Looking through the past: better knowledge retention for generative replay in continual learning

Valeriya Khan[1], Sebastian Cygert[1,2], Bartłomiej Twardowski[1,3,4], and Tomasz Trzciński[1,5,6,7]
[1]IDEAS NCBR, [2]Gdańsk University of Technology, [3]Computer Vision Center, [4]Universitat Autònoma Barcelona, [5]Warsaw University of Technology, [6]Jagiellonian University, [7]Tooploox

Abstract

In this work, we improve the generative replay in a continual learning setting.We notice that in VAE-based generative replay, the generated features are quite far from the original ones when mapped to the latent space. Therefore, we propose modifications that allow the model to learn and generate complex data. More specifically, we incorporate the distillation in latent space between the current and previous models to reduce feature drift. Additionally, a latent matching for the reconstruction and original data is proposed to improve generated features alignment. Further, based on the observation that the reconstructions are better for preserving knowledge, we add the cycling of generations through the previously trained model to make them closer to the original data. Our method outperforms other generative replay methods in various scenarios.

1. Introduction

A popular setting for continual learning is Class Incremental Learning (CIL), where the goal is to train the classifier on new classes in consequent incremental steps [8]. Typically, different types of regularizations are applied [6, 12], however, without using any exemplars of the previous tasks, the results are far away from being satisfactory. Hence, there is an interest in generative models [1], which allow replaying the synthetic data from previous tasks using a trained generative model.

Generative replay models often have poor results on datasets with more complex data or a greater number of different classes [5]. This is mainly because modeling high-dimensional images in incrementally trained generative models is very challenging, as from task to task the quality of generated data degrades. Therefore, some recent works [7] incorporated feature-based replay when the data is first passed through the trained and frozen feature extractor, and only then it is used for training the generator part. One significant benefit of utilizing feature replay is that the distribution that needs to be learned by the generative model is usually much simpler and has lower dimensionality.

One of the recent works in the generative replay that utilizes the feature replay is Brain-Inspired Replay (BIR) [11]. This work performed several modifications to make variational autoencoder able to learn and generate more complex data, even in long sequences. Upon in-depth analysis, we have observed, that there is still a significant difference between features from the real data and those produced by the generator. We hypothesize that this may have a detrimental effect on the quality of the data replay, and hence we add two modifications to the model that mitigate the problem.

Overall, the main contributions of this work are threefold:

- We analyzed existing feature-generative replay methods for class-incremental learning and identified the weaknesses of recent VAE-based approaches, such as degraded generated samples and a mismatch in the distribution of current features and generated ones.
- We propose a new method for class-incremental learning with generative feature replay. Our method improves the matching of latent representations between reconstructed and original features through distillation, and using cycling of generative replay to effectively reduce the discrepancy between new and old samples for classification.
- Through a series of experiments, we demonstrate that our method significantly outperforms the baseline approach (BIR).

2. Method

2.1. Problem definition

In this work, we focus on image classification in a class-incremental setting. The model is trained on the sequence of tasks $T_1, T_2, ..., T_n$. The training data $\{X^{(t)}, Y^{(t)}\}$ is drawn from the distribution $D^{(t)}$, where $X^{(t)}$ are the training samples, $Y^{(t)}$ are the ground truth labels, and $1 \leq t \leq n$

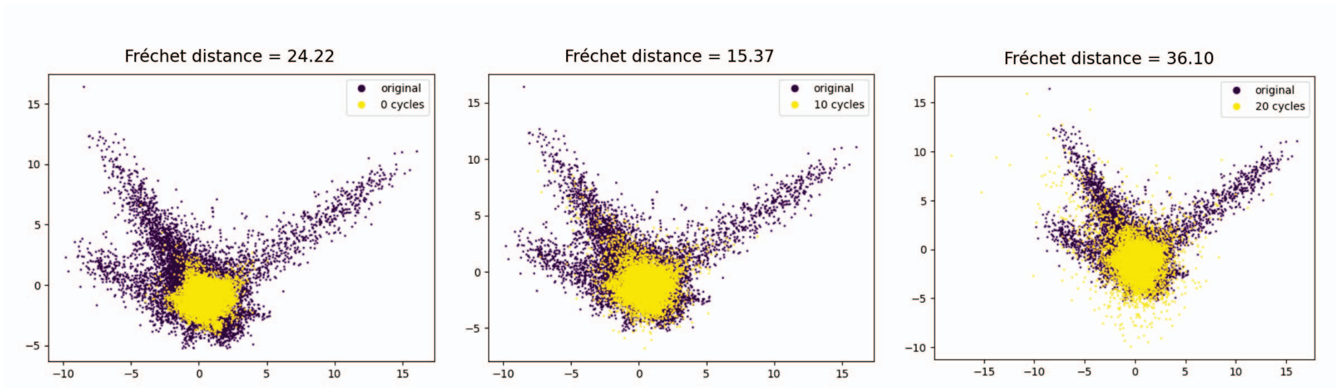

Figure 1: Principal Component Analysis (PCA) plots were computed on original latent vectors and generated ones when doing 0, 10, and 20 cycles respectively. By looking at both the PCA plots and Fréchet distances we can observe the generated latents are more aligned with the original ones when using an appropriate number of cycles.

is the current task id. In this context, the *task* means an isolated training phase with access only to this task data (cannot recall old data).

2.2. Baseline model

Our work is based on the Brain-Inspired Replay (BIR) method [11]. The model contains two main parts: a pre-trained feature extractor and a symmetrical VAE on top of it. The VAE is used as a feature generator in BIR to replay old knowledge. It consists of the encoder q_ϕ and the decoder p_ψ. The encoder maps the input x to stochastic latent variables z, and the decoder maps these latent variables back to reconstructed vector $\hat{x}$. Usually, a VAE model is trained by maximizing a variational lower bound on the evidence (ELBO), which is analogous to minimizing the following per-sample loss:

$$L^G(x;\phi,\psi) = E_{z\sim q_\phi(.|x)}[-\log p_\psi(x|z)] \\ +D_{KL}(q_\phi(.|x)||p(.)) = L^{recon}(x;\phi,\psi)+L^{latent}(x;\phi), \tag{1}$$

where $q_\phi(.|x) = \mathcal{N}(\mu^{(x)}, {\sigma^{(x)}}^2 I)$ and $p(.) = \mathcal{N}(0,I)$ are the posterior and prior distributions over the latent variables respectively, and D_{KL} is the Kullback-Leibler divergence.

To generate samples of specific classes, the standard normal prior is substituted by the Gaussian mixture with a separate distribution for each class.

For the current task, classification loss is given by:

$$L^C(x,y;\theta) = -\log p_\theta(\mathcal{Y}=y|x), \tag{2}$$

where p_θ is the conditional probability distribution defined by the parameters of the model.

For the replay part in BIR, the knowledge distillation loss is used instead of classification loss. Usually, knowledge distillation [4] is incorporated in transferring the knowledge from the teacher model to the student model. It is performed by minimizing the loss where the target is the result of the softmax function with temperature on the teacher model logits. The distillation loss is calculated as follows:

$$L^D(x,\tilde{y};\theta) = -T^2\sum_{c=1}^{N_{\text{classes}}}\tilde{y}_c \log p_\theta^T(\mathcal{Y}=x|x), \tag{3}$$

where T is the softmax temperature.

2.3. Improved feature replay

In this section, we describe three improvements that we propose to the base method that address particular problems with VAE-based feature replay: (1) reconstruction misalignment, (2) features drift in continual learning, (3) discrepancy between generated features and ones coming from the original data.

2.3.1 Latent matching for reconstructions and original data

The first modification that we add is an additional loss term for minimizing the difference between the latent vectors of the original sample and its reconstruction. In order to do that we pass the original sample x through the encoder and obtain the latent vector for the original sample z_o. Then we pass this latent vector through the decoder to get the reconstruction $\hat{x}$. After that, we pass the reconstruction through the encoder again and receive the latent vector z_r. We utilize the mean squared error (MSE) loss for measuring the difference between two vectors:

$$L^{\text{latent match}}(z_o;\phi,\psi) = -\frac{1}{2}(z_r - z_o)^2 \tag{4}$$

2.3.2 Latent distillation

The BIR method, as described above in Sec 2.2 has no mechanism for preventing feature drift. Hence, we add a latent distillation loss which is similar to the feature distillation [10] however performed on the latent space level, similar as in [7]. During the training of task t, we use the previously trained model consisting of encoder E_{t-1} and decoder D_{t-1}. We use additional loss between the latent vector obtained by passing the sample through the previous model encoder z_{t-1} and the latent vector produced by the current training model encoder z_t. The calculation of difference coincides with the calculation of latent matching loss defined before but with different inputs given:

$$L^{\text{latent distill}}(z_{t-1}; \phi_{t-1,t}, \psi_{t-1,t}) = -\frac{1}{2}(z_t - z_{t-1})^2 \quad (5)$$

2.3.3 Cycling

Even with the proposed changes, we hypothesized that there might be a large distance between generated latents and original ones. Using the motivation from [2] we decided to pass the generations during training several times through the previous model, to make the generations closer to reconstruction. In the mentioned paper, authors use a similar approach in a context of *classifier cards* that capture and reconstruct knowledge from previous tasks better. In our case, we do not use the memory buffer to save any data and generate the features using the previous model. To verify our assumption we measure the distance between original features and generated ones we compute the Fréchet distance [3], which measures the distance between two Gaussian distributions. Figure 2 shows how the Fréchet distance is reduced between generated latents and original ones as we use cycling. This motivates us to incorporate it during training.

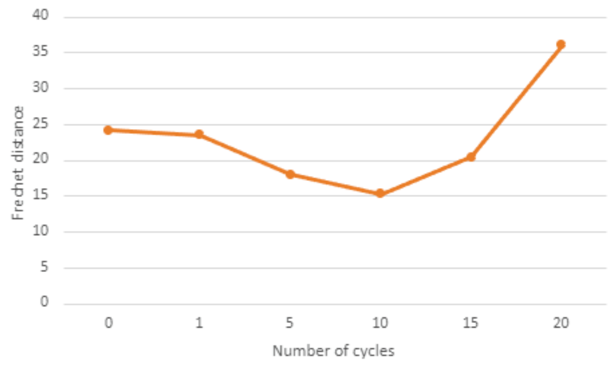

Figure 2: Fréchet distance between original and generated latents as a function of a number of cycles. 0 stands for the standard model (no cycling). As we increase the number of cycles (up to some point) the generated latent vectors match more closely those from original data.

In our improved version of the baseline VAE method (BIR) we combine all of the described components into a single training objective function for the class-incremental learning session. It consists of two main parts namely L^{current} and L^{replay}. L^{current} is the loss that is calculated for the data of the current task, and it is given by:

$$L^{\text{current}} = L^G + L^C + L^{\text{latent match}} \quad (6)$$

L^{replay} is calculated for the generations as follows:

$$L^{\text{replay}} = L^G + L^D + L^{\text{latent distill}} \quad (7)$$

The final loss function is the combination of these two losses:

$$L^{\text{total}} = L^{\text{current}} + L^{\text{replay}} \quad (8)$$

We use this loss to train the encoder, decoder, and classifier with current task data and data from the generative feature replay, additionally aligned with cycling through VAE. For the final loss, we start with a simple version without using any additional tradeoffs (coefficients) to balance each component.

3. Experimental setup

We utilize PyTorch as our framework [9]. We pretrain ResNet-32 as the feature extractor on the first 50 classes of the CIFAR-100 after randomly shuffling the data. For the pretraining stage, we use strong data augmentations from the PyCIL framework [13], which improves the performance of generative replay methods. In incremental steps, when we use an already pretrained feature extractor, we change data augmentation to one introducing less distortions to the inputs: images are firstly padded by 4 and then are randomly cropped to have size 32×32. In addition, we use random horizontal flips for augmentation.

4. Results and Analysis

We performed the experiments on CIFAR-100 with the first task containing 50 classes. The rest 50 classes were split equally into 5, 10, and 25 tasks. To evaluate the overall performance we calculate average incremental accuracy over all tasks. It is obtained by taking the average of accuracies after each task.

The average incremental accuracies are shown in Table 1, and the accuracies after each task for T = 5, 10, 25 are shown in the form of plots in Figure 3. Our method outperforms the regularization methods, and also the baseline BIR method. BIR combine with the SI is better than the other methods for six tasks, however, it falls behind if the number of tasks is increasing. In our method, we change the regularization methods from SI to distillation-based for latent alignment. By that, we see systematic improvements in all scenarios.

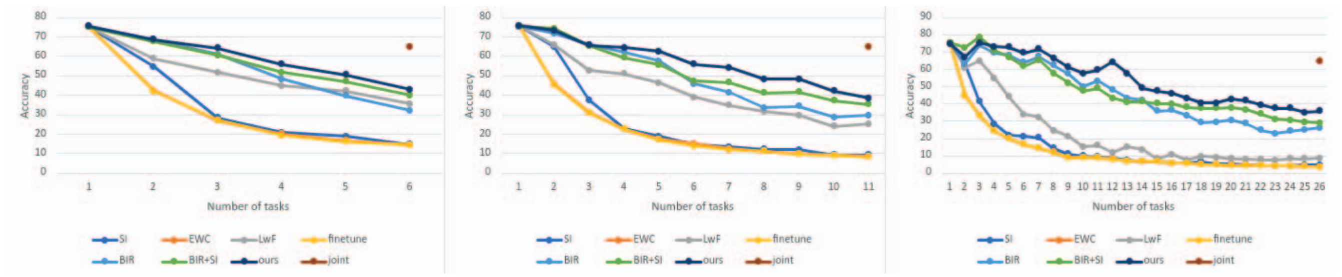

Figure 3: Comparison of average accuracies after each task for 6, 11, and 26 tasks with the first task containing 50 classes.

Table 1: The average incremental accuracies on CIFAR 100 with the first task containing 50 classes and the rest 50 classes split into 5, 10, and 25 tasks equally.

CIL Method	**T=6**	**T=11**	**T=26**
SI	35.46	26.42	15.42
EWC	32.66	23.56	13.37
LwF	51.38	43.31	22.56
BIR	54.22	51.93	45.59
BIR+SI	57.02	52.39	47.95
Finetune	32.46	23.3	13.33
Ours	**59.78**	**57.2**	**53.62**
Joint		64.7	

4.1. Number of cycles

We perform the analysis of how the number of cycles influences the average incremental accuracy for $T = 6$. As Figure 4 shows the accuracy firstly drops but with an increased number of cycles the performance improves by a significant margin. The number of cycles should be treated as a hyperparameter and tuned for different datasets and split scenarios.

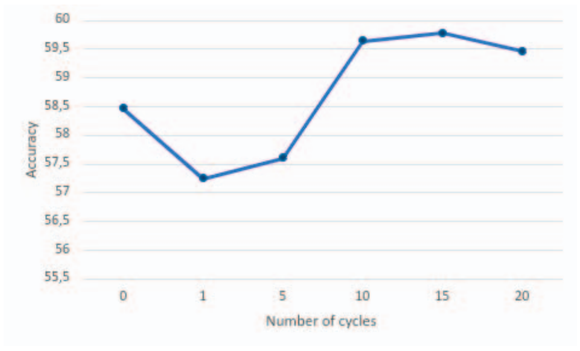

Figure 4: Average incremental accuracy of the model depending on number of cycles for *T=6*.

Table 2: Ablation study of our method for CIL setting with T=6 and CIFAR-100. Avg. inc. accuracy is reported for ResNet32.

Approach	Latent match	Latent distillation	10 cycles	Acc.(%)
baseline method - BIR				54.22
w/ latent match	✓			56.21
w/ latent distillation	✓	✓		58.46
w/ 10 cycles	✓	✓	✓	59.78

4.2. Ablation study

We perform an ablation study of our method. By starting from the baseline model (BIR), we add one by one the modifications that we propose. The results of the ablations study are presented in Table 2. As can be seen, all the elements of our method contribute significantly to the overall performance, where in total we reach 5.56% of average incremental accuracy in comparison to BIR.

5. Conclusions and Future Work

In this work, we propose a set of improvements for generative replay in class incremental learning. We observe that the currently used approach for feature-level replay suffers from the mismatch of latent vectors between original and regenerated samples. Based on that we add a loss function that aligns the latent vectors together. On top of that, we have proposed a cycling procedure, which passes the generated features through the model several times, before being used in the training. Through, the ablation study we have shown the improvements coming from each of the introduced components.

For future work, we aim to scale the proposed solution to more challenging datasets, such as ImageNet, and longer sequences of more diversified tasks. Another interesting direction is to prepare VAE-based feature replay models for task-free scenarios in CIL.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in Neural Information Processing Systems, 2014.

[2] Saisubramaniam Gopalakrishnan, Pranshu Ranjan Singh, Haytham Fayek, Savitha Ramasamy, and Arulmurugan Ambikapathi, *Knowledge capture and replay for continual learning*, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022, pp. 10–18.

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, NeurIPS, 2017.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean, *Distilling the knowledge in a neural network*, NIPS Deep Learning and Representation Learning Workshop, 2015.

[5] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat, *Generative Models from the perspective of Continual Learning*, IJCNN, 2019.

[6] Zhizhong Li and Derek Hoiem, *Learning without forgetting*, Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV, Lecture Notes in Computer Science, vol. 9908, 2016, pp. 614–629.

[7] 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*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 226–227.

[8] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer, *Class-incremental learning: Survey and performance evaluation on image classification*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022), 1–20.

[9] 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*, (2017).

[10] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, *Fitnets: Hints for thin deep nets*, International Conference on Learning Representations (ICLR) (2015).

[11] 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** (2020), no. 1, 4069.

[12] Friedemann Zenke, Ben Poole, and Surya Ganguli, *Continual learning through synaptic intelligence*, International Conference on Machine Learning, ICML 2017, 2017.

[13] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan, *Pycil: a python toolbox for class-incremental learning*, SCIENCE CHINA Information Sciences **66** (2023), no. 9, 197101–.