Class-Incremental Learning with Generative Classifiers

Gido M. van de Ven[1,2,*], Zhe Li[1] & Andreas S. Tolias[1,3]

[1] Center for Neuroscience and Artificial Intelligence, Baylor College of Medicine, Houston, Texas, USA
[2] Computational and Biological Learning Lab, University of Cambridge, Cambridge, United Kingdom
[3] Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA

Abstract

Incrementally training deep neural networks to recognize new classes is a challenging problem. Most existing class-incremental learning methods store data or use generative replay, both of which have drawbacks, while 'rehearsal-free' alternatives such as parameter regularization or bias-correction methods do not consistently achieve high performance. Here, we put forward a new strategy for class-incremental learning: generative classification. Rather than directly learning the conditional distribution $p(y|\boldsymbol{x})$, our proposal is to learn the joint distribution $p(\boldsymbol{x}, y)$, factorized as $p(\boldsymbol{x}|y)p(y)$, and to perform classification using Bayes' rule. As a proof-of-principle, here we implement this strategy by training a variational autoencoder for each class to be learned and by using importance sampling to estimate the likelihoods $p(\boldsymbol{x}|y)$. This simple approach performs very well on a diverse set of continual learning benchmarks, outperforming generative replay and other existing baselines that do not store data.

1. Introduction

Deep neural networks excel in supervised learning tasks, but only when all the classes to be learned are available at the same time. Incrementally training a deep neural network to distinguish between a gradually growing number of classes has turned out to be very challenging [12, 43, 48, 50]. Successful strategies for class-incremental learning generally either rely on storing a subset of the past data and/or on replaying (representations of) past data, both of which have important disadvantages. Storing data is not always possible in practice (*e.g.* due to safety/privacy concerns or because of limited storage capacity), while replay — or rehearsal — is computationally expensive as it involves constant retraining on past data.

These drawbacks have sparked recent interest in 'rehearsal-free' continual learning [32], in which storing data or using replay are not allowed. In the past few years several methods have been proposed that can do class-incremental learning without replay or stored data [7, 19, 31, 34]. However, those methods rely on protocols with explicit task boundaries and/or their performance critically depends on the availability of a suitably pre-trained feature extractor.

In this paper, we put forward generative classification as a promising new strategy for class-incremental learning. Specifically, instead of training neural networks to directly learn the conditional distribution $p(y|\boldsymbol{x})$, we propose to train them to learn the joint distribution $p(\boldsymbol{x}, y)$, factorized as $p(\boldsymbol{x}|y)p(y)$, and then to perform classification using Bayes' rule. A key benefit of this strategy is that it rephrases a challenging class-incremental learning problem as a more easily addressable task-incremental learning problem (see Section 4.1).

To demonstrate the potential of generative classification for class-incremental learning, as a proof-of-principle we implement this strategy by training a variational autoencoder model for each class to be learned and by using importance sampling to estimate the class-conditional likelihoods during inference. We find that such a straightforward implementation of a generative classifier performs very well on a diverse range of class-incremental learning problems, outperforming generative replay and existing rehearsal-free methods. Moreover, this approach does not use replay, it does not store data, it can be applied to arbitrary class-incremental data streams (*i.e.* no need for task boundaries) and it does not rely on pre-trained networks, although if available those can be used effectively.

2. Problem formulation

In continual or incremental learning, an algorithm does not have access to all data at the same time, but it encounters the data in a sequence [13, 17, 41]. Recently, three different types, or 'scenarios', of continual learning have been described [52]: in task-incremental learning an algorithm must incrementally learn a set of clearly distinct tasks, in domain-incremental learning an algorithm must

*Corresponding author: ven@bcm.edu

learn the same task but with changing contexts, and in class-incremental learning an algorithm must incrementally learn to distinguish between a growing number of classes. In this paper, we focus on class-incremental learning, which is generally considered to be the most challenging continual learning scenario [6, 35, 42].

2.1. Class-incremental learning

There are various different ways in which a class-incremental learning problem can be set up. This makes direct comparisons between studies challenging, even when they use the same datasets. We therefore start by discussing some important assumptions that vary between studies.

2.1.1 Task-based *vs.* task-free

The goal of class-incremental learning is to learn, given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$, a classification rule that maps an input $x \in \mathcal{X}$ to a predicted label $y \in \mathcal{Y}$. However, unlike in classical machine learning, the algorithm that must learn this mapping is not given access to the entire dataset at once. Instead, the data is made available according to a particular class-incremental protocol.

Task-based class-incremental learning A commonly used class-incremental learning protocol is to split up the dataset into distinct 'tasks' (or 'episodes'), whereby each task contains a different subset of classes [*e.g.* 43, 48, 52]. The algorithm is then sequentially given access to the data of each task (Figure 1A). Importantly, after transitioning from one task to the next, the data from the previous task is no longer available. During each task, the training data of that task could either be given to the algorithm all at once, or it might be presented according to a fixed stream that is not controlled by the algorithm (see Appendix B in [1]).

Task-free class-incremental learning It has been argued that task-based protocols are not representative of real-world problems, and that the community should shift its focus to 'task-free' continual learning [2, 3, 19, 56]. In a task-free protocol, the algorithm is presented with an arbitrary stream of data, without any prior knowledge about the structure of this stream (Figure 1B). Many existing methods for class-incremental learning cannot deal with this setting, because they rely on the presence of 'task boundaries' (see Table B.1 in [1] for an overview).

In general, benchmarks for task-free class-incremental learning need to include a protocol for how the data stream should be generated (*i.e.* they should specify when samples from each class are presented). An open, largely unaddressed research question relates to the development of a principled way to design such data streams. In this paper we side-step this question, because for the particular implementation of generative classifier considered here — with a separate generative model for each class — the actual class-incremental sequence of the data stream does not matter.

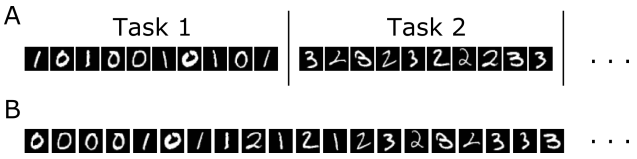

Figure 1. Schematic illustrating the distinction between (A) task-based and (B) task-free class-incremental learning.

Task-free continual learning has also been referred to as 'streaming' or 'online' continual learning. In that case, sometimes additional constraints are that each training sample should only be presented once and that the mini-batch size should be one [2, 19]. However, it is worth pointing out that these constraints relate to the sample efficiency of an algorithm and its robustness to noisy updates and, although they are topics worth studying, these are independent from the distinction between task-based and task-free class-incremental learning. For one of the benchmarks reported in this paper, we follow this more strict definition of streaming learning.

2.1.2 Other critical assumptions

Data storage An important assumption made by many class-incremental learning methods is that it is acceptable to store a limited amount of past samples in a memory buffer. The size of this memory buffer is typically one of the most important determinants of a method's performance [4, 42]. In practice, storing data is not always possible (*e.g.* safety or privacy concerns), and in this study we do not allow data storage, a setting which has been referred to as memoryless class-incremental learning [7].

Pre-training Another assumption commonly made in the class-incremental learning literature, especially by studies that do not allow storing data, is that a suitably pre-trained network or feature extractor is available or that there is an extended, non-incremental initialization phase that can be used for pre-training [*e.g.* 19, 30, 34, 50]. While the importance of the assumption about data storage seems to be widely acknowledged, this assumption about pre-training has received less attention. Here we investigate the importance of pre-training by considering both benchmarks with pre-trained networks available (CIFAR-100 and CORe50) and benchmarks without (MNIST and CIFAR-10).

3. Existing class-incremental learning methods

3.1. Methods relying on stored data

Many class-incremental learning methods store a subset of past data in a memory buffer. That data could be replayed when training on new data [10, 33, 46], they

could be used as exemplars or prototypes to guide classification decisions [12, 43] or they could be used in other ways [5, 21, 54]. Important questions when storing data are which samples to store [37, 39] and in what format [9, 18]. As discussed, we do not consider methods that store data.

3.2. Generative replay

If it is not possible to store data, an alternative is to replay generated 'pseudo-data' [45]. This strategy has been shown to be successful for toy problems with relatively simple inputs [48, 52], but it struggles on problems with more complex inputs, such as natural images [2, 28]. Some recent studies have shown competitive performance with generative replay on class-incremental learning problems with natural images [11, 30, 50], but the approaches in those studies depend on pre-trained networks (or on an extensive, non-incremental initialization phase [30]).

We include two generative replay methods in our comparison: deep generative replay [DGR; 48], which replays pixel-level representations, and brain-inspired replay [BI-R; 50], which replays latent feature representations.

3.3. Regularization-based methods

A popular strategy for continual learning is parameter regularization, which aims to minimize changes to parameters important for previously learned tasks. Examples of this strategy are elastic weight consolidation [EWC; 25] and synaptic intelligence [SI; 55]. Although it is well-established that these parameter regularization methods by themselves do not perform well in the class-incremental learning scenario [15, 22, 51], we include them in our comparison for completeness. Some regularization-based methods can be interpreted as performing approximate Bayesian inference on the parameters of the neural network [16, 25, 38] (*i.e*. Bayes' rule is used to find $p(\boldsymbol{\theta}|\mathcal{D})$, with $\mathcal{D}$ the observed data). Note that this is different from the generative classification strategy proposed in this paper, which uses Bayes' rule for the classification decision (*i.e*. to find $p(y|\boldsymbol{x})$).

3.4. Bias-correcting algorithms

When a standard softmax-based classifier is trained on a class-incremental learning problem, it ends up predicting only the most recently seen classes [29]. It has been argued that this is due to a bias in the output layer [6, 54], and several recent class-incremental learning methods aim to correct this bias by making the magnitude of the output weights of all classes comparable. Examples of this strategy are 'CopyWeights with Re-init' [CWR; 31] and its improved version CWR+ [34]. A disadvantage of these two methods is that they freeze the parameters of all hidden layers after the first task, so representation learning is limited. To address this, the method AR1 was proposed [34], which is similar to CWR+ except that it does not freeze the hidden layers but regularizes them using a modified version of SI.

There are also several bias-correcting algorithms that rely on stored data from previously seen classes [*e.g*. 5, 54], but as discussed we do not consider those methods here.

Related to these bias-correction algorithms, a trick to prevent large differences in the magnitude of the output weights between tasks in the first place, is to always only train on the classes from the current task (*i.e*. only include the output units of classes from the current task in the softmax-normalization, see Appendix A.1.5 in [1] for details). Zeno *et al*. [56] called this the 'labels trick'. A limitation of this trick is that there is no attempt to train the network to distinguish between classes from different tasks.

3.5. Other methods

Incremental linear discriminant analysis [23, 40] is a popular method in the data mining community that is suitable for class-incremental learning. Until recently, this method had largely been ignored in the continual learning community, likely because it can only learn a linear classifier. However, a recent study applied this method — now referred to as streaming linear discriminant analysis [SLDA; 19] — to the features extracted by a fixed, pre-trained deep neural network, which resulted in impressive performance on several class-incremental learning problems. The main disadvantage of SLDA is that it is not capable of representation learning, which means that its performance will likely heavily depend on the availability of suitably pre-trained networks. Here we test this: on the benchmarks in this paper for which no pre-trained networks are available, we apply SLDA directly on the input space.

4. Proposed strategy: generative classification

4.1. General framework & intuition

In deep learning, the typical approach to classification is to train a neural network to directly learn the conditional distribution $p(y|\boldsymbol{x})$ that we are interested in, for example by training a feed-forward classifier with a softmax output layer using cross-entropy loss. When all classes are available at the same time, this approach indeed works very well. In the incremental setting, however, this direct approach breaks down. A softmax classifier trained in the standard way heavily over fits to the most recently seen classes, a phenomenon referred to as catastrophic forgetting. A reason for this catastrophic forgetting is that, based on the most recently seen data, the empirical version of $p(y|\boldsymbol{x})$ — which the softmax classifier aims to learn — is indeed heavily biased towards the most recent classes. So far, as reviewed in Section 3, the dominant approach in the continual learning field has been to try to find methods and tricks to alleviate catastrophic forgetting.

Here we propose a shift of gears. Breaking with the traditional deep learning approach of training classifiers discriminatively, we propose to tackle class-incremental learning with generative classifiers. Rather than training deep neural networks to directly learn the conditional distribution $p(y|\boldsymbol{x})$, we propose to train them to learn the joint distribution $p(\boldsymbol{x}, y)$ — factorized as $p(\boldsymbol{x}|y)p(y)$ — and to use Bayes' rule for classification. The key benefit of this proposed strategy is that, in a class-incremental learning setting, based on the most recently seen data the empirical version of $p(\boldsymbol{x}|y)$ should not have any particular bias. Only the empirical version of $p(y)$ is biased, but learning this distribution without catastrophic forgetting is typically straightforward (*e.g.* the number of times each label is observed could be counted) or not needed (*e.g.* if it can be assumed that all labels have the same prior probability).

Class-incremental problem becomes task-incremental Another way to describe the benefit of the proposed generative classifier strategy is that it turns a challenging class-incremental learning problem into an easier task-incremental learning problem. This is the case because learning $p(\boldsymbol{x}|y)$ can be interpreted as a task-incremental problem whereby each 'task' consists of learning a class-conditional generative model for a specific label y. An important advantage of task-incremental learning is that it is possible to train networks with task-specific components [*e.g.* 36, 47, 53], or even to use completely separate networks for each task to be learned. This last insight is used for our proof-of-principle implementation of a generative classifier with a separate generative model for every class. Note however that it should be possible to use other task-incremental learning techniques to enable parameter sharing between these models (see also the discussion).

4.2. Implementation: VAEs & importance sampling

In this paper, to demonstrate the potential of the proposed generative classification strategy, we implement a generative classifier by training a variational autoencoder [VAE; 24] model for each class to be learned[1] and by using importance sampling to estimate the likelihoods $p(\boldsymbol{x}|y)$. For $p(y)$ we use a uniform distribution over all possible classes, as all benchmarks have an approximately equal amount of samples per class. In general, $p(y)$ could be learned from the data as well, for example by counting the number of times each class is observed in the training data.

4.2.1 Variational autoencoder

To learn the distribution $p(\boldsymbol{x}|y)$, we train a VAE model for each class to be learned. For the experiments on MNIST and CIFAR-10, a completely separate VAE model is learned for every class, while for the experiments on CIFAR-100 and CORe50 the lower, pretrained layers are shared between all models (see Section 4.3).

A VAE model consists of an encoder $q_{\boldsymbol{\phi}}$ that maps an input $\boldsymbol{x}$ to a posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ in latent space, a decoder $p_{\boldsymbol{\theta}}$ that maps a latent variable $\boldsymbol{z}$ back to a distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ in the input space and a prior distribution $p_{\text{prior}}(\boldsymbol{z})$. For the VAE models used in this paper, these distributions are given by:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{\mu}_{\boldsymbol{\phi}}^{(\boldsymbol{x})}, {\boldsymbol{\sigma}_{\boldsymbol{\phi}}^{(\boldsymbol{x})}}^2 I\right) \tag{1}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{\mu}_{\boldsymbol{\theta}}^{(\boldsymbol{z})}, I\right) \tag{2}$$

$$p_{\text{prior}}(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z} \,|\, \mathbf{0}, I\right) \tag{3}$$

whereby $\boldsymbol{\mu}_{\boldsymbol{\phi}}^{(\boldsymbol{x})}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}^{(\boldsymbol{x})}$ are the outputs of the encoder network when $\boldsymbol{x}$ is fed in, and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{(\boldsymbol{z})}$ is the output of the decoder network when $\boldsymbol{z}$ is fed in. For both the encoder network and the decoder network, we use deep neural networks. See Appendix A.3 in [1] for full details on the architectures that are used for the different benchmarks. Importantly, for each benchmark, the architecture of the VAE models is chosen so that the *total* number of parameters of the generative classifier is similar to the number of parameters used by generative replay.

The VAE models are trained by optimizing a variational lower bound to the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\text{prior}}(\boldsymbol{z}) d\boldsymbol{z}$. This lower bound, or ELBO, is given by:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}\right) &= E_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \\ &= E_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\text{prior}}(\boldsymbol{z}))\end{aligned} \tag{4}$$

where D_{KL} is the Kullback-Leibler divergence. Full details of the VAE training are given in Appendix A.2 in [1].

4.2.2 Importance sampling

To estimate the likelihoods $p(\boldsymbol{x}|y)$, we use importance sampling [8, 44]. This means that the likelihood of a test sample $\boldsymbol{x}$ under the VAE model of class y is estimated using:

$$p(\boldsymbol{x}|y) = \frac{1}{S} \sum_{s=1}^{S} \frac{p_{\boldsymbol{\theta}_y}\left(\boldsymbol{x}|\boldsymbol{z}^{(s)}\right) p_{\text{prior}}\left(\boldsymbol{z}^{(s)}\right)}{q_{\boldsymbol{\phi}_y}\left(\boldsymbol{z}^{(s)}|\boldsymbol{x}\right)} \tag{5}$$

whereby $\boldsymbol{\theta}_y$ and $\boldsymbol{\phi}_y$ are the parameters of the VAE model of class y, S is the number of importance samples and $\boldsymbol{z}^{(s)}$ is the s^{th} importance sample drawn from $q_{\boldsymbol{\phi}_y}(\boldsymbol{z}|\boldsymbol{x})$. For the results in Table 2, we use $S = 10,000$ importance samples

[1] Note that this setup could also be described as a single VAE model with class-specific masks whereby for each class a different, non-overlapping subset of parameters is unmasked.

Table 1. Overview of the benchmarks used in this paper. Each benchmark consists of an image dataset split up into a number of distinct tasks, with all tasks containing an equal number of classes. Such a task-based design is not needed for our generative classifier, but it is used to enable a comparison with other methods. Within each task, the training data is presented to the algorithm in a random, i.i.d stream, with the number of iterations per task and the mini-batch size being part of the benchmark. Another important aspect of each benchmark is whether pre-trained models are available. For all benchmarks considered in this paper, storing data is not allowed.

	Dataset Info		**Data-Stream Parameters**			**Pretrained**
	Classes	Image-type	Tasks	Iterations	Batch size	**Models?**
MNIST	10	28x28, grey	5	2000	128	-
CIFAR-10	10	32x32, RGB	5	5000	256	-
CIFAR-100	100	32x32, RGB	10	5000	256	ConvLayers
CORe50	10	128x128, RGB	5	single pass	1	ResNet18

for each likelihood estimation. The effect of reducing the number of importance samples is explored in Table 4.

Based on Bayes' rule: $p(y|\boldsymbol{x}) \propto p(\boldsymbol{x}|y)p(y)$, classification is then done using:

$$\hat{y}^{(\boldsymbol{x})} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(\boldsymbol{x}|y)p(y) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(\boldsymbol{x}|y) \qquad (6)$$

whereby $\hat{y}^{(\boldsymbol{x})}$ is the class label predicted by the generative classifier for test sample $\boldsymbol{x}$. Note that the last equality in Eq. 6 holds because, in this paper, $p(y)$ is modelled with a uniform distribution over all possible classes.

4.3. When pre-trained models are available: reconstruction loss in the feature space

The generative classifier approach described so far does not depend on the availability of pre-trained networks, as it is possible to train the full generative models from scratch. If pre-trained models are available, however, there are various ways in which they could be used. For example, suppose that pre-trained convolutional layers are available. One option would be to use these to initialize the convolutional layers of the encoder networks of the VAE models, and then to proceed with training in the standard way. Another option, which is the approach taken in this paper, is to use the pre-trained convolutional layers as a fixed feature extractor, and then to train the VAE models on the extracted features rather than on the raw inputs. An advantage of this second approach, which is reminiscent of recent studies that performed generative replay in the feature space [30, 50], is that it appears to be easier to learn good generative models for such extracted features, presumably because they are less complex than the raw inputs.

5. Experiments

In this section we test the above implementation of the proposed generative classification strategy on a diverse set of class-incremental learning benchmarks. On each benchmark, we compare our generative classifier with the applicable methods discussed in Section 3 that do not store data (see Appendix A.1 in [1] for technical details of all compared methods). As far as possible, we use the same "base network" architecture and the same training settings for all compared methods. Full details of the architectures and training settings used for each benchmark are provided in Appendix A.3 in [1]. Documented code for all experiments (including for all compared methods) is available online: https://github.com/GMvandeVen/class-incremental-learning.

5.1. Benchmarks

An overview of the benchmarks used in this paper is provided in Table 1. All benchmarks are set up as task-based, in order to be able to compare with current state-of-the-art class-incremental learning methods, even though our generative classifier can be applied to task-free protocols as well.[2] Important aspects of each benchmark are the number of tasks, the number of iterations per task, the mini-batch size and whether pre-trained models are available. For all benchmarks, within tasks the training data is always fed to the network in an i.i.d. stream, although some of the compared methods (EWC, and SLDA for the first task) additionally assume they can access a task's training data in one large batch (see Appendix B in [1]).

5.1.1 MNIST

The first benchmark is based on the MNIST dataset [27], which is split up into 5 tasks with 2 digits each. Following previous studies [22, 51], this benchmark has 2000 iterations per task and a mini-batch size of 128. The base network for this benchmark is a fully-connected network with 2 hidden layers of 400 ReLU units and a softmax output layer. No pre-training is used.

[2]For the specific implementation of the generative classifier used in this paper, with a separate model for each class, the performance does not depend on the specific class-incremental sequence at all. The reason is that the class-specific VAE models are trained only on samples of their own class, and it therefore does not matter if those classes are intermingled in certain ways.

Table 2. Final test accuracy (as %) of all compared methods on the different benchmarks. Evaluation is according to the "class-incremental learning scenario" or the "single-headed setting" (*i.e.* the model has to chose between all classes). Only methods that do not store data are included. All experiments were performed 10 times with different random seeds, reported are the means (± SEMs) over these runs.

Strategy	**Method**	**MNIST**	**CIFAR-10**	**CIFAR-100**	**CORe50**
Baselines	*None*	*19.92 (± 0.02)*	*18.74 (± 0.29)*	*7.96 (± 0.11)*	*18.65 (± 0.26)*
	Joint	*98.23 (± 0.04)*	*82.07 (± 0.15)*	*54.08 (± 0.27)*	*71.85 (± 0.30)*
Generative Replay	DGR	91.30 (± 0.60)	17.21 (± 1.88)	9.22 (± 0.24)	-
	BI-R	-	-	21.51 (± 0.25)	60.40 (± 1.04)
	BI-R + SI	-	-	34.38 (± 0.21)	62.68 (± 0.72)
Regularization	EWC	19.95 (± 0.05)	18.63 (± 0.29)	8.47 (± 0.09)	18.56 (± 0.31)
	SI	19.95 (± 0.11)	18.14 (± 0.36)	8.43 (± 0.08)	18.69 (± 0.26)
Bias-correction	CWR	32.48 (± 2.64)	18.37 (± 1.61)	21.90 (± 0.68)	40.28 (± 1.13)
	CWR+	37.20 (± 3.11)	22.32 (± 1.08)	9.34 (± 0.25)	40.12 (± 1.06)
	AR1	48.84 (± 2.55)	24.44 (± 1.08)	20.62 (± 0.45)	45.27 (± 1.02)
	Labels Trick	32.46 (± 1.95)	18.43 (± 1.31)	23.68 (± 0.26)	42.59 (± 1.03)
Other	SLDA	87.30 (± 0.02)	38.35 (± 0.03)	44.49 (± 0.00)	**70.80 (± 0.00)**
Generative Classifier		**93.79 (± 0.08)**	**56.03 (± 0.04)**	**49.55 (± 0.06)**	**70.81 (± 0.11)**

5.1.2 CIFAR-10 without pre-training

For this benchmark the CIFAR-10 dataset [26] is split up into 5 tasks with 2 classes each. The number of iterations per task for this benchmark is 5000 and the mini-batch size is 256. Following previous studies [2, 12, 33], the base network is a small version of ResNet18 [20] with three times less feature maps across all layers. No pre-training is used.

5.1.3 CIFAR-100 with pre-training on CIFAR-10

This benchmark is taken from the study that proposed BI-R [50]. The CIFAR-100 dataset [26] is split up into 10 tasks with 10 classes each. There are 5000 iterations per task with mini-batch size of 256. The base network is a convolutional neural network with 5 pre-trained convolutional layers followed by 2 randomly initialized fully-connected layers with 2000 ReLU units and a softmax output layer. The convolutional layers were pre-trained on CIFAR-10. To enable a direct comparison, we use the exact same pre-trained convolutional layers as in [50], which were made publicly available by the authors.

5.1.4 CORe50 with pre-training on ImageNet

The final benchmark is based on the CORe50 dataset [31]. This dataset is made up of image-frames cropped from short 15 second videos of moving objects. There are 10 different classes, with each class represented in the dataset by 5 different objects that were each filmed in 11 different environments. As in [14, 31], we use the images from eight of these environments for training and the others for testing. This results in approximately $10,500$ training images per class. The dataset is split up into 5 tasks with 2 classes each. This benchmark follows the more strict definition of streaming learning: each training image is presented by itself (*i.e.* mini-batch size of 1) and only once. Following [19], a standard ResNet18 pretrained on ImageNet is used as a fixed feature extractor. The base network on top of this feature extractor consists of one fully connected layer with 1024 ReLU units and a softmax output layer.

5.2. Results

Table 2 shows the performance of our generative classifier on the four benchmarks described above, along with the performance of the methods discussed in Section 3 that also do not store data. The generative classifier performed very strongly, comfortably outperforming all compared methods on three out of four benchmarks. Of special note are the substantial gaps with the generative replay variants, while these methods used similar number of parameters. Only on the CORe50 benchmark, in which an extensively pre-trained network was used and in which each sample was presented only once, the performance of the generative classifier was comparable to that of SLDA, while still substantially higher than that of the other compared methods.

An interesting result is that SLDA still performed competitively when it was applied directly on the raw inputs of MNIST and CIFAR-10. Although its performance was well below that of our generative classifier, it outperformed almost all other methods.

Another thing to note from these results is the modest performance of the bias-correction methods, especially on benchmarks where no pre-training was used. When pre-trained networks were available the relative performances

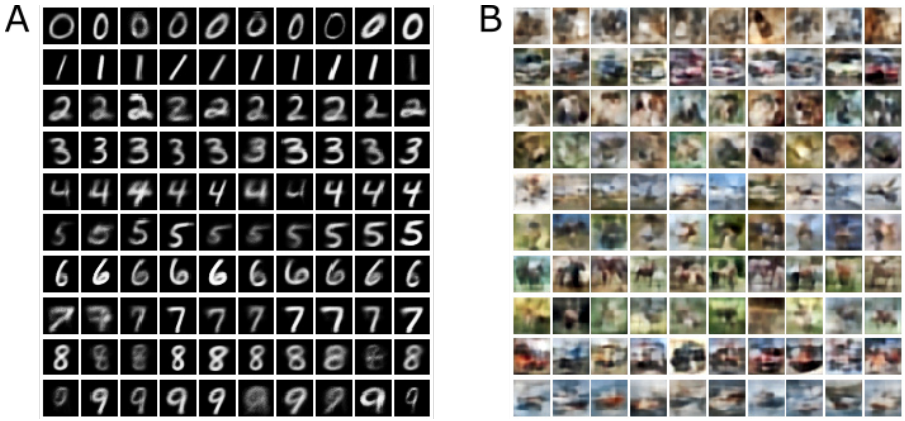

Figure 2. Samples randomly drawn from the VAE models of the generative classifier for (A) MNIST and (B) CIFAR-10.

Table 3. Comparison of the performance of the generative classifier with the performance of a softmax-based classifier discriminatively trained on samples from the VAE models of the generative classifier. Shown is the test accuracy (as %) over all classes. All experiments were performed 10 times with different random seeds, reported are the means (± SEMs) over these runs.

	MNIST	**CIFAR-10**	**CIFAR-100**	**CORe50**
Generative classifier	**93.79 (± 0.08)**	**56.03 (± 0.04)**	**49.55 (± 0.06)**	**70.81 (± 0.11)**
Discriminative classifier trained on generated samples	85.93 (± 0.43)	13.71 (± 0.61)	33.84 (± 0.14)	47.86 (± 1.77)

of these methods improved, but they did not come close to those of the best performing methods.

5.3. Generative classification *vs.* generative replay

An intriguing result from the above comparisons is that our generative classifier consistently and sometimes substantially outperformed generative replay. This suggests that directly using generative models to perform classification might be a better strategy than using those models indirectly to generate replay for discriminatively training a classifier. However, it could be argued that this conclusion is not completely warranted by these results, as both strategies did not use the exact same generative models (even though the total number of parameters was similar). For generative replay one large generative model was incrementally trained on all classes, while for the generative classifier a series of smaller, separate generative models was trained.

To more directly compare generative classification and generative replay, we trained — in an i.i.d. manner — a softmax-based classifier on samples generated by the VAE models of the generative classifier (see Appendix A.4 in [1] for full details on this experiment). Another way to phrase this experiment is that a discriminative classifier was trained exclusively with 'generative replay' produced by the same generative models as used by the generative classifier. For all benchmarks, we found that the generative classifier substantially outperformed the discriminative classifier that was trained on its own samples (Table 3). This suggests that, also when the same generative models are used, generative classification outperforms generative replay.

The results in Table 3 also indicate that the quality of the samples produced by the VAE models of our generative classifiers was not so good. To check this, we visualized samples drawn from the VAE models of the generative classifier for the MNIST and CIFAR-10 benchmarks (Figure 2). While for MNIST the generated samples look reasonable, for CIFAR-10 they are indeed not great. This thus indicates that competitive class-incremental learning performance could be obtained by a generative classifier even without high-quality generative models.

6. Discussion

Class-incremental learning is a challenging problem. So far the deep learning community has tackled this problem by directly learning a discriminative classifier, which only seems to work in combination with tricks such as pre-training, storing data or replay. Here we proposed an alternative strategy — to learn a generative classifier — and we showed that it can outperform generative replay and existing rehearsal-free methods.

An interesting finding from our comparison of class-incremental learning methods was the strong performance of SLDA [19]. It outperformed generative replay variants on three out of four benchmarks, and it achieved competitive performance even when applied directly on the raw inputs. We believe this strong performance can be explained

Table 4. Performance of generative classifier as function of number of importance samples used for inference. Shown is test accuracy (as %) over all classes. Experiments were performed 10 times with different random seeds, reported are the means (± SEMs) over these runs.

	$S = 1$	$S = 10$	$S = 100$	$S = 1,000$	$S = 10,000$
MNIST	91.14 (± 0.08)	92.46 (± 0.09)	93.25 (± 0.09)	93.62 (± 0.10)	93.79 (± 0.08)
CIFAR-10	50.86 (± 0.10)	54.64 (± 0.09)	55.43 (± 0.10)	55.83 (± 0.09)	56.03 (± 0.04)
CIFAR-100	45.02 (± 0.10)	48.45 (± 0.10)	49.26 (± 0.10)	49.48 (± 0.08)	49.55 (± 0.06)
CORe50	61.00 (± 0.19)	69.09 (± 0.14)	70.33 (± 0.14)	70.62 (± 0.14)	70.81 (± 0.11)

because SLDA can be interpreted as a generative classifier. SLDA learns a mean vector $\boldsymbol{\mu}_y$ for each class y and a covariance matrix Σ that is shared between all classes. The generative model that SLDA implicitly assumes for each class y is given by $p(\boldsymbol{x}|y) = \mathcal{N}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{\mu}_y, \Sigma\right)$. SLDA is however "a generative classifier in disguise" because it does not explicitly compute the likelihoods during inference, since with its assumptions the decision boundaries implied by the underlying generative models can be computed analytically.

The main disadvantage of SLDA is that it can only learn linear classifiers. To further improve upon SLDA, it seems necessary to find a way to do representation learning in a class-incremental way. This is exactly what our deep generative classifiers are able to do. Learning good representations is not easy, and it is not surprising that this ability comes at a cost of increased sample complexity. However, (complex) representation learning is not a necessary component of our proposed strategy. When the amount of training data is small, or when the representations provided by a pre-trained network are already good, it is probably better to learn relatively simple generative models. Indeed, SLDA's performance can be seen as the minimal attainable performance for a generative classifier, upon which can be improved when sufficient data is available.

Compared with generative replay, an important advantage of generative classifiers is that training is less costly, as replay is not necessary. On the flip-side, inference (*i.e.* making a classification decision) with generative classifiers is relatively costly, as it involves computing/estimating the likelihood of a test sample under the generative model of each possible class. For our specific implementation, this seems especially problematic because a large number of importance samples tends to be needed for high precision likelihood estimates with VAE models [49]. For the results reported in Table 2, we used $10,000$ importance samples for each likelihood estimation. However, we found that the number of importance samples could be lowered substantially without large drops in performance (Table 4). Even using just a single importance sample resulted in state-of-the-art class-incremental learning performance on three out of four benchmarks. Moreover, there are also other tricks that could speed up inference: it might be possible to use uncertainty estimates (which can be obtained from the generative models [37]) to inform the number of importance samples to use, or the classification decision could be made hierarchical (*e.g.* first decide whether it is a cat or a dog, then decide on the specific breed).

Another disadvantage of our specific implementation of the generative classifier is that a completely new generative model is learned for each new class. It could be questioned how scalable this is. In this regard, we believe it is important to point out three things. Firstly, to ensure a fair comparison between our generative classifier and generative replay, we controlled for the *total* number of parameters. Secondly, as illustrated by SLDA, even using small or minimal generative models for each class can result in competitive performance. Finally, and perhaps most importantly, the main point of this paper is to highlight the potential of generative classification for class-incremental learning: our implementation with independent VAE models is a proof-of-principle. For practical applications, the generative models of the different classes should probably share substantial parts of their networks. Such sharing introduces the risk of interference, but it also opens up the possibility of positive transfer between the generative models. Importantly, as pointed out in Section 4.1, learning the different class-conditional generative models is a task-incremental problem, which is an important simplification compared to the original class-incremental problem [52]. We therefore expect the question of how to optimally share parts of the generative models to be a fruitful topic for further research.

Acknowledgments

We thank Siddharth Swaroop and Martin Mundt for useful comments. This research project has been supported by the Lifelong Learning Machines (L2M) program of the Defence Advanced Research Projects Agency (DARPA) via contract number HR0011-18-2-0025 and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, IARPA, DoI/IBC, or the U.S. Government.

References

[1] Authors [...]. Supplementary material, 2021. Supplied as `suppl_material.pdf`.

[2] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019.

[3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019.

[4] Yogesh Balaji, Mehrdad Farajtabar, Dong Yin, Alex Mott, and Ang Li. The effectiveness of memory replay in large scale continual learning. *arXiv preprint arXiv:2010.02418*, 2020.

[5] Eden Belouadah and Adrian Popescu. Scail: Classifier weights scaling for class incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1266–1275, 2020.

[6] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 2020.

[7] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. Initial classifier weights replay for memoryless class incremental learning. In *British Machine Vision Conference*, 2020.

[8] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

[9] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptive quantization modules. In *International Conference on Machine Learning*, pages 1240–1250. PMLR, 2020.

[10] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[11] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. GAN memory with no forgetting. In *Advances in Neural Information Processing Systems*, 2020.

[12] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. *arXiv preprint arXiv:2009.00919*, 2020.

[13] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[14] Arthur Douillard and Timothée Lesort. Continuum: Simple management of complex continual learning scenarios. *arXiv preprint arXiv:2102.06253*, 2021.

[15] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

[16] Sebastian Farquhar and Yarin Gal. A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*, 2019.

[17] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

[18] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020.

[19] Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 220–221, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[21] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

[22] Yen-Chang Hsu, Yen-Cheng Liu, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[23] Tae-Kyun Kim, Björn Stenger, Josef Kittler, and Roberto Cipolla. Incremental linear discriminant analysis using sufficient spanning sets and its applications. *International Journal of Computer Vision*, 91(2):216–232, 2011.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] 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*, page 201611835, 2017.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *International Joint Conference on Neural Networks*. IEEE, 2019.

[29] Shuang Li, Yilun Du, Gido M van de Ven, Antonio Torralba, and Igor Mordatch. Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*, 2020.

[30] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020.

[31] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

[32] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-free continual learning over small non-iid

batches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 989–998, 2020.

[33] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6470–6479, 2017.

[34] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[35] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation. *arXiv preprint arXiv:2010.15277*, 2020.

[36] Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, pages E10467–75, 2018.

[37] Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *arXiv preprint arXiv:2009.01797*, 2020.

[38] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[39] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. In *Advances in Neural Information Processing Systems*, 2020.

[40] Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, 35(5):905–914, 2005.

[41] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[42] 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.

[43] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[44] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

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

[46] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

[47] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[48] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2994–3003, 2017.

[49] L Theis, A van den Oord, and M Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, 2016.

[50] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11:4069, 2020.

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

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

[53] Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Ali Geisa, Gido M van de Ven, Emily Chang, Chenyu Gao, Weiwei Yang, Bryan Tower, Jonathan Larson, Christopher M White, and Carey E Priebe. Omnidirectional transfer for quasilinear lifelong learning. *arXiv preprint arXiv:2004.12908*, 2020.

[54] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[55] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, 2017.

[56] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123v3*, 2019.