Training Time Adversarial Attack Aiming the Vulnerability of Continual Learning

Gyojin Han[1*] **Jaehyun Choi**[1*] **Hyeong Gwon Hong**[2] **Junmo Kim**[1]
[1]School of Electrical Engineering [2]Kim Jaechul Graduate School of AI
Korea Advanced Institute of Science and Technology (KAIST)
`{hangj0820,chlwogus,honggudrnjs,junmo.kim}@kaist.ac.kr`

Abstract

Generally, regularization-based continual learning models limit access to the previous task data to imitate the real-world setting which has memory and privacy issues. However, this introduces a problem in these models by not being able to track the performance on each task. In other words, current continual learning methods are vulnerable to attacks done on the previous task. We demonstrate the vulnerability of regularization-based continual learning methods by presenting simple task-specific training time adversarial attack that can be used in the learning process of a new task. Training data generated by the proposed attack causes performance degradation on a specific task targeted by the attacker. Experiment results justify the vulnerability proposed in this paper and demonstrate the importance of developing continual learning models that are robust to adversarial attack.

1 Introduction

Recent continual learning (also termed lifelong or incremental learning) [2] methods have overcome *catastrophic forgetting* [5] showing remarkable performance in both past and current tasks. However, in the real-world situation, it is impossible to verify whether the model still works well on past tasks since previous data is not available due to memory and privacy problems. In other words, the continual learning models can only be validated by the training accuracy thereby blindly utilizing a model if the training accuracy is high even when the performance on one of the tasks is low. The difficulty of tracking the reliability of a model on past tasks is a serious problem especially if the attack is done on a specific task that a model has learned in the past as it would not affect the training accuracy. Despite such a problem, adversarial attacks [6] and defenses are not actively discussed in the field of continual learning.

In this paper, we bring forward the aforementioned problem in continual learning for the first time and experimentally justify the vulnerability with a simple task-specific training time adversarial attack. The attack is designed to not affect the training accuracy of a model thus indistinguishable when training. More specifically, we generate adversarial data [3] during training that behaves in a new way, to work in continual learning. The generated adversarial data of a new task only severely degrades the performance of a targeted task.

2 Related Works

Continual Learning. Continual learning has three streams: rehearsal-based methods [14, 11], architecture-based methods [15, 18, 4, 13, 16], and regularization-based methods [8, 19]. Among them, regularization-based methods add a regularization loss term to the loss function when learning a

[*]Equal contribution.

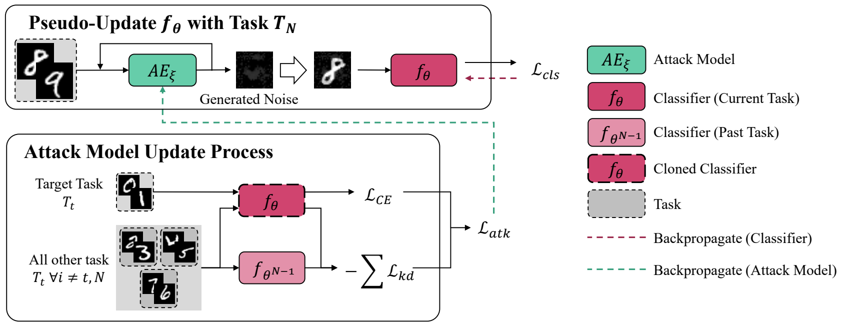

Figure 1: The attack model loss $\mathcal{L}_{atk}$ is calculated from the pseudo-updated classifier f_θ and classifier from the past task $f_{\theta^{N-1}}$. The gradient is transmitted to the attack model AE_ξ through partial differentiation. The attack model AE_ξ then generates noise that gets added to the original data from the current task which updates the pseudo-updated classifier f_θ with cross-entropy loss.

new task to reduce the amount of change in parameters that are important for classifying the previous tasks. In this paper, we consider the regularization-based method setting, having no access to data from past tasks during the training process for the new task. As there are no previous task data available, the regularization-based continual learning method cannot track the performance of the past task hence relying solely on the training accuracy of the current task when deploying the model. This opens the chance for adversarial attacks to easily attack the continual learning methods on previous tasks without getting detected during training time. We demonstrate how vulnerable the continual learning methods are with a simple task-specific training time adversarial attack.

Adversarial Attack. First introduced in [6], adversarial examples refer to samples with a very small perturbation, usually imperceptible by human eyes but noticeable by machine learning models, creating a gap in the inference results between human and machine learning models. Adversarial attacks are methods of generating such adversarial examples. It can be categorized into test time adversarial attacks [6, 12, 1] and train time adversarial attacks [3]. Test time adversarial attacks generate images during inference and aim for incorrect inference results whereas train time adversarial attacks generate images when training to make the model to be trained erroneously. In this work, we propose a train time adversarial attack for continual learning that adds perturbation to the training data of a new task so that the victim continual learning model loses information about the previous task specified by the attacker while learning the new task.

3 Proposed Method

In this section, we describe how an attacker generates adversarial data that cause a continual learning model to lose information on a previous task. More specifically, the model trainer is provided with adversarial training data that does not affect the training accuracy of a new task to prevent the trainer from detecting the attack. Moreover, as the model user might lose trust in the model if the model does not work for all previous tasks, we propose an attack that only affects the performance of a specific task that the attacker intends to drop the performance of. We assume a regularization-based method setting in which training data $\{D_1, \ldots, D_N\}$ corresponding to N tasks $\{T_1, \ldots, T_N\}$ were provided sequentially. Training on the data of the new task proceeds without access to previous data. The attacker is provided with the training data $\{D_1, \ldots, D_N\}$ and a classifier $f_{\theta^{N-1}}$ trained up to $(N-1)$-th task. The goal of the attacker is to make the victim classifier lose knowledge about a target task T_t while being trained well on the new task. We emphasize that the attacker only slightly modifies the training data of T_N for flawless training on T_N while losing information of T_t.

3.1 The attack process

We use an attack model AE_ξ with an encoder-decoder structure to manipulate the training data of a new task into adversarial data. It takes the clean training data of the new task D_N as input and generates noise that is bounded by $(-\epsilon, \epsilon)$. The generated noise is added to D_N, and becomes the adversarial training data ${D_N}'$ that can degrade the performance of the continual learning model. To train the attack model AE_ξ, we use the optimization method proposed by [3] with modifications in training process. The modified training process repeats the following two steps, (1) recording the trajectories of a temporary model by updating it with adversarial data, and (2) training the attack model along the trajectories by pseudo-updating the recorded parameters. Figure 1 illustrates the second step of training process of the attack model AE_ξ.

Recording the trajectories of a temporary model. For the optimization of AE_ξ, we need to approximate the trajectories of $f_{\theta^{N-1}}$ when it learns the adversarial image generated by AE_ξ. Therefore, we use a temporary model f_θ for episodic training. f_θ is trained with adversarial training data ${D_N}'$ generated by fixed AE_ξ. At this time, f_θ should be trained with a continual learning approach, as in the actual situation. Therefore, the loss for f_θ, $\mathcal{L}_{cls}$ is:

$$\mathcal{L}_{cls} = \mathcal{L}_{CE}\left(f_\theta(x^N + AE_\xi(x^N)), y^N\right) + \Omega_m^{N-1} \tag{1}$$

where Ω_m^{N-1} is the regularization term of the continual learning method m, and (x^N, y^N) is the mini-batch of the training data of the N-th task. Through the loss $\mathcal{L}_{cls}$, the parameters θ are updated and recorded as follows:

$$\theta \leftarrow \theta - \alpha_f \cdot \nabla_\theta \mathcal{L}_{cls} \tag{2}$$

where α_f is the learning rate for f_θ.

Training the attack model along the trajectories. We pseudo-update recorded f_θ using the image generated by AE_ξ with same loss $\mathcal{L}_{cls}$ as in step (1). Cross-entropy loss is calculated from the data of the target task D_t using pseudo-updated f_θ, and gradient values are transmitted to AE_ξ through the loss. Then AE_ξ can be updated with the gradient ascent. However, owing to the relevance between tasks, if an attack on the target task T_t is attempted without any restrictions, the performance of the classifier against the other tasks involved will also be reduced. Therefore, appropriate constraints are required to maintain the classifier performance for other tasks. We want to preserve the outputs of inferences of $f_{\theta^{N-1}}$ for other tasks even if it is trained using adversarial training data. Therefore, we add the knowledge distillation loss term [7] to the loss function for training AE_ξ. The knowledge distillation loss between the outputs of $f_{\theta^{N-1}}$ and f_θ, prevents adversarial data from affecting the inferences on other tasks. To calculate knowledge distillation loss, we sampled the data from the training data of each task. The knowledge distillation loss term for T_k is:

$$\mathcal{L}_{kd} = \alpha_{kd} \cdot T^2 \mathcal{L}_{KLD}\left(\sigma\left(\frac{f_\theta(x^k)}{T}\right), \sigma\left(\frac{f_{\theta^{N-1}}(x^k)}{T}\right)\right) \tag{3}$$

where α_{kd} is the balancing parameter of the knowledge distillation loss, $\mathcal{L}_{KLD}$ is the KL-divergence loss, T is the temperature parameter, and $\sigma(\cdot)$ is the softmax function. Because AE_ξ is trained via gradient ascent, the knowledge distillation loss term for all tasks except T_t and T_N is subtracted from the cross-entropy loss. The loss for AE_ξ, including the knowledge distillation loss term is:

$$\mathcal{L}_{atk} = \mathcal{L}_{CE}\left(f_\theta(x^t), y^t\right) - \sum_{i \neq t, N} \mathcal{L}_{kd}(f_\theta(x^i), f_{\theta^{N-1}}(x^i)) \tag{4}$$

Finally, the parameters of the attack model ξ are updated as follows.

$$\xi \leftarrow \xi + \alpha_{AE} \cdot \nabla_\xi \mathcal{L}_{atk} \tag{5}$$

4 Experiments

We experiment the proposed attack on two continual learning methods, Elastic Weight Consolidation (EWC) [8] and Synaptic Intelligence (SI) [19] with two variants of MNIST [9] dataset, permuted MNIST [17] and split MNIST [19]. The details regarding baseline model, dataset, and training process is described in the Supplementary.

Experiment details. The attack model consists of an encoder and decoder. The encoder has 3 $\times$ 3 convolution layers with 16, 64, and 128 channels, and the decoder has a 5 $\times$ 5 convolution layer with 128 channels and a 2 $\times$ 2 convolution layer with 64 channels. We train the attack model with Adam optimizer for 10 epochs with learning rate of 0.0001 and batch size of 256. The weight ϵ which determines the magnitude of the generated noise when adding to the clean sample is set to 0.2. All of the experiments including baseline model are trained with SGD optimizer.

Permuted MNIST

Method		T_1	T_2	T_3	T_4	T_5
SGD		0.8059	0.8817	0.9306	0.9558	0.9520
EWC	Plain	0.9029	0.9493	0.9570	0.9623	0.9700
	Noise	0.8657	0.9356	0.9273	0.9687	0.9617
	Ours-T_1	**0.6005**	0.9344	0.9360	0.9666	0.9620
	Ours-T_2	0.9054	**0.5625**	0.9486	0.9639	0.9619
SI	Plain	0.9102	0.9605	0.9529	0.9658	0.9717
	Noise	0.9152	0.9561	0.9324	0.9589	0.9646
	Ours-T_1	**0.5190**	0.9364	0.9386	0.9592	0.9589
	Ours-T_2	0.8584	**0.5618**	0.9013	0.9183	0.962

Split MNIST

Method		T_1	T_2	T_3	T_4	T_5
SGD		0.4019	0.5901	0.1441	0.9084	0.9844
EWC	Plain	0.4317	0.7424	0.1254	0.9305	0.9813
	Noise	0.4132	0.6459	0.1660	0.8676	0.9803
	Ours-T_1	**0.3825**	0.5843	0.1596	0.9592	0.9773
	Ours-T_2	0.4463	**0.5563**	0.2006	0.9350	0.9692
SI	Plain	0.4790	0.8242	0.3010	0.9728	0.9531
	Noise	0.4643	0.7919	0.3116	0.9733	0.9531
	Ours-T_1	**0.3939**	0.8095	0.4242	0.8454	0.9576
	Ours-T_2	0.4577	**0.7767**	0.4248	0.8348	0.9551

Table 1: Final accuracy of the victim classifier f_θ for checking the performance of our attack on various training methods and datasets.

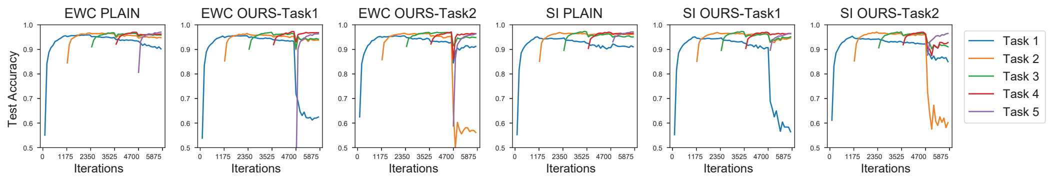

Figure 2: Test accuracy on permuted MNIST for five tasks using EWC and SI. The graphs named 'PLAIN' show the effect of continual learning when no attacks are applied. The graphs named 'OURS-Task1' and 'OURS-Task2' show the test accuracy when T_1 and T_2 were attacked by our method, respectively. Best viewed zoomed in.

Results. Table 1 shows the final accuracy of the continual learning model after training for all tasks is completed. SGD in Table 1 is the baseline method. The 'Plain' results of EWC and SI show the effectiveness in continual learning task by being higher than baseline results. Additionally, 'Noise' denotes the model trained on images with random uniform noise. 'Ours-T_1' and 'Ours-T_2' denote our task-specific training time attack done on task 1 and task 2, respectively.

As can be seen in Figure 2, the perturbation created by the proposed attack method caused the victim classifier to forget the knowledge about the specific target task as it learns T_5. This proves the existence of adversarial data that causes much more severe catastrophic forgetting compared with clean data as can be seen by the results of 'Noise', 'Ours-T_1', and 'Ours-T_2' in Table 1. The targeted task result of 'Ours-T_1' and 'Ours-T_2' being lower than the 'Noise' shows that our attack method successfully attacks the targeted task. More importantly, the performance of non-targeted tasks stays in the reasonable range in line with the continual learning methods. This demonstrates that the attack on the target task cannot be determined until inference about the target task occurs even for the deployed models. Furthermore, this shows that highly covert attacks on past tasks are possible because of the untraceable accuracy problem for past tasks in continual learning.

Future works. In this study, we focused on unveiling the vulnerability of continual learning for the first time. We expect that experiments on various datasets and continual learning methods will be conducted in the future, and more effective attack & defense methods for continual learning will be studied.

5 Conclusion

In this paper, we propose a simple training time task-specific adversarial attack and experimentally prove the vulnerability of continual learning. More specifically, the inability to access previous tasks' data in continual learning causes model users to solely rely on the training accuracy without acknowledging the performance for each task. We highlight the importance of developing continual learning models that are robust to task-specific adversarial attacks by experimentally demonstrating the performance degradation on a specific task with the proposed simple training time task-specific adversarial attack.

References

[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[2] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[3] Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. Learning to confuse: Generating training time adversarial data with auto-encoder. In *Advances in Neural Information Processing Systems*, pages 11971–11981, 2019.

[4] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[5] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[6] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

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

[8] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[9] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. 1998.

[10] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[11] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[13] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] 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 (CVPR)*, July 2017.

[15] 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.

[16] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557. PMLR, 10–15 Jul 2018.

[17] 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.

[18] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[19] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

A Algorithm of the Proposed Attack

In Section 3, we propose a task-specific training time adversarial attack on continual learning. The entire procedure of the attack is illustrated in Algorithm 1.

Algorithm 1 Proposed Attack

```
Require: victim classifier $f_{\theta^{N-1}}$, training data $\{D_1, \dots, D_N\}$
Ensure: attack model $AE_\xi$
 1: $\xi \leftarrow RandomInit()$
 2: for $epoch \leftarrow 1$ to $max\_epochs$ do
 3:     $\theta_0 \leftarrow \theta^{N-1}$
 4:     $L \leftarrow$ new list
 5:     for $i, (x^N, y^N)$ in $enumerate(D_N)$ do
 6:         $L$.append($\theta_i, \mathcal{L}_{cls}(f_{\theta_i}(x^N + AE_\xi(x^N)), y^N)$)
 7:         $\theta_{i+1} \leftarrow \theta_i - \alpha_f \cdot \nabla_{\theta_i} \mathcal{L}_{cls}(f_{\theta_i}(x^N + AE_\xi(x^N)), y^N)$
 8:     end for
 9:     for $i, (\theta_i, \mathcal{L}_{cls})$ in $enumerate(L)$ do
10:         $\theta' \leftarrow \theta_i - \alpha_f \cdot \nabla_{\theta_i} \mathcal{L}_{cls}$
11:         $\mathcal{L}_{atk} \leftarrow 0$
12:         for $j \leftarrow 1$ to $N-1$ do
13:             $(x^j, y^j) \leftarrow RandomSample(D_j, sample\_size)$
14:             if $j = t$ then
15:                 $\mathcal{L}_{atk} \leftarrow \mathcal{L}_{atk} + \mathcal{L}_{CE}(f_{\theta'}(x^j), y^j)$
16:             else if $j \neq t$ then
17:                 $\mathcal{L}_{atk} \leftarrow \mathcal{L}_{atk} - \mathcal{L}_{kd}(f_{\theta'}(x^j), f_{\theta^{N-1}}(x^j))$
18:             end if
19:         end for
20:         $\xi \leftarrow \xi + \alpha_{AE} \cdot \nabla_\xi \mathcal{L}_{atk}$
21:     end for
22: end for
23: return $AE_\xi$
```

B Dataset and Baseline Model Details.

For permuted MNIST, we randomly permute pixels of MNIST dataset differently for each task while maintaining the original MNIST dataset as one of the tasks. Following previous methods, we construct a classifier with 4 fully-connected layers consisting of 400 neurons with dropout set to 0.5. We set the hyperparameter of EWC and SI, λ and c, to 40 and 0.2, respectively, and set the batch size and learning rate as 256 and 0.1, respectively.

For split MNIST, we split the original MNIST dataset into 5 subsets each containing consecutive digits. Thereby, the model learns to distinguish the difference between two consecutive digits from 0 to 10. We construct a classifier with 3 fully-connected layers with 400 neurons and dropout set to 0.2. We set the hyperparameter of EWC and SI, λ and c, to 40 and 0.1. The batch size and learning rate to 128 and 0.001, respectively.

C Adversarial Samples

In this section, we provide adversarial samples generated using the proposed attack. The samples of permuted MNIST and Split MNIST are in Figure 3 and Figure 4, respectively.

C.1 Permuted MNIST

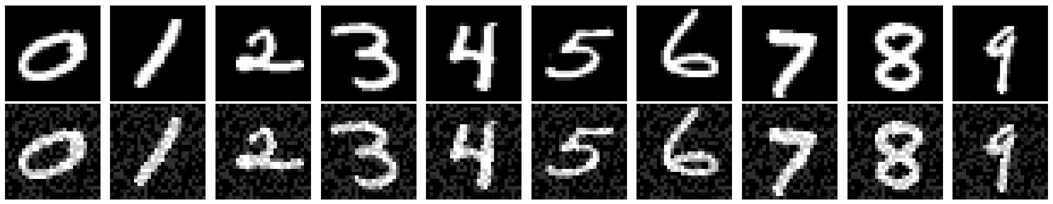

Figure 3: Images obtained by setting the new task to the original MNIST for visualizing adversarial data. Clean samples (upper) and adversarial samples created using the attack model (bottom).

C.2 Split MNIST

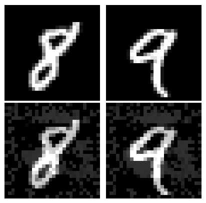

Figure 4: Clean samples (upper) and adversarial samples created using the attack model (bottom).

D Ablations

D.1 Observation of regularization loss

Regularization loss is the only information regarding previous tasks that the trainer can identify during the learning process. Therefore, we observe the changes in the regularization loss due to the attack. Figure 5 shows the changes in regularization loss of continual learning models that are used

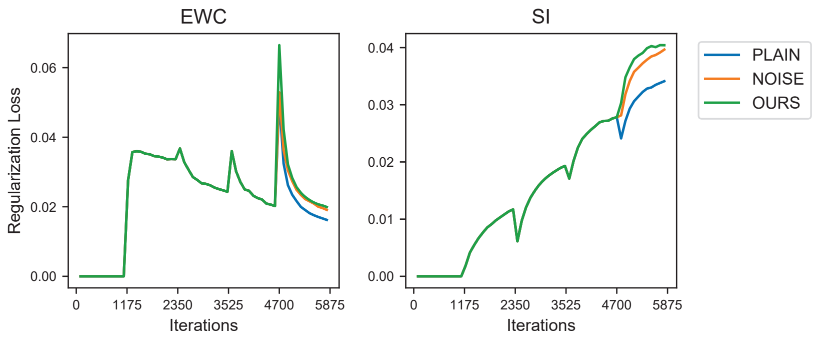

Figure 5: Regularization loss during training of continual learning models.

in the experiments. The regularization loss increases slightly when the proposed attack is applied. However, not only is the increase in regularization loss small, but for the trainer, there is nothing to compare the regularization loss with. In addition, we confirmed that depending on the initialization of the classifier, the regularization loss of the unattacked classifier is often greater than that of the attacked classifier. These points make it extremely difficult for the trainer to establish a policy to detect the proposed attack based on regularization loss.

D.2 Knowledge distillation loss

We calculated the backward transfer [10] of the new task for the remaining tasks, except the target task to confirm the effectiveness of the knowledge distillation loss term. The backward transfer B was calculated as follows:

$$\mathrm{B} = \frac{1}{N-2} \sum_{i \neq t, N} R_{N,i} - R_{i,i} \tag{6}$$

where $R_{i,j}$ is the test accuracy of the classifier on T_j just after trained with T_i.

	Plain	Ours w/o $\mathcal{L}_{kd}$	Ours with $\mathcal{L}_{kd}$
EWC	-0.0129	-0.0717	-0.0195
SI	-0.0087	-0.0600	-0.0237

Table 2: Backward transfer of T_5 for T_2, T_3, and T_4 on permuted MNIST.

As shown in Table 2, by placing a constraint on using the knowledge distillation loss when training the attack model, the increase in negative backward transfer owing to the attack is significantly reduced.