An Incremental Learning Network Model Based on Random Sample Distribution Fitting

Wencong Wang[1], Lan Huang[1,2], Hao Liu[1], Jia Zeng[1], Shiqi Sun[1], Kainuo Li[3], and Kangping Wang[1,2(✉)]

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China
wangkp@jlu.edu.cn

[2] Key Laboratory of Symbolic Computation and Knowledge Engineering, Jilin University, Changchun 130012, China

[3] Information Center of Political and Law Committee of Jilin Provincial Committee, Changchun 130001, China

Abstract. The training of the classification network has tough requirements of data distribution. The more the training data did not consistent with the distribution of the real target function, the higher error rate the network will produce. In the context of incremental learning, the data distribution of the subsequent training tasks may not consistent with the data distribution of the previous tasks. To handle this problem, lots of learning methods were introduced, most of these methods are complicated and heavy computing. In this paper, a novel method which is faster and simpler is proposed to uniform subsequent training data. Artificial training samples are produced from random inputs in current trained network. In subsequent task, these artificial samples were mixed with new real data as training data. The experiments with proper parameters show that new features from new real data can be learned as well as the old features are not forgot catastrophically.

Keywords: Incremental learning · Sample distribution fitting · Classification network

1 Introduction

1.1 Relevant Background

Incremental learning is a type of learning mode in which the system can incessantly obtain new knowledge from persistent training samples. In some cases, since the training data cannot be obtained in the same time or stored completely,

Supported by the Jilin Province Science and Technology Development Plan Project under Grant 20190201273JC, and by the Jilin Provincial Key Laboratory of Big Data Intelligent Computing.

© Springer Nature Switzerland AG 2020

the training process may be spilt into many batches, each batch is called as a task. Transfer learning [16] is often used to establish the transformation from different tasks in multi-task incremental learning. If the knowledge that task A and task B going to learn are similar, the model of task A can be used while training task B, we can also use a part or all of the parameters in the network of task A to initialize the training network of task B. The transfer info can be the sample data of the original training set [2], the parameters of the model, or the features that the network extracted [18]. In this way, we can build a process of learning tasks from A to B and the process of learning task B to C, and so on. Although transfer learning can handle some problems in multi-task incremental learning, there are still some problems left. One problem is we can only obtain the trained network model and network weights instead of training data of task A when training task B. If the data of task A and task B do not have the same distribution, the ability of solving task A will gradually decline in the process of training task B. This sharp decline in the performance of previous learning tasks when new knowledge was added is called "catastrophic forgetting" [3], which is exactly the key issue that incremental learning needs to handle.

1.2 Related Work

The main idea of incremental learning is retaining the relevant information of previous tasks while training new tasks. Since convolutional neural network do not have a good interpretability [20], it has been found that incremental learning becomes difficult after convolutional networks begin to be applied to machine learning. In order to deal with this problem, the concept of knowledge distillation is first proposed in [5], which information of the old network model is transmitted to the new network through the redefined softmax function, in this way the new network is updated without catastrophic forgetting. Exploiting the relevant ideas of knowledge distillation, the LwF (Learning without Forgetting) method was proposed in [7]. LwF works better on both new tasks and old tasks comparing with Feature Extraction and Finetune-FC. It only needs information from the new tasks, that is more flexible than Joint Training method. The new loss function of the update network is defined in [7], which is the sum of the distillation loss value from the old model and the loss value of the new model. Using this new loss function, [14] and [15] have applied incremental learning in the field of target detection and semantic segmentation. The knowledge from early learning can still be forgotten using the LwF method after multiple tasks of learning. In order to improve performance, the iCaRL (Incremental classifier and Representation Learning) [12] method is proposed, which uses a part of representative data feature of the old task along with the data of the new task. iCaRL needs the data of the old task, which may consume an extra amount of storage. It's obviously that a network's parameters consume less storage than a big training data set. Based on this idea, [13] used the GAN model to train an experience playback of the old tasks' data while training the new tasks. This method will train a GAN [4] network which generate data to simulate old tasks data. For a classification network, we can also train a CGAN network [9,19].

GAN is hard to train and the data that generate by GAN is very likely to be quite different distribution with the real data. In the other way, comparing the weight difference or gradient difference between tasks in the process of learning can also achieve good performance. By comparing the weight difference, an Elastic Weight Consolidation (EWC) method is proposed in [6], which defines the importance of each weight in the network affects the training results by Fisher matrix [10], which guides the network to update the unimportant weights of the network while training and minimally affecting the previous results. The methods of comparing gradient difference such as GEM (Gradient Episodic Memory) [8] and its improved version A-GEM [1] constrain the gradient update direction of the new tasks. This can avoid catastrophic forgetting caused by too much critical deviation from the old tasks' gradient.

2 Random Sample Fitting Distribution Model (RFD)

2.1 Network Structure

Our classification network contains convolutional and fully connected layers, which is modified from LeNet [17] includes five layers which are input layer, convolutional layer, activation layer, pooling layer, and fully connected layer. This structure is inspired by an incremental learning model which uses GAN network to assist the incremental [13]. In the referred model, the tasks are training both on the classification network and on the GAN network in same time. After training, the generator of the GAN network can generate samples with the same distribution as the training set, and the classification network can also classify the samples well. The subsequent tasks use samples generated by the GAN network and real samples to train a new classification network. The training time of the GAN network is very long and its discriminator network does not outperform our model. We try to erase the GAN network and prove that the GAN network is unnecessary for preventing forgetting. Our incremental learning network consists of two parts, the Frozen network and the Update network. Frozen network is not changing during a training task while Update network is training. In the start of next task, the new Frozen network is copied from Update network. While training in a new task, the supervise data are mixing from new real data and outputs of the Frozen network generated by random inputs. These data are flow into the Update network together for typical gradient descent. This proposed network architecture is shown in Fig. 1.

2.2 Formula Derivation

Adding noise to training samples is a way of data augmentation [11], so can all samples be able to represent network information? From the Frozen Network, lots of supervised samples can be generated that reflect the parameters of previous Update Network. It's not easy to ensure the distribution of random input is consistent with previous training data. But our goal is rather preventing forgetting than keeping consistency. So, input distribution is uniform and proportions

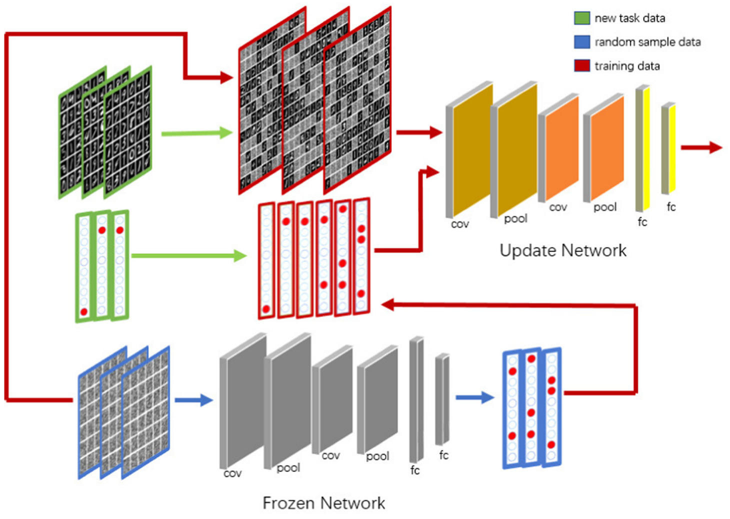

Fig. 1. A depiction of Random sample Fitting Distribution incremental learning network (RFD). Red color represents the training data composing by the new task data (green color) and the random sample data (blue color) generate by Frozen Network (grey color). These data finally flow into the Update Network (yellow color). (Color figure online)

of each category are same as previous training data. We hope to some extent that the random samples and their classification-generated by the old network can prevent forgetting effectively. Assuming that the previous training data is represented by D_{old}, the new training data is represented by D_{new} and the randomly generated data is represented by D_{random}. There are $|D_{old}|$ number of data in the previous training data set, $|D_{new}|$ in the new training data, and $|D_{random}|$ in the randomly generated data. Let n be the number of categories in all training data. The vector $P_{old} = (p_1, p_2, p_3, p_4, \cdots p_n)$ is used to represent the data classification of each category in the previous training data. A component p_a in $|P_{old}|$ represents the probability distribution of the old training data set on a category. It can be calculated that there are a total of $p_a \cdot |D_{old}|$ previous training data on a category. Similarly, the vector $P_{new} = (p'_1, p'_2, p'_3, p'_4, \cdots p'_n)$ is used to represent the data classification of each category of the new data. $P_{random} = (p''_1, p''_2, p''_3, p''_4, \cdots p''_n)$ represents the data classification of each category in generated data. The randomly generated data is generated by the Frozen network, which represents the data distribution of the previous training model, so we suppose $P_{random} = (p''_1, p''_2, p''_3, p''_4, \cdots p''_n) = (p_1, p_2, p_3, p_4, \cdots p_n)$. The distribution of the exact training data set in next tasks as follows.

$$\begin{aligned} P'_{new} =& \frac{p_1 \cdot |D_{old}| + p_1 \cdot |D_{random}| + p'_1 \cdot |D_{new}|}{|D_{old}| + |D_{random}| + |D_{new}|} + \frac{p_2 \cdot |D_{old}| + p_2 \cdot |D_{random}| + p'_2 \cdot |D_{new}|}{|D_{old}| + |D_{random}| + |D_{new}|} \\ &+ \cdots + \frac{p_n \cdot |D_{old}| + p_n \cdot |D_{random}| + p'_n \cdot |D_{new}|}{|D_{old}| + |D_{random}| + |D_{new}|} \end{aligned} \tag{1}$$

When the number of training data $|D_{new}| \ll |D_{random}|$, $P'_{new} = P_{old}$, which will not lead to catastrophic forgetting, but the network will never learn new information. If the number of new training data is $|D_{new}| \gg |D_{random}|$, $P'_{new} \neq P_{old}$, the network is prone to forgetting. Let $|D_{new}| = \phi|D_{random}|$, set ϕ to a larger number, and the network can be updated at a slower speed without catastrophic forgetting. In this way, the total Loss function L_{total} of the network can be defined as follows.

$$L_{total} = L_{new} + \phi L_{random} \tag{2}$$

The function L_{total} is derived from the distribution formula (see formula 1). When the network is updated, the loss value consists of two parts. The loss value generated by the random sample is recorded as L_{random}, and the loss value generated by the post-training data is recorded as L_{new}. The Frozen network has the frozen loss value of 0 does not participate in the update of the gradient. It can also be concluded from the above-mentioned distribution definition (see formula 2) that changes in ϕ will affect the update speed of the network. In neural networks, the direction of network updating is generated by the gradient of data. We can change the loss function formula (see formula 2) to a gradient formula (see formula 3) for a single update of the network. The total gradient G_{total} consists of gradient G_{new} from new data and gradient G_{random} of random generated data.

$$G_{total} = G_{new} + \phi G_{random} \tag{3}$$

In more extensive application scenarios, we can apply our model to different distribution data which have the same gradients. Keeping the random generated data to have the same gradient as the old data will have the same influence to network, which is the original intention of our model design.

3 Experiments

The verification experimental data in this paper are the MNIST handwritten dataset, which contains a training set and a test set. There are 60,000 samples in the training set and 10,000 samples in the validation set. Each sample is a 28×28 single-channel picture sequence and corresponding label.

3.1 Comparison with Non-incremental Learning Method

Failure Caused by Inconsistent Distribution of Training Data. When two tasks in incremental learning have obeyed different distributions, the training of the second task will affect the knowledge already learned in the first task. To verify this opinion, 800 samples in the data set are selected for the first task. These samples are uniformly distributed, and the probability of occurrence of 0–9 labels is 1/10. The process of training on LeNet is shown in Fig. 2. After 40 rounds of training, the accuracy of the MNIST test set reached 93%. As the number of trainings increases, the accuracy of the training set and val set gradually rises to 98% and 92%, and the loss value gradually reduced to 0.06 and 0.03, which is an acceptable value. There are another 800 samples with different distribution selected for second task. Among these samples, only the data labeled 0–4 appeared, and the probability of each type is 1/5. The data labeled 5–9 are not included, the probability is 0. Under this situation, the difference distribution will result in catastrophic forgetting. The training process for the second task is shown in Fig. 3. After 40 rounds of training, the training accuracy in the train set reached nearly 99%, and the loss value in the train set also dropped to a very low level. But after 17 epochs of training the accuracy of val set gradually decreases, and the loss value is not getting smaller, which indicates that the network training has been overfitted. Notice that the test set still contains 5 categories that were not included in training data for second task. After 40 epochs, the model has an accuracy of 50.77% on the test set of MINST. The model has forgotten lots of knowledge obtained from the first training when learning new knowledge. The remaining knowledge of the model is only the data labeled 0–4, and the data labeled 5–9 is forgotten, so the total accuracy is close to 50%.

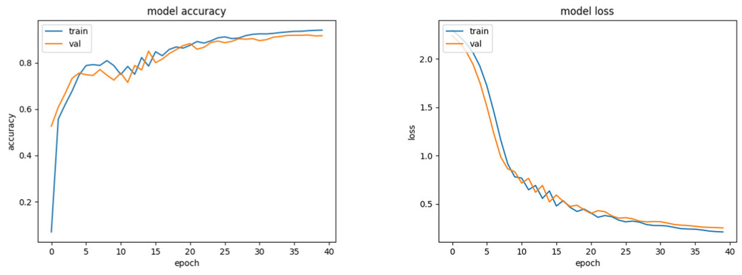

Fig. 2. The process of training in first task.

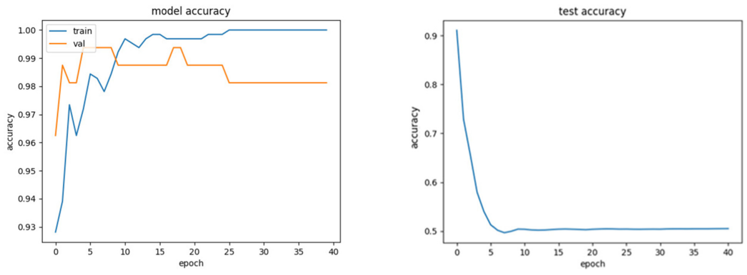

Fig. 3. Failure due to inconsistent distribution of training data, the training accuracy and training loss perform good during the second stage training, but the test loss and accuracy break up which means a catastrophic forgetting has happened.

Jitter Caused by Small Training Step. In some situation, the network can only get a small number of samples once, and the samples cannot be stored. In order to show this process, a network similar to the previous experiment (see Sect. 3.1) is trained, There are 10000 samples which are split into 100 tasks equally, and the epoch size for each task is 10. The total epochs size is 10000(100 $\times$ 10). This experiment is different with mini-batch method whose batch size is 10. The mini-batch method only performs one gradient descent, while 10 gradient descents are performed for the same 10 samples in this experiment. The training process is shown in Fig. 4. Because the whole training data have same distribution with test data, the accuracy increases when training data accumulate. But there are lots of jitters caused by multi-times training with a small part of training data which pull down final accuracy. In incremental learning, the training data were input like stream data which not available in next task. It's necessary to train multi times for new knowledge. There is contradiction between multi-times training and jitters avoiding. Some improvement should be introduced to handle this problem. The jitters which are representation of partly forgetting is harmful and should be eliminated as much as possible.

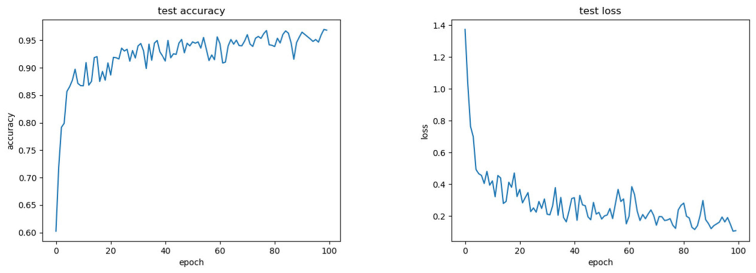

Fig. 4. Jitters caused by small training step. The jitter occurred not only on the training set but also on test set.

3.2 RFD Method

Suppressing Catastrophic Forgetting. First, the experiments show the inhibition of our method between two tasks which have different distributions, like the first situation of Sect. 3.1. Task 1 uses 800 10-label data to train to an accuracy of 93%, then Task 2 uses 800 5-label data for incremental learning. The results on the test set of MNIST are shown in Fig. 5. A significant difference can be seen after using our method in the training process of Task 2. In Task 2, the accuracy of a typical back propagation algorithm on the test set drops rapidly to less than 10% and the loss quickly increases. Exploiting our method, the accuracy only drops to 83%, and the speed of network forgetting is significantly slowed down.

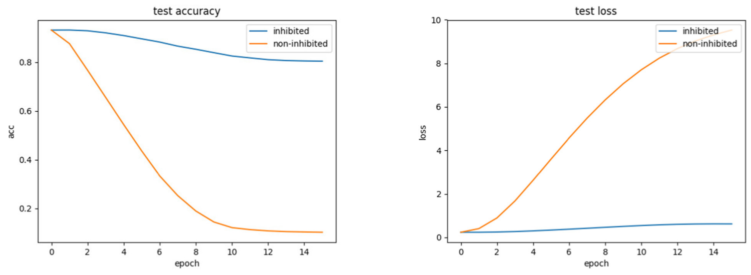

Fig. 5. Inhibition of forgetting by using model, the accuracy value drops slowly when using our model and the loss value rise only a little.

Continuous Learning with RFD. We have shown a harmful jitter in the second experiment of Sect. 3.1 which is mainly caused by partial forgetting. As a comparison, we made another experiment with same training dataset as the second experiment in Sect. 3.1. The results are shown in Fig. 6. Compared with Fig. 4, Fig. 6 has smoother lines, which is consistent with our expectation. RFD method can also suppress partial forgetting caused by too small step size in continuous learning.

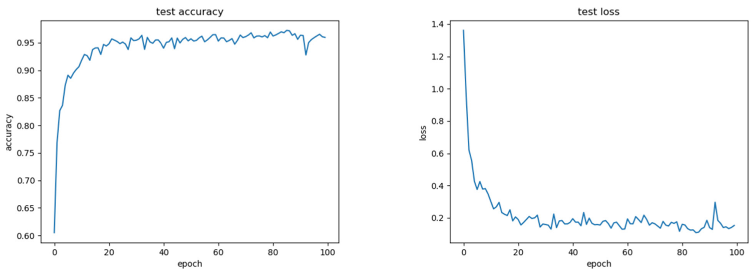

Fig. 6. Suppressing Jitter in Continuous learning. There are 100 data in a single task with 10 epochs for each task is performed. The ϕ in RFD method is set to 60.

3.3 Network Feature Verification

We want to make a comparison with non-incremental models with low learning rates to prove some characteristics of our model. Let the current network weights be W and the updated network weights is W', the learning rate is α, and the gradient is G, then the formula for a single gradient descent can be written as $W' = W + \alpha G$. Our Loss function is composed of two parts, L_{random} and L_{new}. At each gradient descent, Gradients come from both new task samples and random samples. Let the gradient provided by the new task samples be G_{new} and the gradient provided by the random samples be G_{random}. The gradient G

is a function of loss, it can be written as a function $G = f(loss)$. The formula for the single gradient descent of our model can be written as formula 4).

$$W' = W + \alpha \cdot (\phi G_{random} + G_{new}) = W + \alpha \cdot f(\phi L_{random} + L_{new}) \quad (4)$$

It can be guessed that if the random sample does not provide useful knowledge information, the gradient brought by the random samples is also random. The total gradient of all random samples should be close to 0, that is, $G_{random} \approx 0$. At this time, the loss of the model is only related to L_{new}. In this way, the learning rate of the model is equivalent to the original $\alpha/(\phi + 1)$, which is equivalent to a model with a low learning rate. We simultaneously train non-incremental models with various learning rates. A comparison of 100 data in a single task with 10 epochs for each task is performed. This comparison results are shown in Fig. 7. The loss curve and the accuracy curve of the non-incremental model have severely jittered, and this phenomenon is as same as the previous experiments that the small training step size causes jitters (see Fig. 4). And our model has both a fast descent rate and a good accuracy rate. The knowledge of the previous task is indeed contained in the artificial samples. In the definition of the Loss method (see formula 2), the ϕ coefficient is a very important parameter. According to the previous discussion, it can be known that when the ϕ is large, the network tends not to update, and when the ϕ is small, the network tends to forget the past knowledge, so the proper ϕ value has become a key issue. We compare the various values of ϕ respectively. The comparison results are as follows (see Fig. 8). In order to measure the degree of this forgetting, we use a smooth filtered curve to fit the points on the current polyline. The distance between this smooth curve and the accuracy polyline is the degree of forgetting during training. Manhattan Distance (L1) is selected to measure the distance between the two curves (red and blue). The bigger the distance between the two curves are, the greater the curve jitter is and the more forgetting is. The distances are shown in Table 1. L1 tends to become smaller when ϕ value increases, that is,

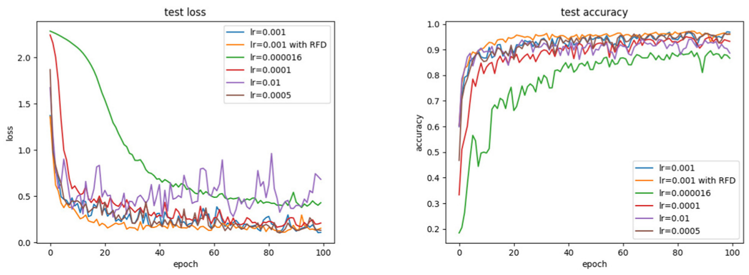

Fig. 7. Comparison with low learning rate training, we select learning rate of 0.01, 0.001, 0.0005, 0.0001 and a special value 0.000016. The learning rate α of our RFD model is 0.01, and the ratio of sample numbers ϕ is 60:1. If random samples do not provide useful knowledge information, the equivalent learning rate is $\alpha/(\phi + 1) = 0.000016$ (Color figure online)

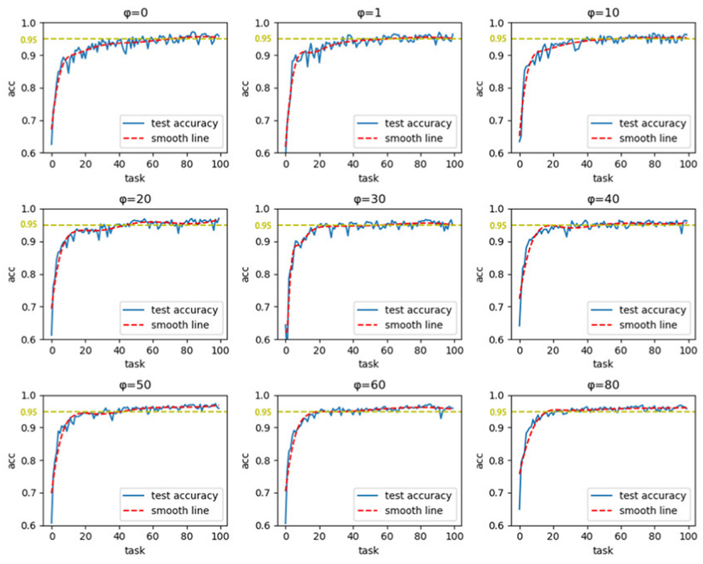

Fig. 8. Test set accuracy under various values of ϕ. This time we choose 60 data in a single task, so after 100 tasks of training all MNIST training data were trained. The blue line represents the accuracy changes when new tasks were added and the red line fitting the blue line. (Color figure online)

the degree of forgetting of the network is becoming smaller, which is consistent with our expectation. When ϕ is large enough, the training cost of the network becomes high because the training data increases. If the training effect does not improve significantly, we should not continue to increase the value of ϕ. This value is approximately 60 in previous experiments.

Table 1. L1 results under various values of ϕ. These values are calculated from Fig. 8.

ϕ	$\phi = 0$	$\phi = 1$	$\phi = 10$	$\phi = 20$	$\phi = 30$	$\phi = 40$	$\phi = 50$	$\phi = 60$	$\phi = 80$
L1	1.098	1.0554	1.004	0.9041	0.917	0.9142	0.8164	0.7455	0.7321

3.4 Comparison with Other Networks

Experiments are run on a laptop with an Intel i7-6700HQ CPU, a NVIDIA 960M GPU, and 16GB memory. LwF method, GAN-based incremental learning method, and EWC method are tested for comparison. To ensure that these methods are comparable, same data and related proportions are provided. The core of the LwF method is loss distillation. The loss value of the method is expressed as $L = L_{new} + \sigma L_{distillation}$. Using the same proportionality coefficient $\sigma = \phi$, RFD and LWF can be compared. The incremental training model based on GAN consists of fake data and new task data. It can be considered that the gradient is composed of new data and fake data. The proportion of new data and fake data is as same as our model. The loss function of the EWC method refers to the weights of the previous task and is defined as $L'(\theta) = L(\theta) + \lambda(W_i - W_0)$. Similarly, the proportionality coefficient λ and ϕ can ensure the comparability of the model. The test accuracy of the four models are comparable, and the accuracy on the test set of the incremental model on the MNIST is also very close. But the time costs are very different. The comparison results are shown in Table 2. The training time of the GAN method consists of a classification network and a GAN network. It is very time-consuming, so the total time is longest. The EWC method needs to repeatedly calculate the fisher matrix, which will become a large time overhead when the amount of data is large. Our method is faster in total time because it is easier to generate data than LwF (Fig. 9).

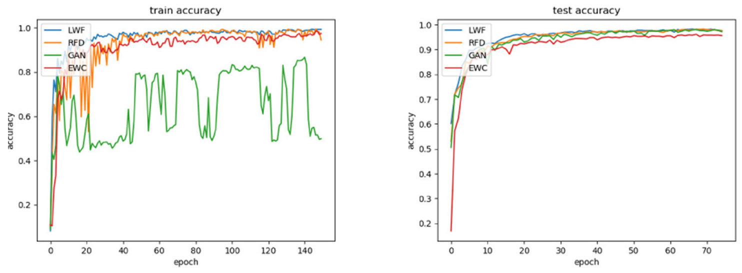

Fig. 9. Comparison results, typical three increasement learning methods and our method (RFD).

Table 2. Comparison of training time and accuracy.

Method	Total time	Model train time	Accuracy
LWF	186.93 s	**79.64 s**	97.59%
GAN	3886.02 s	93.71 s + 3434.74 s	97.19%
EWC	1674.700	1647.96	95.65%
Ours (RFD)	**169.77 s**	79.74 s	**97.66%**

4 Conclusion

In this paper, We propose an incremental learning network model based on a random sample fitting distribution. This method mainly simplifies the generator of a GAN-based incremental learning model. It consists of two parts: The Frozen network responsible for generating random samples and the New network used for training. In the experiments, our new model has the same accuracy as the traditional method while learning faster.

References

1. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. In: International Conference on Learning Representations (ICLR) (2019)
2. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning (2007)
3. French, R.M.: Catastrophic forgetting in connectionist networks. Trends Cogn. Sci. **3**(4), 128–135 (1999)
4. Goodfellow, I.J., et al.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 2672–2680 (2014)
5. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. Comput. Sci. **14**(7), 38–39 (2015)
6. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. U.S.A. **114**(13), 3521–3526 (2017)
7. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. **40**, 2935–2947 (2017)
8. Lopez-Paz, D., Ranzato, M.A.: Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems 30, pp. 6467–6476 (2017)
9. Mehdi Mirza, S.O.: Conditional generative adversarial nets. Comput. Sci. 2672–2680 (2014)
10. Pascanu, R., Bengio, Y.: Revisiting natural gradient for deep networks. Comput. Sci. **37**(s 10–11), 1655–1658 (2013)
11. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
12. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: ICARL: incremental classifier and representation learning. In: CVPR (2016)
13. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Advances in Neural Information Processing Systems 30 (2017)
14. Shmelkov, K., Schmid, C., Alahari, K.: Incremental learning of object detectors without catastrophic forgetting. In: IEEE International Conference on Computer Vision (ICCV), vol. 1, pp. 3420–3429 (2017)
15. Tasar, O., Tarabalka, Y., Alliez, P.: Incremental learning for semantic segmentation of large-scale remote sensing data. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. **PP**(99), 1–14 (2019)
16. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017
17. Yann, L., Leon, B., Bengio, Y., Patrick, H.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
18. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 3320–3328 (2014)
19. Zhai, M., Chen, L., Fred Tung, J.H., Nawhal, M., Mori, G.: Lifelong GAN: continual learning for conditional image generation. In: ICCV (2019)
20. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: CVPR, June 2016