebalance [14], the generative image replay method MeRGAN [49], and our generative feature replay. Exemplar methods normally store 20 images per class (from ImageNet or CIFAR-100), and the memory needed thus increases dramatically from 6.2MB to 375MB for 100 classes. Our approach, however, requires only a constant memory of 4.5MB for the generator and discriminator. For $256\times256\times3$ images, our model is equivalent to only 24 total exemplars. Note that it is hard for exemplar-based methods to learn with only 24 exemplars.

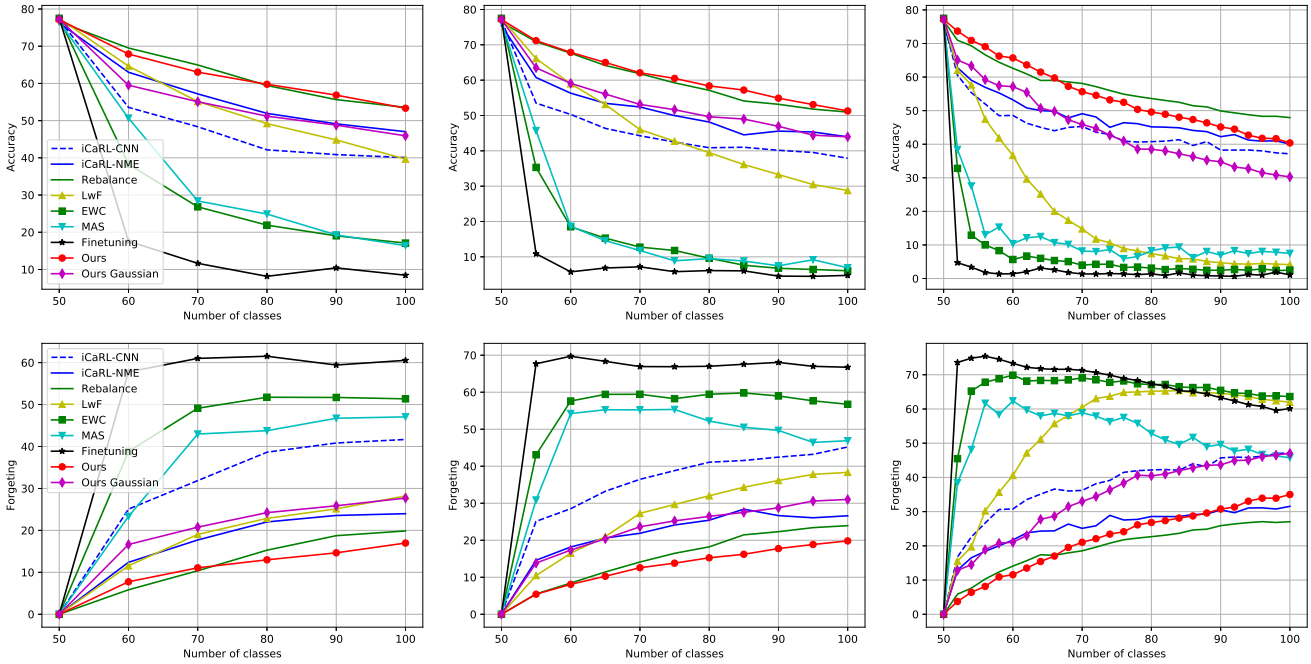

Figure 5. Comparison in the average accuracy (Top) and the average forgetting (Bottom) with various methods on CIFAR-100. The lines with symbols are methods without using any exemplars, and without symbols are methods with 2000 exemplars. (Joint Training: 72.0)

For larger numbers of classes such as full ImageNet-1000, it takes 3.8GB to store 20 samples per class. MeRGAN requires 8.5MB of memory, which is almost double the memory usage of ours. However, MeRGAN has difficulty generating realistic images for both CIFAR-100 and ImageNet and therefore obtains inferior results.

5.3. Generation of features at different levels

For our ablation study we use CIFAR-100 with 4 tasks with an equal number of classes. In Table 2 we look for the best depth of features to apply replay and distillation. We found that replaying at shallower depth results in dramatically lower performance. This is probably caused by: (1) the complexity of generating convolutional and lower-level features compared to the generation of linear high-level features from Block 4 (Ours); and (2) the difficulty of keeping the head parameters unbiased towards the last trained task when moving replay down in the network.

Table 2. Ablation study of replaying different features on CIFAR-100 for the 4-task scenario. For generative image replay, we use MeRGAN [49], Blocks 1, 2, and 3 are the features after the corresponding residual block in ResNet. Block 4 is the high-level linear features for our method. Average accuracy of all tasks is reported.

	T1	T2	T3	T4
Image (MeRGAN)	82.4	37.7	17.8	9.7
Block 1	80.7	41.6	26.5	20.1
Block 2		41.0	26.5	20.0
Block 3		51.1	37.0	26.6
Block 4 (Ours)		57.6	48.2	41.5

6. Conclusions

We proposed a novel continual learning method that combines generative feature replay and feature distillation. We showed that it is computationally efficient and scalable to large datasets. Our analysis via CCA shows how catastrophic forgetting manifests at different layers. The strength of our approach relies on the fact that the distribution of high-level features is significantly simpler than the distribution at the pixel level and therefore can be effectively modeled with simpler generators and trained on limited samples. We perform experiments on the ImageNet and CIFAR-100 datasets. We outperform other methods without exemplars by a large margin. Notably, we also outperform storage-intensive methods based on exemplars in several settings, while the overhead of our feature generator is small compared to the storage requirements for exemplars. For future work, we are especially interested in extending the theory to feature replay for continual learning of embeddings [54].

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.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2, 6

[2] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *ICCV*, pages 583–592, 2019. 3

[3] Andrew Brock, Jeff Donahuey, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 3

[4] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547, 2018. 2, 6

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *ICLR*, 2019. 3

[6] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *NeurIPS*, 2017. 3

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019. 2

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 6

[9] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. 6

[10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 3

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. 3

[12] Guillermo L Grinblat, Lucas C Uzal, and Pablo M Granitto. Class-splitting generative adversarial networks. *arXiv preprint arXiv:1709.07359*, 2017. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 1

[14] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2, 3, 6, 7

[15] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetful learning for domain expansion in deep neural networks. In *AAAI*, 2018. 2

[16] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 3

[17] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *ICLR*, 2018. 3

[18] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, pages 1857–1865, 2017. 3

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *pnas*, page 201611835, 2017. 2, 6

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[21] Jeongtae Lee, Jaehong Yun, Sungju Hwang, and Eunho Yang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018. 2

[22] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020. 6

[23] Zhizhong Li and Derek Hoiem. Learning without forgetting. *pami*, 40(12):2935–2947, 2018. 2, 4, 6

[24] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *ICPR*, 2018. 2

[25] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, pages 6467–6476, 2017. 2, 3

[26] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal. *A Large-Scale Study. ArXiv e-prints*, 2(4), 2017. 5

[27] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generationwith fewer labels. In *ICML*, 2019. 3

[28] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, pages 67–82, 2018. 2

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

[30] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer. Ternary feature masks: continual learning without any forgetting. *arXiv preprint arXiv:2001.08714*, 2020. 3

[31] Michael McCloskey and Neal 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

[32] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *ICCV Workshops*, pages 0–0, 2019. 5

[33] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 4

[34] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *ICLR*, 2018. 4

[35] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jähnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, 2019. 4

[36] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019. 2

[37] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017. 6

[38] Lorenzo Pellegrini, Gabrile Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. *arXiv preprint arXiv:1912.01100*, 2019. 3

[39] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional GANs for image editing. In *NeurIPS 2016 Workshop on Adversarial Training*, 2016. 3

[40] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NeurIPS*, pages 6076–6085, 2017. 4

[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542. IEEE, 2017. 2, 3, 6, 7

[42] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, pages 1060–1069, 2016. 3

[43] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. 2

[44] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[45] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4555–4564, 2018. 2, 3

[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, pages 2990–2999, 2017. 2, 3, 4

[47] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374, 2019. 5

[48] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 2

[49] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: learning to generate images from new categories without forgetting. In *NeurIPS*, 2018. 2, 3, 4, 6, 7, 8

[50] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2, 3

[51] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *CVPR*, pages 5542–5551, 2018. 3

[52] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-VAEGAN-D2: A feature generating framework for any-shot learning. In *CVPR*, 2019. 3

[53] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *ICCV*, pages 6619–6628, 2019. 3

[54] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 8

[55] Lu Yu, Vacit Oguz Yazici, Xialei Liu, Joost van de Weijer, Yongmei Cheng, and Arnau 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

[56] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995. JMLR. org, 2017. 2

[57] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xialei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, pages 5908–5916, 2017. 3

[58] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2242–2251, 2017. 3