Exemplar-Supported Generative Reproduction for Class Incremental Learning

Chen He[1,2]
chen.he@vipl.ict.ac.cn
Ruiping Wang[1,2]
wangruiping@ict.ac.cn
Shiguang Shan[1,2]
sgshan@ict.ac.cn
Xilin Chen[1,2]
xlchen@ict.ac.cn

[1] Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS,
Beijing, 100190, China

[2] University of Chinese Academy of Sciences,
Beijing, 100049, China

Abstract

Incremental learning with deep neural networks often suffers from catastrophic forgetting, where newly learned patterns may completely erase the previous knowledge. A remedy is to review the old data (i.e. rehearsal) occasionally like humans to prevent forgetting. While recent approaches focus on storing historical data or the generator of old classes for rehearsal, we argue that they cannot fully and reliably represent old classes. In this paper, we propose a novel class incremental learning method called *Exemplar-Supported Generative Reproduction (ESGR)* that can better reconstruct memory of old classes and mitigate catastrophic forgetting. Specifically, we use Generative Adversarial Networks (GANs) to model the underlying distributions of old classes and select additional real exemplars as anchors to support the learned distribution. When learning from new class samples, synthesized data generated by GANs and real exemplars stored in the memory for old classes can be jointly reviewed to mitigate catastrophic forgetting. By conducting experiments on CIFAR-100 and ImageNet-Dogs, we prove that our method has superior performance against state-of-the-arts.

1 Introduction

We humans live in an ever-changing environment where learning takes place throughout our life. According to [33], the ability to learn incrementally is one of the six lessons we should learn from babies when developing intelligent agents. In machine learning, however, incremental learning is highly underrated—we usually focus on non-incremental tasks where the whole training set is fixed from the beginning. That is not the case in real-world applications such as intelligent robots where the environment is dynamic and new objects appear continuously, thus we call attention back to this area. Incremental learning assumes that data come continuously and new knowledge needs to be incorporated into the existing model over time.

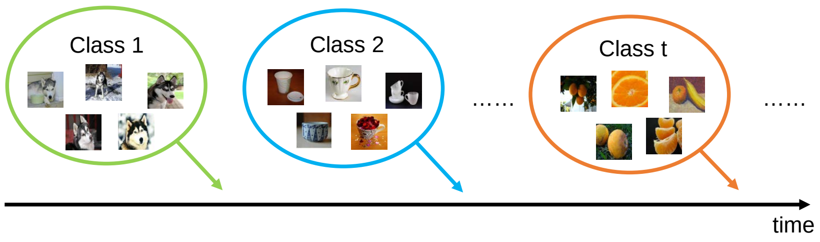


Figure 1: For real-world applications like robots or self-driving cars, the environment is non-static and new objects appear continuously. Class incremental learning, a simplified version of what we encounter in the real world, assumes that samples of one new class or a batch of new classes arrive at a time.

In conventional incremental learning, each incoming sample may come from an arbitrary class. In this paper, we focus on a simplified version called class incremental learning [27] where samples from one new class or a batch of new classes arrive at a time (Figure 1).

A notorious phenomenon that often co-occurs with incremental learning is catastrophic forgetting, where newly learned patterns can completely disrupt or erase the previous knowledge. No matter how well the model is trained on new tasks, the overall performance can be largely pulled down due to the forgetting of old knowledge. While recent works tried to alleviate catastrophic forgetting by storing old data [27] or the generator of old classes [32], we argue that neither is optimal. Old data are reliable, but they are rare and lack generalization ability; the generator can offer variety, but it is hard to train well, making the generated data less reliable. Thus, it is better to use a fusion of them to depict the real distributions of old classes. To this end, we propose a novel class incremental learning method called Exemplar-Supported Generative Reproduction (ESGR) that can better reconstruct memory of old classes (i.e. memory reproduction) and mitigate catastrophic forgetting. Specifically, we use the most outstanding generative models in recent years—Generative Adversarial Networks (GANs)—to model the underlying distributions of old classes and select additional real exemplars as anchors to support the learned distributions (Figure 2). Synthesized data generated by GANs and real exemplars stored in the memory for old classes can be jointly reviewed to alleviate catastrophic forgetting when adding samples of new classes. Experiments on CIFAR-100 and a subset of ImageNet containing all sub-categories of dogs (named ImageNet-Dogs) show that our method has a better performance against state-of-the-arts.

2 Related Works

Incremental learning. Incremental learning requires that the algorithm learns from streaming data with limited memory without sacrificing accuracy. The main challenge is the stability-plasticity dilemma [9]. Stability is to prevent the model from forgetting old knowledge, whereas plasticity is to quickly integrate new knowledge. It is difficult to have them simultaneously, and all incremental learning algorithms basically focus on how to strike a balance between them. Before deep learning era, there are various incremental learning approaches like incremental SVMs [6], ensemble methods [25, 28], distance-based methods [21] and so on.

Mitigating catastrophic forgetting. Catastrophic forgetting, an expression of the excessive plasticity that usually appears in neural networks, was first documented in [20]. According to the ways of using historical data to train the new classifier, existing methods can be classified into three main categories: rehearsal, pseudo-rehearsal, and non-rehearsal. **Re-**

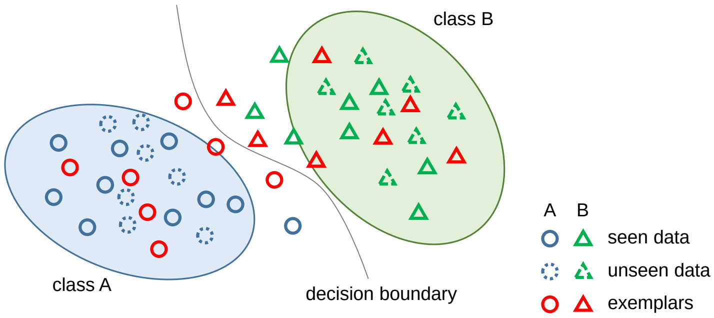


Figure 2: Intuitive illustration of Exemplar-Supported Generative Reproduction. We not only preserve learned class distributions (large ovals in the figure) but also typical samples as anchors (exemplars). It can better reconstruct memory of old classes.

hearsal algorithms explicitly review the old class samples. Joint training ([19] first uses this name) is a typical method of this kind. It is not an original creation by anyone else (if anything, say [5]), but usually seen as an upper bound of all other class incremental learning methods. The idea is to simply store all historical data and retrain the model whenever adding new classes. **Pseudo-rehearsal** methods implicitly review the old class samples. The original pseudo-rehearsal method [29] generates random inputs and gets their outputs using the existing model. These input-output pairs (pseudo-patterns) mixed with new data are used to retrain the model. Learning without forgetting (LwF) [19] can be seen as a special case of the original pseudo-rehearsal method. The prototype-based class incremental learning method iCaRL [27] is based on LwF as well. **Non-rehearsal** methods store no previous data. Practical approaches include dual-memory models [2, 12], evolving networks [18, 31], weight consolidation [15], and other variants of neural networks [4, 23].

Generative adversarial networks. Generative adversarial networks (GANs) are the most outstanding generative models for image generation in recent years. Besides generating high-quality images [8, 10, 22, 24], GANs have been used in image classification [34], zero-shot learning [3, 35] etc. Deep generative replay [32] also employs the GAN to model the cumulative distribution of seen classes; its differences from our method will be illustrated in detail in Section 3.3. Instead of generating raw images, there are other works that generate features [3, 35], but it is hard to judge whether the generated feature really corresponds to a sample of the given class or not. That is why we insist on using GANs to generate raw images instead of features.

3 Method

3.1 Problem Formulation

Class incremental learning assumes that samples from one class or a batch of classes arrive at a time. To better illustrate our algorithm, we simply assume that exactly one class is added at a time (one-class adding) throughout Section 3. The idea is basically the same when adding a batch of classes at a time (batch-of-class adding).

Supposing that there are C classes in order. The whole training set X_{real} can be denoted $\{X_{real}^{(1)}, \dots, X_{real}^{(C)}\}$ where $X_{real}^{(t)}$ is the training set for class $t (t \in \{1, \dots, C\})$. A “real” subscript is added here to distinguish it from the generated samples $X_{gen}^{(t)}$ and the real exemplars $X_{exem}^{(t)}$ for class t to be mentioned later. A training session is defined as the time from the arrival of the current class to that of the next class. Class incremental learning requires that the algorithm gives competitive accuracy on the test set of seen classes for each training session.

3.2 Exemplar-Supported Generative Reproduction

The idea of our method is to create a balanced training set that covers all seen classes (old and new) and use it to train the classifier for each training session. It is in essence converting a sequential learning problem to a concurrent one.

For each training session, our algorithm does the following steps (Figure 3): training generators, training classifiers, selecting exemplars. We take the t-th training session for example and elaborate on those steps respectively.

Training generators. When $X_{real}^{(t)}$ arrives, an independent generator G_t is trained from scratch using $X_{real}^{(t)}$.

Training classifiers. A balanced training set for all seen classes is prepared before training the classifier. For the new class t, since the real data $X_{real}^{(t)}$ is available at present, we use them all. For the old class $i (1 \leq i < t)$, we first take out the real exemplars $X_{exem}^{(i)}$ stored in the memory. By mixing it with $X_{gen}^{(i)}$ generated by G_i, we get $X_{mix}^{(i)}$ as the training samples for class i. The number of the samples in $X_{mix}^{(i)}$ should be the same as that in $X_{real}^{(t)}$. Accordingly, we get $\{X_{mix}^{(1)}, \ldots, X_{mix}^{(t-1)}, X_{real}^{(t)}\}$ as a balanced training set to train classifier t. Notice that when $t = 1$, there is only one seen class and we don't need to train a classifier.

Selecting exemplars. After the classifier for class t is well trained, we use $X_{real}^{(t)}$ as inputs and get their response vectors using the classifier. For simplicity, we denote $f(x)$ as the softmax output of the network, indicating the probabilities of the sample x belonging to each class. For each $x \in X_{real}^{(t)}$, we use the t-th element in $f(x)$ as its score for class t. The higher the score, the more confident it is a sample from class t. We sort the scores and choose the top-K samples as exemplars by default. In Section 4.5, we compare other selection strategies like random-K and bottom-K. Note that K is dynamic, since we use a similar memory mechanism introduced by iCaRL [27]. The memory has a fixed capacity M, and the number of exemplars stored for each class is the memory capacity size divided by the number of seen classes $\frac{M}{t}$. For old classes which have more exemplars stored in the memory, we simply keep the top-$\frac{M}{t}$ samples and remove the others to make room for exemplars of the new class.

For simplicity, we use ESGR-*mix* to denote the method that use both generated data and real exemplars for old classes. According to the composition of the training set, our method has two variants—ESGR-*gens* and ESGR-*reals*. ESGR-*gens* only uses generated data for old classes, so the corresponding training set is $\{X_{gen}^{(1)}, \ldots, X_{gen}^{(t-1)}, X_{real}^{(t)}\}$, whereas ESGR-*reals* only uses the real exemplars for old classes, so the training set is $\{X_{exem}^{(1)}, \ldots, X_{exem}^{(t-1)}, X_{real}^{(t)}\}$. Note that $X_{exem}^{(i)}$ usually has fewer samples than $X_{real}^{(t)}$. To get a balanced training set for each class, we simply sample $X_{exem}^{(i)}$ multiple times, making it the same size as that of $X_{real}^{(t)}$.

3.3 Differences from Related Methods

Among all the class incremental learning methods, iCaRL [27] and Deep generative replay [32] are the most related to ours.

iCaRL. iCaRL and our method both employ a memory mechanism to store exemplars, but the criteria for selecting exemplars are totally different. iCaRL is a prototype-based

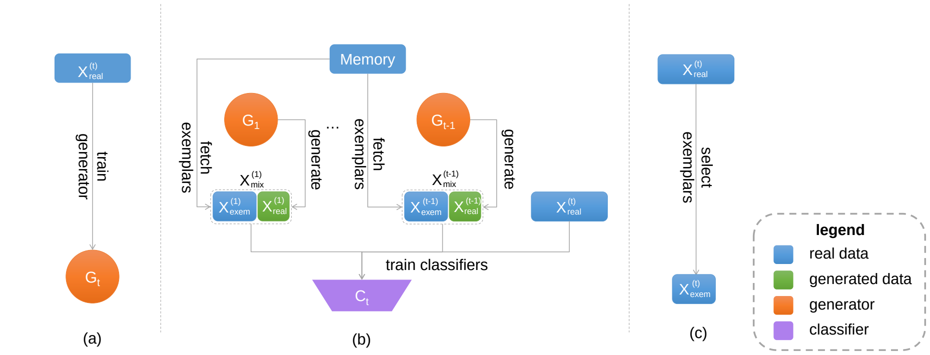


Figure 3: The flow diagram of our algorithm in the t-th training session. For each training session, our algorithm does the following steps: a) Training generator. b) Training classifier. c) Selecting exemplars

method, so the criterion for exemplar selection is to let the mean of the exemplars approximate the class mean as close as possible in the feature space. Whereas we take an end-to-end deep learning approach to directly classify samples using the softmax classifier. Thus, we select the most discriminative samples as exemplars. In Section 4.5, we perform comparisons of different exemplar selection strategies.

Deep generative replay (DGR). DGR and our method both use GANs to "replay" old data. The differences are: a) DGR uses only one generator to model the cumulative distribution of seen classes; we use multiple generators because the capacity for one generator is not enough for difficult datasets. b) The training of the new generator in DGR depends on data generated by the old generator. There might be information loss inside the knowledge transfer and it suffer from a build-up of errors. In contrast, our method uses the real data only. c) DGR uses an unsupervised generator for all seen classes. It cannot generate samples for specific classes. To attach labels to the generated samples, it uses an extra classifier; we use auxiliary conditional GANs and we can generate samples for a given class directly.

4 Experiment and Results

4.1 Experiment Setups

Datasets. We evaluate different class incremental learning algorithms on CIFAR-100 [16] and ImageNet-Dogs. ImageNet-Dogs is based on ILSVRC2012 [30]. We down-sampled it to 64*64 according to [7] and only choose the 120 dogs, which are the same as the classes in Stanford Dogs [14]. The reason for down-sampling and using 120 dogs only is that running on the original version of ILSVRC2012 is very time-consuming. But compared to [32], our datasets are more challenging.

Evaluation protocol. The mean accuracy on seen classes is reported as the performance measure. For each training session, we calculate a mean accuracy value. By connecting these values of different training sessions, we can get an accuracy curve for each method. The higher the accuracy curve, the better the method is. Since different class orders may lead to slightly different accuracy curves, for the small dataset CIFAR-100, we use 3 different class orders and report the mean accuracy curve which also shows the standard deviation in the figure.

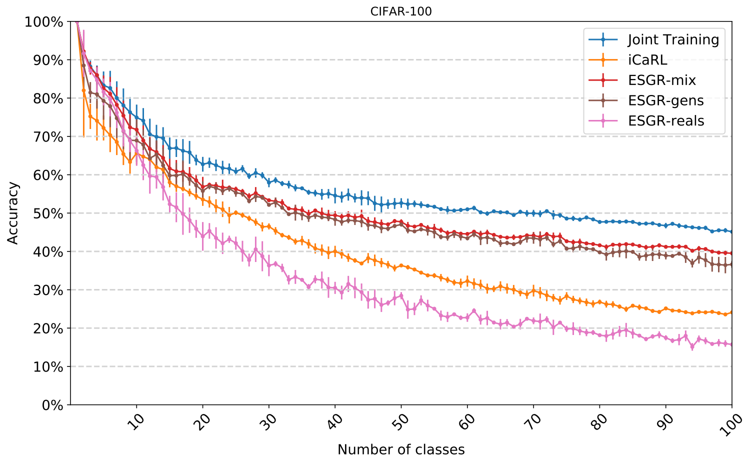


Figure 4: Accuracy curves of different methods on CIFAR-100 adding 1 class at a time. Joint training is the upper bound. ESGR-*mix* and ESGR-*gens* are our proposed methods and ESGR-*reals* is for ablation study (Section 4.3). We only compare iCaRL here because other methods are intrinsically not suitable for the one-class adding setting.

Compared methods. We compare joint training (upper bound), iCaRL [27], Learning without Forgetting (LwF) [19], Deep generative replay (DGR) [32] and our methods—ESGR-*mix*, ESGR-*gens*, ESGR-*reals*.

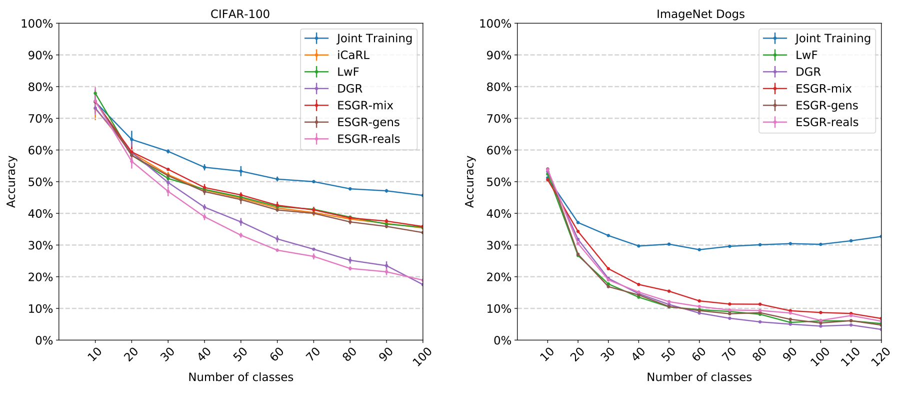
Implementation. The code is implemented in Tensorflow [1][1]. For CIFAR-100, we use LeNet [17] as the classifier and WGAN with gradient penalty [10] as the generator for each class. Each training session consists of 70 epochs. The base learning rate starts at 0.2 and it is divided by 5 after 49 and 63 epochs inspired by iCaRL. For ImageNet-Dogs, a small ResNet [11] with only 4 residual blocks is used as the classifier and AC-GAN [24] is used as the generator. Each training session consists of 60 epochs. The base learning rate starts at 0.2 and it is divided by 5 after 20, 30, 40 and 50 epochs. For both datasets, we use a weight decay of 0.00001 and a mini-batch size of 128. For optimization, we use stochastic gradient descent with a momentum of 0.9. As for implementation of other methods, basically we use the same network architecture and almost the same hyper-parameters to make fair comparisons. Detailed settings of different methods are further explained in the supplementary material.

4.2 Overall Performance

We did experiments using one-class adding on CIFAR-100 and 10-class adding for both CIFAR-100 and ImageNet-Dogs. The results are shown in Figure 4 and Figure 5 respectively. Generally, ESGR-*mix* has a better performance than others especially under the one-class adding setting on CIFAR-100. ESGR-*reals* is for ablation study only.

iCaRL is much lower than ESGR-*mix* under the one-class adding setting on CIFAR-100 (Figure 4). The reason might be that iCaRL is based on LwF for representation learning and LwF is intrinsically not suitable for one-class adding setting. ESGR-*mix*/*gens* learns generators for each class independently, thus doesn't have such a problem. However, when changing to 10-class adding setting (Figure 5(a)), iCaRL has a huge improvement and ESGR-*mix*/*gens* indeed drops a little bit. The reason for the improvement of iCaRL is that adding more classes at a time is generally more advantageous to all incremental learning algorithms. An extreme case is to add 100 classes, which is identical to training a conventional 100-class

[1]Our source code is available at http://vipl.ict.ac.cn/resources/codes.



(a) (b)

Figure 5: Accuracy curves of different methods on (a) CIFAR-100 and (b) ImageNet-Dogs adding 10 class at a time

classifier directly and it will give the best performance. The abnormal effect of ESGR-*mix*/*gens* may be connected with the composition of our training set. Although we try to create a balanced training set, it is in essence "imbalanced". Some of the generated data may be blurred or distorted, making it hard to judge which class they belong to (i.e. noisy data). These samples make less contribution to the adjustment of the decision boundaries, making the whole training set bias newly added classes and may pull down the overall performance a little bit. A possible solution is to use generated data for the new class as well. Preliminary experiments show that the performance can improve 1-2% in the final evaluation (more details in the supplementary material).

Using the same dataset under the 10-class adding setting (Figure 5(a)), ESGR-*mix*/*gens*, LwF, iCaRL perform almost the same and ESGR-*mix* is slightly higher. DGR performs less well; the reason might be that it suffers from a build-up of errors when facing a sequence of tasks, since it uses predicted labels of the previous classifier as ground-truth labels when training the new classifier.

On ImageNet-Dogs (Figure 5(b)), all methods don't perform so well because it is, in fact, a difficult fine-grained dataset. Among them, ESGR-*mix* is obviously much higher. iCaRL is not showed in the figure because it is not higher than LwF. It is an undesirable result since iCaRL is based on LwF. A possible reason would be that iCaRL is very sensitive to the dimension of features extracted by the network. Our network outputs a lower dimension of features and hence not suitable for iCaRL. From another perspective, it reflects one drawback of iCaRL that the representation learning part does't necessarily benefit the final prototype-based classification. There is a gap between the two objectives.

4.3 Ablation Study

In Figure 4 and 5, it can be seen that the performance of using a mix of generated data and real exemplars is better than using generated data or real exemplars only. The reason is that the generated data has large diversity, but lacks details. In contrast, the real exemplars are rare, but clear in the details which may contain discriminative parts that avail classification. On CIFAR-100, ESGR-*gens* outperforms ESGR-*reals* because CIFAR-100 is an easy dataset and GANs can be easily trained to generate satisfactory pictures. But for ImageNet-Dogs, the pictures have higher resolution and more background clutter, making GANs difficult to

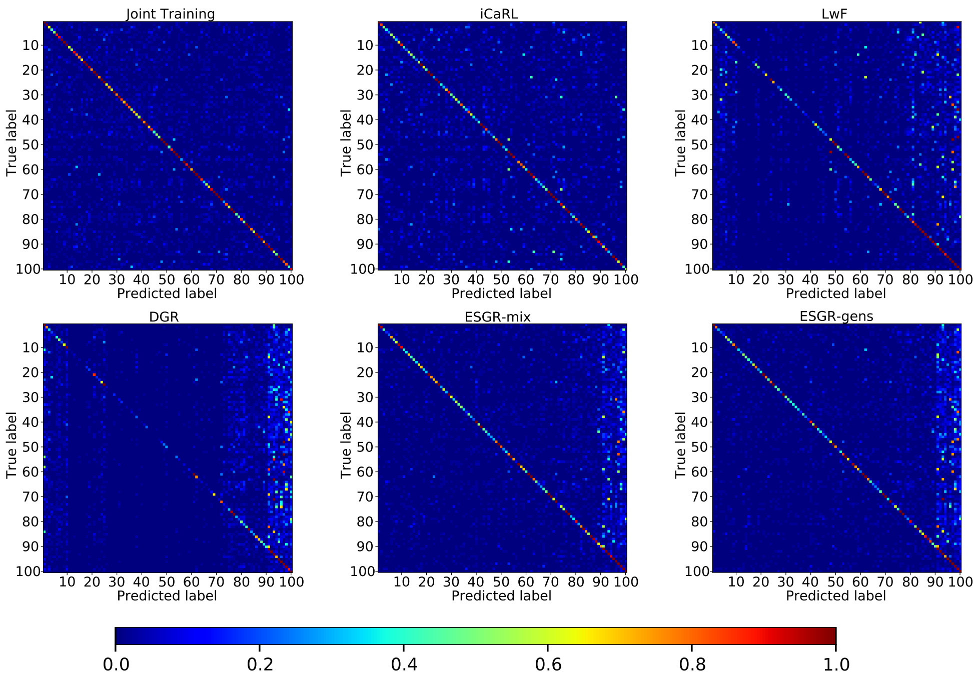


Figure 6: Visualization of the confusion matrices of different methods in the final evaluation of the 10-class adding setting on CIFAR-100. The element in the i-th row and j-th column indicates the percentage of samples with ground-truth label i that are classified into class j. The diagonal elements are the accuracies of each class. All of those confusion matrices share the same color bar which is shown below.

learn well. Thus, ESGR-*reals* performs better than ESGR-*gens* (Figure 5(b)).

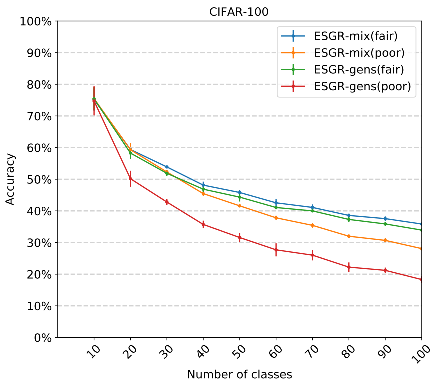
4.4 Detailed Analysis

To further analyze the properties of different methods, we first visualize their confusion matrices in the final evaluation under the 10-class adding setting on CIFAR-100 (Figure 6). An obvious phenomenon is that ESGR-*mix*/*gens* and DGR has a distinct bar in the right of the confusion matrix, implying that it is more inclined to predict a sample to come from the last 10 classes (newly added classes), whereas the problems in iCaRL and LwF are not obvious, especially iCaRL. The reason is again because our training set is in essence “imbalanced”. DGR has the same problem because it also uses generated data for old classes and real data for new classes to train the classifier as we do.

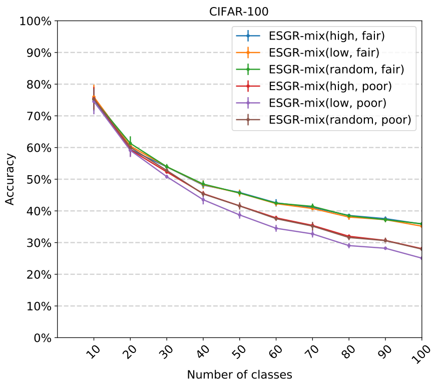
To analyze the accuracy change in the temporal axis, we introduce *final score*, *adaption score* and *forget rate* inspired by [13]. Let $Acc_t^{(i)} (i \leq t)$ denote the accuracy of class i in the t-th training session. The *final score* is the mean accuracy of all classes in the final evaluation i.e. $\frac{1}{C}\sum_{i=1}^{C} Acc_C^{(i)}$. The *adaptation score* is the mean accuracy of all classes immediately after they’re added i.e. $\frac{1}{C}\sum_{i=1}^{C} Acc_i^{(i)}$. The *forget rate* is the difference between *adaption score* and *final score* over the *adaptation score* i.e. $1-\frac{\sum_{i=1}^{C} Acc_C^{(i)}}{\sum_{i=1}^{C} Acc_i^{(i)}}$. Intuitively, *adaptation score* can measure the plasticity and *forget rate* can measure the stability of the algorithm. We calculate these performance measures under the 10-class adding setting on CIFAR-100.

Table 1: Results of different methods under the 10-class adding setting on CIFAR-100

Method	Final	Adaptation	Forget Rate
Joint training	0.4611	0.5437	15.19%
iCaRL [27]	0.356	0.4865	26.82%
LwF [19]	0.311	0.6357	51.08%
DGR [32]	0.1791	0.6024	70.27%
ESGR-*mix*	0.3576	0.6243	42.72%
ESGR-*gens*	0.3388	0.6393	47.00%



(a)



(b)

Figure 7: Accuracy curves for changing (a) the performance of the generators and (b) the exemplars selection strategies.

From Table 1, we can see that although ESGR-*mix* and iCaRL have nearly the same *final score*, they display totally different properties. ESGR-*mix* has higher *adaptation score* and *forget rate*, whereas iCaRL has lower *adaptation score* and *forget rate*. Another interesting fact is that although iCaRL is based on LwF, their results of these measures vary considerably. The reason might be again due to the composition of the training set mentioned above. ESGR-*mix* and LwF are prone to biasing newly added classes but iCaRL is not. We argue that the forgetting in our method is intrinsically a problem of imbalanced data. By converting ESGR-*gens* into a balanced version, the *forget rate* can decrease to about 22% (lower than iCaRL), preliminary results are shown in the supplementary material.

4.5 Design Evaluation

We tried many different hyper-parameters of our method, among which the most important ones are the performance of the generators and the exemplar selection strategy. The following experiments are done under the 10-class adding setting on CIFAR-100.

Performance of the generators. We use a fair generator and a poor generator. They have the same network architecture and they are trained for the same number of iterations. The only difference is the base learning rate of Adam optimizer. The higher one (1e-3) performs better than the lower one (1e-4). Although the difference is subtle as we see the generated images or measured by inception scores, the influence on the final accuracy curves is huge. From Figure 7(a) we can see that the performance of both ESGR-*gens* and ESGR-*mix* have a remarkable improvement by changing from a poor generator to a fair one.

Selection strategies of exemplars. We choose samples with top-K, random-K and bottom-K scores as exemplars according to the algorithm mentioned in Section 3. From Figure 7(b) we can see, for a fair generator, it is not sensitive to the choice of the exemplars and they perform almost the same. This is probably because the generator can already generate high-quality images and don't need real exemplars for augmentation any more. For a poor generator, top-K and random-K perform almost the same (top-K slightly better), but bottom-K is much lower. It indicates that choosing simple samples (top-K) instead of hard ones (bottom-K) will benefit the learning process.

5 Conclusion

In this paper, we propose a class incremental learning algorithm called *Exemplar-Supported Generative Reproduction (ESGR)* that leverages both generated data and real exemplars to mitigate catastrophic forgetting. Generally, ESGR has superior performance over others on the proposed benchmarks especially under the one-class adding setting. Besides, we perform a comprehensive analysis of related class incremental learning methods and show their strengths and limitations. Although there are limitations in our framework such as the additional training time and memory cost since it involves an extra generator training step, we think it is a necessary step to explore what should be left as old knowledge in incremental learning. Further works would be integrating generator and classifier to make it end-to-end, applying the idea of evolving network into our method etc.

Acknowledgements

This work is partially supported by Natural Science Foundation of China under contracts Nos. 61390511, 61772500, 973 Program under contract No. 2015CB351802, Frontier Science Key Research Project CAS No. QYZDJ-SSW-JSC009, and Youth Innovation Promotion Association CAS No. 2015085.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Bernard Ans and Stéphane Rousset. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 320(12):989–997, 1997.

[3] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Generating visual representations for zero-shot classification. In *International Conference on Computer Vision (ICCV) Workshops: TASK-CV: Transferring and Adapting Source Knowledge in Computer Vision*, 2017.

[4] Gail A Carpenter and Stephen Grossberg. Art 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, 1987.

[5] Rich Caruana. Multitask learning. In *Learning to Learn*, pages 95–133. Springer, 1998.

[6] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2001.

[7] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[9] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[13] Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *arXiv preprint arXiv:1708.02072*, 2017.

[14] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

[18] Jeongtae Lee, Jaehong Yun, Sungju Hwang, and Eunho Yang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[19] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[20] 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.

[21] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[23] Jacob MJ Murre. *Learning and Categorization in Modular Neural Networks*. Psychology Press, 2014.

[24] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[25] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001.

[26] Amal Rannen Triki, Rahaf Aljundi, Matthew Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[27] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5533–5542, 2017.

[28] Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of random forests for large-scale image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):490–503, 2016.

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

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

[31] 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.

[32] 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.

[33] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11(1-2):13–29, 2005.

[34] William Wang, Angelina Wang, Aviv Tamar, Xi Chen, and Pieter Abbeel. Safer classification by synthesis. *arXiv preprint arXiv:1711.08534*, 2017.

[35] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. *arXiv preprint arXiv:1712.00981*, 2017.