Targeted Data Poisoning Attacks Against Continual Learning Neural Networks

Huayu Li and Gregory Ditzler

Department of Electrical & Computer Engineering, *University of Arizona*
Tucson, AZ 85721
hl459@email.arizona.edu, ditzler@arizona.edu

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892774

***Abstract*—Continual (incremental) learning approaches are designed to address catastrophic forgetting in neural networks by training on batches or streaming data over time. In many real-world scenarios, the environments that generate streaming data are exposed to untrusted sources. These untrusted sources can be exposed to data poisoned by an adversary. The adversaries can manipulate and inject malicious samples into the training data. Thus, the untrusted data sources and malicious samples are meant to expose the vulnerabilities of neural networks that can lead to serious consequences in applications that require reliable performance. However, recent works on continual learning only focused on adversary agnostic scenarios without considering the possibility of data poisoning attacks. Further, recent work has demonstrated there are vulnerabilities of continual learning approaches in the presence of backdoor attacks with a relaxed constraint on manipulating data. In this paper, we focus on a more general and practical poisoning setting that artificially forces catastrophic forgetting by clean-label data poisoning attacks. We proposed a *task targeted data poisoning* attack that forces the neural network to forget the previous-learned knowledge, while the attack samples remain stealthy. The approach is benchmarked against three state-of-the-art continual learning algorithms on both domain and task incremental learning scenarios. The experiments demonstrate that the accuracy on targeted tasks significantly drops when the poisoned dataset is used in continual task learning.**

***Index Terms*—continual learning, adversarial machine learning, data poisoning attack.**

I. Introduction

Neural networks suffer catastrophic forgetting [1] when they are tasked with learning from sequential or streaming data [2]–[4]. As a result of catastrophic forgetting, a well-trained neural network can partially or even entirely forget the information about the previously-learned knowledge. Unlike traditional offline learners trained on single task where the entire dataset is sampled from an i.i.d. distribution, continual learners are trained from tasks that typically change over time. Continual (incremental) learning approaches are designed to retain knowledge of historical tasks by protecting the model parameters that are important for previous tasks from intensive change when training models on new coming tasks. Hence, a continual learning setting requires that a model maintain a healthy balance between *stability* and *plasticity* [3]. Stability is the property of a model to retain old knowledge and plasticity is the property that allows the model to learn from new data. Continual learning approaches enable intelligent systems to achieve life-long learning on massive data under the limitation of finite storage and evolve on rapidly updating data sources while preserving the capability to process the earlier tasks.

Continual learning focuses on learning from tasks – datasets – over time. Tasks are presented to the model sequentially over time for training and the model can be asked to predict tasks that it has previously learned. Therefore, it is crucial to maintain a healthy balance between *stability* and *plasticity*. Further, information learned about the different tasks can be transferred to new tasks when they are presented to the model. Note that the definition of continual learning is similar to transfer and multi-task learning; however, there are differences worth noting. The field of transfer learning focuses on techniques that can effectively learn new tasks from older tasks and the goal is to perform well on the new task. Transfer learning has been successfully applied in many domains where limited data are available from the target task [5]. Similarly, multi-task learning focuses on maximizing the performance across the tasks presented to the model. The concept of lifelong learning has aspects of transfer and multi-task learning [6]; however, it differs in that it seeks to learn new tasks by efficiently building on prior task learning while maximizing the performance across all tasks.

Unfortunately, recent work has shown that neural networks are vulnerable to adversarial attacks that cause deleterious predictions and decrease the robustness of the model [7], [8]. Adversaries can inject small perturbations into the original data that cause errors. These perturbations can be injected at training or testing time, which are referred to as poisoning or evasion attacks, respectively. In this work, we focus on adversarial poisoning attacks where the adversary has access to the training data (or some knowledge about the task). Their goal is to craft examples that decrease the performance of the neural network. Recent work has shown that many approaches that generate poison sample are actually easy to detect (see the *strength vs. detectability dilemma* [9]). Poisoning attacks can be identified as one of three primary categories. Backdoor attacks [10] inject the backdoor instances into the training data that mislead the learning system make the incorrect predictions on test data with backdoor patterns. Targeted data poisoning attacks [11]–[13] achieve triggerless backdoor attacks without adding the backdoor patterns to the target data to fool the network. Finally, availability attacks [14] can cause more damage to the model by directly degrading the performance

without any modification on the data during testing. As a result, the neural networks trained on the malicious data will be less trustworthy.

While several data poisoning methods for offline learning models have been recently proposed, data poisoning in continual learning scenarios has received significantly less attention. There are two primary objectives of poisoning attacks against continual learning. First, the attackers aim to erase the knowledge about all tasks, which is identical to attacking offline learning. Another goal is to attack the specific tasks' knowledge while not affecting others. Recently, backdoor attacks against continual learning have been proposed to create a "false memory" about the targeted task [15], [16]. The samples from the tasks under attack with the backdoor patterns can easily bypass the neural network. However, there is still an issue with the existing approaches: *can we make catastrophic forgetting intentional by data poisoning while the attack samples are stealthy (i.e., difficult to detect)*?

This paper explores task-specific availability attacks against continual learning. Specifically, this work develops a novel data poisoning attack that significantly reduces the target task performance over time and the adversarial samples are difficult to detect. We demonstrate that injecting poison samples into upcoming tasks could produce catastrophic forgetting to the targeted task artificially while making minimal interference on performance on non-targeted tasks. Compared with the previous works [15], [16], the presented work considered a new data poisoning setting. First, the samples from the historic task will not show up in current training data. Second, a clean label attack was adopted that the corresponding labels of the training data are not corrupted. Further, unlike previous works, we do not add backdoor patterns in the test data at inference time. We benchmark our approach against three continual learning algorithms on two datasets.

II. Related works

A. Continual learning

In general, continual learning algorithms are categorized into replay-based, architecture-based, and regularization-based methods. Replay-based methods store the original samples from previous tasks [17], [18] or use generative models to generate the samples of previous tasks [19]. The stored or generated examples from prior tasks are replayed and concatenated with the training data of the current task during the training phase to overcome catastrophic forgetting. Architecture-based methods separate the neural network into sub-networks for each task to ensure the parameters for each task have minimal effect on the other sub-networks [20], [21]. Regularization-based approaches were proposed to address the data storage and privacy problem in replay-based methods and architectural complexity issues associated with architecture-based methods [22]–[24]. These methods compute the importance to the learned tasks of each parameters, and store the importance into matrix. When the network is trained on new tasks, the importance matrix is used as regularization terms to prevent too large updates on the parameter associated with old tasks [6] This work explores the vulnerability of three common regularization-based algorithms, namely, Elastic Weight Consolidation (EWC) [22], Online Elastic Weight Consolidation (online EWC) [23], and Synaptic Intelligence (SI) [24]. We choose these methods due to their popularity, and they do not require previous data samples be retained.

In this paper, two critical and practical scenarios known as domain and tasks incremental learning [25] are considered for evaluating the vulnerability of continual learning models. In domain incremental learning, the data are sampled from distributions/domains between tasks with fixed numbers and type of class. In contrast, incremental task learning is a more challenging because the distributions/domains of both the data and the classes differ between tasks, while only the number of the classes remains fixed. Further, when the tasks are learned sequentially, the labels are assigned to class IDs based on their original categories. Thus, the inference phase can predict the classes of the inputs without task IDs.

B. Adversarial Machine Learning

The reliability and robustness of neural networks has drawn much attention with the broad application of neural networks in recent years. Adversarial machine learning explores the vulnerabilities of machine learning algorithms and performs attacks to control the behavior of machine learning algorithms in two major aspects. Evasion/adversarial attacks [8], [26]–[29] exploit the blindspots of trained models where they catastrophically misclassify the inputs with adversarial perturbations at the test time. Evasion attacks find adversarial perturbations by maximizing the loss with respect to the inputs under certain constraints. The input data with even imperceivable perturbations in the direction would significantly change the feature representations in a deep network and the logit outputs. Poisoning attacks inject malicious data points into the training data to enforce the model converging to the wanted points of the attackers.

Poisoning attacks can broadly categorized into backdoor, targeted data poisoning, and poison availability attacks. In the backdoor attacks setting [10], the adversarial samples are generated with backdoor patterns that are inserted into the training images to create an association between the backdoor pattern and the incorrect labels. During testing, the neural network can be fooled by inputs tagged with predefined backdoors – or triggers – while having similar performance to the normal model when the triggers do not activate the backdoor. Thus, backdoor attacks are regarded as an intermediary between data poisoning attacks and evasion attacks. Targeted data poisoning attacks [11]–[13] insert triggerless backdoor with only access to the training data, but cannot modify test data. Targeted data poisoning aims to cause the target samples to be misclassified to a specific class. Most work on poison availability attacks focused on reducing the accuracy in general that makes the model unusable [14], and producing gradient vanishing when model learning on the modified data that make the data unexploitable [30].

This paper proposes a new scenario for poison availability attacks against continual learning models. The poisoned samples are designed to produce an intentional catastrophic

Algorithm 1 Regularization-based CL algorithms

Input Training data samples received for task τ: $\{X_\tau, Y_\tau\}$; total number of tasks: T; model parameter: θ; regularization parameter: λ

for τ = 1 to T **do**
  if $\tau == 1$ **then**
    $\theta^*_\tau = \arg\min_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y_\tau)$
  else
    $\theta^*_\tau = \arg\min \mathcal{L}(f(X_\tau; \theta_\tau), Y_\tau) + \lambda \left\| \theta_\tau - \theta^*_{\tau-1} \right\|^2$
  end if
end for

forgetting that degrades the neural network's performance on a specific historical task when training on current tasks. Unlike conventional poison availability attacks, the presented method forces the network to forget the previously learned data instead of reducing the overall performance on all learned data. Further, our method inserts false memory of the data sampled from a targeted domain instead of being limited to specific target samples compared to targeted data poisoning attacks. Thus, we name the proposed method as *task targeted poisoning attack* to distinguish it from poison availability attacks and targeted data poisoning attacks.

III. Task Targeted Poisoning

This section motivates the proposed poisoning attacks and the continual learning settings for streaming data. We denote the output of a neural network with model parameters θ as $f(x; \theta)$, where f has a soft-max output. The goal of the neural network is to minimize the empirical risk is the main objective function $\mathcal{L}(f(x; \theta), y)$ for training the network, where (x, y) refers single input-label pair. The risk, or loss, for the tasks considered in this work is cross-entropy. In contrast, a dataset or a data batch are defined as $\mathcal{D} = \{X, Y\} = \{x_i, y_i\}_{i=1}^N$, as well as the risks on a dataset or a data batch are defined as $\mathcal{L}_\theta(\mathcal{D})$. In the data poisoning setting, we define the dataset for a task τ as $\mathcal{D}_\tau$, where the adversary seeks to target the model to forget task τ. The adversary's objective is to degrade the performance on $\mathcal{D}_\tau$ even when data from new tasks are presented. Finally, we define tasks $\mathcal{D}_{\tau+n}, \forall n = 1, \ldots, T - \tau$ as the non-target tasks where the poison samples are injected.

A. Regularization-based continual learning algorithms

In the continual learning setting, the model receives a new training data from a task and the labels which are denoted by X_τ and Y_τ at time $\tau = 1, \ldots, T$, respectively. The goal of the continual learner is to find the optimal parameters θ^*. By optimal, we assume that the goal is to minimize the empirical risk across all tasks. More formally, this objective is given by

$$\min_\theta \sum_{\tau=1}^{T} \left[\mathcal{L}_{\theta_\tau}(\mathcal{D}_\tau) + \lambda \sum_i I_{\tau-1} (\theta_\tau - \theta^*_{\tau-1})^2 \right] \quad (1)$$

where the regularization term penalizes changes to those parameters proportional to the importance matrix $I_{\tau-1,i}$. The importance matrix is calculated from the previous tasks $\mathcal{D}_{\tau-1}$. The regularization coefficient $\lambda \geq 0$ controls the forgetting factor for previously learned tasks. The weight of the regularizer can be interpreted as follows: the larger λ results in the continual learner retaining more previous knowledge and less learning on new tasks, and vice versa. The primary difference between the regularization-based continual learning algorithms is in the calculation of the importance matrix, $I_{\tau-1,i}$. In our setting, the regularization term can be regarded as a variant of L2 regularization that performs knowledge preservation and natural defense to poisoning perturbations. In our attacking setting, we simply the learning procedure as Algorithm 1 without considering any data-dependent importance matrix.

B. Attack Strategy

In this work, we define the adversary and the defender model as the two parties involved in the data poisoning setting. The adversary's goal is to generate adversarial samples with some prior knowledge about the defender. The defender's knowledge plays a crucial role in how they generate the data. The adversary can add data into defender's training set that is crafted to reduce the defender's performance. The adversary can inject these malicious data into all time steps during continual learning.

We consider a *white-box* attack setting, where the adversary has full knowledge about the defender's model. Note that a white-box adversarial setting is the worst case for the defender since these settings allow the adversary access to the data and the defender's classifier. Thus, the adversary has access to the defender's model parameters $\theta_{\tau-1}$, the dataset for the new task $\mathcal{D}_\tau$, and the dataset of target task $\mathcal{D}_t$ where $t < \tau$. However, due to the consideration for generality, the defender updates their parameters using the cross-entropy loss, and the adversary does not have access to the loss value and the importance matrix. The adversary has limited control over the training data other than the fact they can inject a limited number of samples. The adversary can add ℓ_∞-norm ϵ-bounded perturbations to a fraction of the current training dataset. The adversary performs only clean-label attacks, which means the adversary is not permitted to change the original label of a poisoned image. Meanwhile, samples from the target task is not allowed to be injected into the current training data.

C. Task Targeted Poisoning Attacks

The task targeted poisoning attack scenario is where the defender's network is first trained on a clean target dataset then updated over time on several poisoned datasets. In the threat model of task targeted poisoning attacks, the attacker aims to degrade the defender's performance on the target task after the model is trained on the non-target tasks. More formally, let $\mathcal{D}_\tau$ be the target dataset and $\mathcal{D}_{\tau+n}, \forall n = 1, \ldots, T - \tau$ as the non-target datasets. The adversary modifies $\mathcal{D}_{\tau+n}$ by injecting poisoned samples which creates the adversarial datasets denoted by $\mathcal{D}^{adv}_{\tau+n}$. The adversary's objective function for generating malicious samples is given by:

$$\max_{\mathcal{D}^{adv}_{\tau+n}} \mathcal{L}_{\theta^*}(\mathcal{D}_\tau), \text{and } \theta^* \in \arg\min_\theta \sum_{n=1}^{T-\tau} \mathcal{L}(\mathcal{D}^{adv}_{\tau+n}), \quad (2)$$

Algorithm 2 Poisoning attack against continual learning

Input model parameters: θ; target dataset batches: $S_\tau = \{X_\tau, Y_\tau\}$; poisoning dataset batches: $S_{\tau+n} = \{X_{\tau+n}, Y_{\tau+n}\}$; the size of perturbation: ϵ; iterations: T; decay factor: μ; numbers of class: C; step size: α;

procedure START POISONING
  initialize momentum $g_0 = 0$ and poison samples $X^{adv}_{\tau+n} = X_{\tau+n}$;
  flip target label $Y^{adv}_\tau = Y_\tau + randint(1, C-1)\%C$;
  compute target gradients $\Delta^t_\theta = \nabla_\theta \mathcal{L}(f(X_\tau, \theta), Y^{adv}_\tau)$;
  for $t = 0$ to $T-1$ **do**
    compute poison gradients $\Delta^p_\theta = \nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+n}, \theta), Y_{\tau+n})$;
    compute cosine similarity $\mathcal{H} = \frac{\langle \Delta^t_\theta, \Delta^p_\theta \rangle}{\|\Delta^t_\theta\| \|\Delta^p_\theta\|}$;
    update momentum $g_{t+1} = \mu \cdot g_t + \frac{\nabla_X \mathcal{H}}{\|\mathcal{H}\|}$;
    update $X^{adv}_{\tau+n} = X^{adv}_{\tau+n} + \alpha \cdot \text{sign}(g_{t+1})$;
  end for
  return $X^{adv}_{\tau+n}$.
end procedure

where the $\mathcal{D}_\tau$ and $\mathcal{D}_{\tau+n}$ are not sampled from the same underlying distribution. If we ignore the data-independent regularization term in the simplified version of regularization-based continual learning algorithms in Algorithm 1. The minimization problem is typically optimized using stochastic gradient descent (SGD).

Let us consider a single epoch in gradient descent on the entire dataset for further simplifying the problem, the model parameters after training on a task at task $\mathcal{D}_{\tau+1}$ are updated as:

$$\theta_{\tau+1} = \theta_\tau - \eta \nabla_\theta \mathcal{L}_{\theta_\tau}(\mathcal{D}_{\tau+1}) \tag{3}$$
$$= \theta_\tau - \eta \nabla_\theta \mathcal{L}(f(X_{\tau+1}; \theta_\tau), Y_{\tau+1}), \tag{4}$$

where $\eta \geq 0$ is the learning rate. In practice, the update term might include momentum (e.g., momentumSGD [31]) or adaptive gradient methods (e.g., Adam [32] and AdaGrad [33]); however, for brevity, we omit these terms from the update equation.

The most intuitive way to forget the previous knowledge using gradient descent is to inject samples from the target tasks with incorrect labels on the new tasks. Let us define $\tilde{\mathcal{D}}^{adv}_\tau$ as a subset of the target $\mathcal{D}_\tau$ with flipped labels Y^{adv}_τ. These data are inserted to the new training data $\mathcal{D}_{\tau+1}$ for the defender to use for training their model. Then the gradient descent update to minimize the loss during the training of current task is given by:

$$\theta_{\tau+1} = \theta_\tau - \eta \nabla_\theta \mathcal{L}_{\theta_\tau}(\mathcal{D}_{\tau+1} \cup \tilde{\mathcal{D}}^{adv}_\tau) \tag{5}$$
$$= \theta_\tau - \eta \nabla_\theta \mathcal{L}(f(X_{\tau+1}; \theta_\tau), Y_{\tau+1}) - \tag{6}$$
$$\eta \nabla_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y^{adv}_\tau), \tag{7}$$

Note that this expression shows that the defender's model minimizes the loss on current task and the adversarial loss on the target tasks. The adversarial gradient information about the target task leads the defender's model to have a poor performance on the target task, while seemingly higher performance on the new tasks.

We limit the behavior of the adversary such that it is not permitted to insert the samples from the target task into the non-target tasks nor change the label of those samples. The goal of the proposed attack is to find an adversarial subset $\tilde{\mathcal{D}}^{adv}_{\tau+1}$ of the current task $\mathcal{D}_{\tau+1}$ with perturbed data points $X^{adv}_{\tau+1}$ and clean labels $Y_{\tau+1}$. Then the gradient updates can be expressed as:

$$\theta_{\tau+1} = \theta_\tau - \eta \nabla_\theta \mathcal{L}_{\theta_\tau}(\mathcal{D}_{\tau+1} \cup \tilde{\mathcal{D}}^{adv}_{\tau+1}) \tag{8}$$
$$= \theta_\tau - \eta \nabla_\theta \mathcal{L}(f(X_{\tau+1}; \theta_\tau), Y_{\tau+1}) - \tag{9}$$
$$\eta \nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+1}; \theta_\tau), Y_{\tau+1}), \tag{10}$$

where we notice that difference between Equations 5 and 8 is the malicious gradients. If the gradients on adversarial subset $\tilde{\mathcal{D}}^{adv}_{\tau+1}$ on the defender model are close to the gradients on the label-flipped subset $\tilde{\mathcal{D}}^{adv}_\tau$ then $\tilde{\mathcal{D}}^{adv}_{\tau+1}$ will have the same poison effect as $\tilde{\mathcal{D}}^{adv}_\tau$ for making the model forget the knowledge on the target set $\mathcal{D}_\tau$. Therefore, the adversary's goal is to find the poison data $X^{adv}_{\tau+1}$ to meet the following condition:

$$\nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+1}; \theta_\tau), Y_{\tau+1}) \approx \nabla_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y^{adv}_\tau). \tag{11}$$

The min-max optimization problem in Equation 2 can be transformed into a simple optimization task by minimizing the distance between two gradients:

$$\min_{X^{adv}_{\tau+1}} \mathrm{d}(\nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+1}; \theta_\tau), Y_{\tau+1}), \tag{12}$$
$$\nabla_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y^{adv}_\tau)), \tag{13}$$

where d refers to the distance between two vectors. The cosine similarity is chosen as the distance function due to its being bounded in the interval $[-1, 1]$ and its property of magnitude irrelevant. Therefore, poison samples can be found by minimizing:

$$\min_{X^{adv}_{\tau+1}} - \frac{\left\langle \nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+1}; \theta_\tau), Y_{\tau+1}), \nabla_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y^{adv}_\tau) \right\rangle}{\left\| \nabla_\theta \mathcal{L}(f(X^{adv}_{\tau+1}; \theta_\tau), Y_{\tau+1}) \right\| \cdot \| \nabla_\theta \mathcal{L}(f(X_\tau; \theta_\tau), Y^{adv}_\tau) \|}. \tag{14}$$

The minimization object aims to create an approximation of the malicious gradients of the label-flipped subset $\tilde{\mathcal{D}}^{adv}_\tau$ by modifying the adversarial subset $\tilde{\mathcal{D}}^{adv}_{\tau+1}$. It is noticeable that we minimize the negative value of the cosine similarity since the larger cosine similarity refers to the closer distance of two vectors.

The malicious objective is optimized with the momentum variant of iterative fast gradient method (MI-FGSM) [34] as shown in Algorithm 2. The gradients of the defender model's parameter w.r.t to a batch of data sampled from the target task are first computed as the reference gradients. Each optimization step calculates the gradients of the defender's model w.r.t to the poisoned samples then calculated the gradients of the poisoned samples based on the malicious objective.

IV. Experiments

This section presents the experimental results for our proposed attacks. We identified two datasets to benchmark

	EWC					
Task	*Clean*	5% *poisoned*	10% *poisoned*	15% *poisoned*	20% *poisoned*	10% *label flipping*
1	66.73±2.72	**61.15±2.29**	**56.34±3.65**	**53.27±1.42**	**50.53±2.42**	4.79±0.51
2	40.55±1.05	40.25±0.8	38.2±1.0	39.21±0.68	40.15±0.73	38.9±0.81
3	82.11±0.47	82.57±0.3	82.89±0.49	83.26±0.33	82.77±0.55	82.22±0.89
	Online EWC					
Task	*Clean*	5% *poisoned*	10% *poisoned*	15% *poisoned*	20% *poisoned*	10% *label flipping*
1	69.43±2.35	**62.95±1.23**	**59.24±1.97**	**55.07±2.08**	**50.53±2.42**	2.57±0.35
2	40.88±1.18	39.53±1.0	40.3±0.29	40.65±0.95	40.15±0.73	43.37±1.19
3	81.16±0.52	82.86±0.37	82.23±0.26	82.46±0.51	82.77±0.55	81.56±0.41
	SI					
Task	*Clean*	5% *poisoned*	10% *poisoned*	15% *poisoned*	20% *poisoned*	10% *label flipping*
1	65.97±1.14	**63.32±1.15**	**60.76±1.5**	**54.62±1.95**	**53.65±2.81**	3.79±0.36
2	42.7±0.94	40.98±0.91	40.53±0.56	40.79±0.64	39.67±0.94	47.51±0.91
3	76.84±0.31	77.24±0.1	77.75±0.38	77.56±0.25	77.85±0.19	75.37±0.21

TABLE I

TEST ACCURACY ON MNIST FELLOWSHIP. THE BOLD TEXT REFERS TO THE RESULTS OBTAINED BY THE PROPOSED METHOD.

		TASK 1	TASK 2	TASK 3	TASK 4	TASK 5
EWC	*clean*	73.2±1.89	81.19±1.34	85.49±0.59	89.61±0.32	90.35±0.3
	5% *poisoned*	**69.07±1.33**	78.71±1.52	84.01±0.75	88.87±0.3	90.71±0.19
	10% *poisoned*	**68.07±2.11**	78.63±2.36	84.98±1.2	89.59±0.1	90.34±0.17
	15% *poisoned*	**66.52±0.89**	78.28±1.03	85.65±0.7	89.75±0.35	90.51±0.26
	20% *poisoned*	**67.12±1.27**	78.64±1.25	85.13±0.61	89.72±0.23	90.56±0.15
	10% *label flipping*	34.0±1.15	87.25±0.27	88.14±0.29	88.72±0.23	88.0±0.27
Online EWC	*clean*	67.5±1.01	77.67±1.03	86.56±0.28	91.03±0.06	90.09±0.24
	5% *poisoned*	**62.35±1.12**	76.74±0.79	86.11±0.36	90.7±0.24	90.09±0.24
	10% *poisoned*	**59.1±0.91**	75.68±0.74	86.25±0.48	91.41±0.15	90.35±0.23
	15% *poisoned*	**59.14±0.36**	75.31±0.44	86.65±0.35	90.67±0.14	89.26±0.25
	20% *poisoned*	**57.44±0.88**	74.5±0.76	86.3±0.48	90.94±0.26	89.8±0.47
	10% *label flipping*	24.98±0.8	82.51±0.52	87.51±0.46	90.01±0.2	87.27±0.31
SI	*clean*	59.02±1.9	69.64±2.0	82.9±0.59	89.94±0.26	88.07±0.41
	5% *poisoned*	**55.97±0.74**	68.02±0.99	82.81±0.86	90.12±0.13	88.44±0.21
	10% *poisoned*	**48.94±0.9**	62.16±0.7	79.97±0.53	89.71±0.06	89.12±0.15
	15% *poisoned*	**48.56±0.75**	64.21±0.87	81.59±0.56	90.22±0.1	88.62±0.24
	20% *poisoned*	**44.78±0.69**	61.23±0.49	81.15±0.76	89.97±0.37	88.01±0.16
	10% *label flipping*	27.03±0.82	71.72±0.35	85.05±0.36	89.84±0.19	86.8±0.22

TABLE II

TEST ACCURACY ON ROTATED MNIST. THE BOLD TEXT REFERS TO THE RESULTS OBTAINED BY THE PROPOSED METHOD.

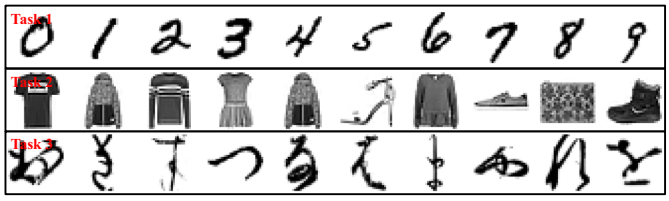


Fig. 1. MNIST fellowship dataset for task incremental learning. The first, second and third tasks are the MNIST, Fashion MNIST, and KMNIST datasets, respectively. Note that each task has the ten classes and the input size is 28x28.

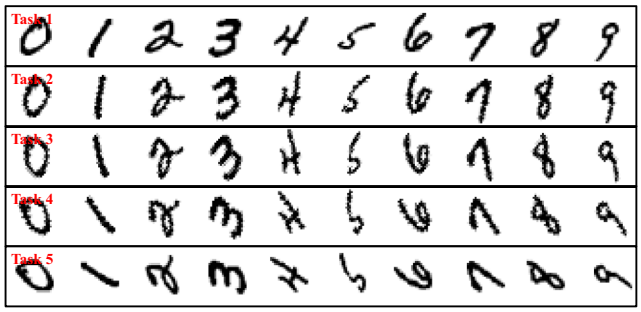


Fig. 2. Visualization of the Rotated MNIST dataset for domain incremental learning. Each task is generated from the original MNIST dataset; however, the tasks are rotated version of the original MNIST dataset.

the continual learning scenario with a multi-layer perceptron (MLP). Namely, we use the Rotation MNIST [17] for domain incremental learning, and MNIST Fellowship [35] for task

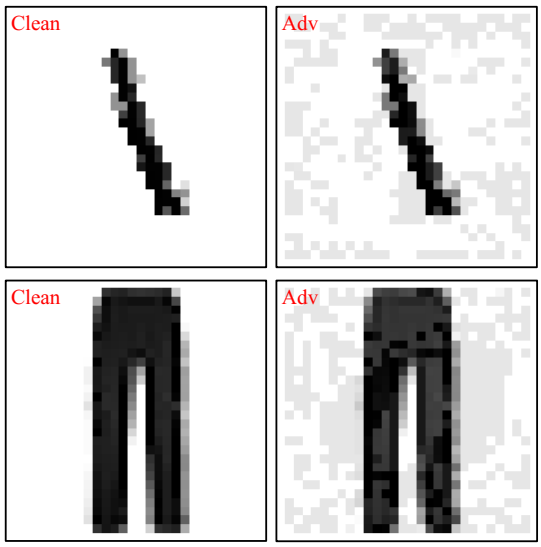


Fig. 3. Examples of adversarial samples generated using the proposed approach. All the samples are generated with an adversarial budget of $\epsilon = 0.1$. The first and second row are the clean and adversarial samples generated from the MNIST and Fashion-MNIST dataset, respectively.

incremental learning. The MLP for domain and task incremental learning has two hidden layers with ReLU activations and a soft-max output. The adversary and defender classifier has the same architecture. There are 400 and 1000 neurons in each hidden layer for domain and task incremental learning, respectively. In addition, we choose a single head configuration for the network, which means all tasks share the same classification layer [25]. All experiments are implemented using PyTorch and run on an NVIDIA Tesla V100 GPU.

A. Dataset

Rotation MNIST [17]: The rotation MNIST dataset consists of five different tasks, where each task is generated with a different rotation of the the images from the original task. Each task in Rotation MNIST is a 10-class classification problem where their labels are the corresponding digits. Thus, each subsequent task involves classification on the same ten digits. Figure 2 shows an example of the five tasks Rotation MNIST.

MNIST Fellowship [35]: The MNIST Fellowship is a combination of three variations of MNIST. The first task is the original MNIST dataset, which is classification of digits. The second task is Fashion MNIST dataset, which is classification of fashion products. Finally, the third task is KMNIST, which contains handwritten characters. Note that each of the tasks have ten classes and the input images are 28×28. Figure 1 shows an example of the three tasks in MNIST fellowship.

B. Results

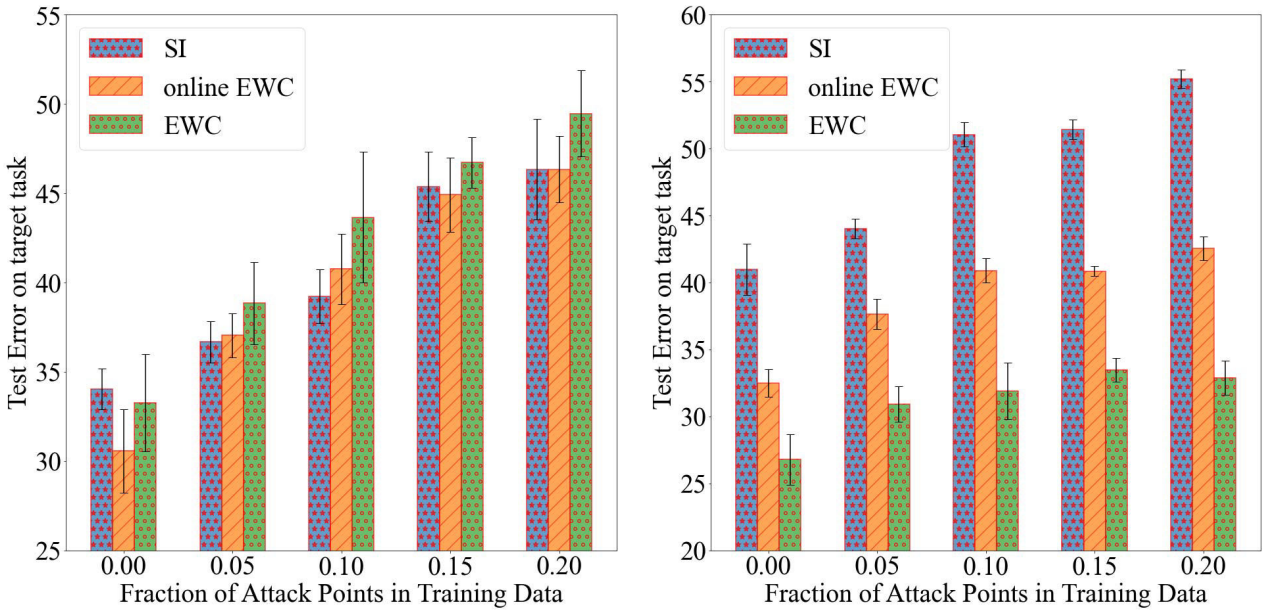
In our poisoning experiments, we chose the initial task as the target task, and poison the remainder of the tasks to degrade the performance on the target task. After the defender's network is train on each task, the adversary uses the defender's network to craft and inject the poisoned samples into the next task.

We first evaluate the poisoning attack against the task incremental learning. The three continual learning models are trained using the Adam optimizer [32] with a learning rate of 0.0001. Each task is trained for ten epochs with batch size 128. The regularization factors are 5 for SI and 5000 for both online EWC and EWC. We evaluate the accuracy on the clean validation sets for models trained on clean datasets, and $5 - 20\%$ poison samples of the training data of tasks two and three as shown in Table I. The results are averaged over five runs and we report the the standard errors with a 95% confidence interval. The poison samples are generated using Algorithm 2 for $T = 240$ iterations with momentum decay factor $\mu = 1.0$. The adversarial perturbations are restricted under ℓ_∞-norm for $\epsilon = 25.5/255$. The comparison on clean images and poison images under $\epsilon = 25.5/255$ can be shown in Figure 3. The step size is set as $4 \cdot \epsilon / T$ for attacking SI and online EWC and $2/255$ for attacking EWC. We also include the results obtained by inserting 10% data from the target task with flipped labels into the rest tasks.

The results for domain incremental learning are reported in Table II. The training configuration is the same as the task incremental learning, while the regularization factors are set as five for SI, 1000 for online EWC, and 750 for EWC. The poisoning setting used for domain incremental learning is the same as attacking against task incremental learning to ensure the method's generality. We also visualized the test errors on validation data of the target task and the fraction of poison samples in the non-target training data in more straightforward error bars as shown in Figure 4. As the poisoning ratio increases, the error rates of the models on the target task show an upward trend.

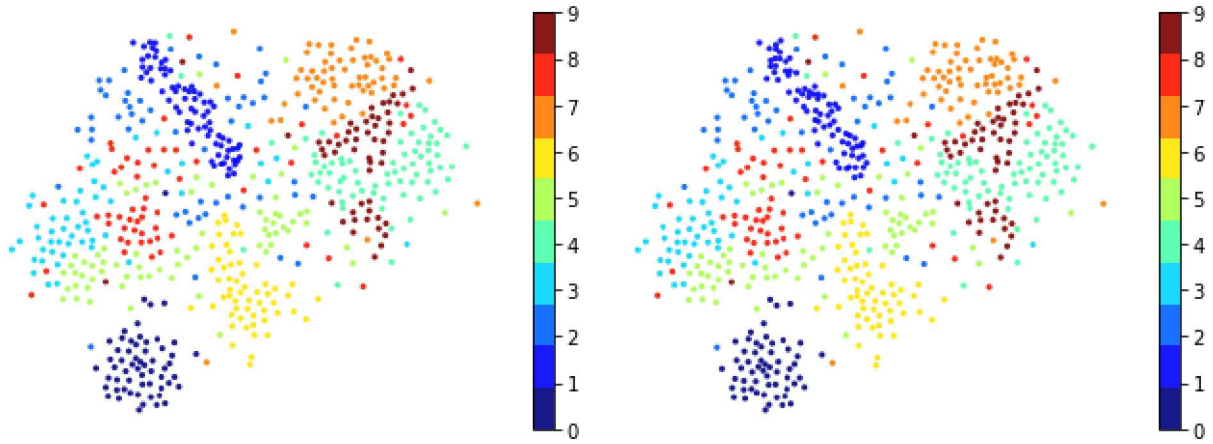
The observations reveal the vulnerability of the continual learning algorithms as the performance degrades as the adversarial samples are injected into the training dataset of the defender. The task incremental learning scenario shows the same vulnerability to the poisoning samples; however, in the domain incremental learning setting, we notice that the EWC is more robust than SI and online EWC. We hypothesize that the robustness of EWC in domain incremental learning arises from the separate Fisher information matrix calculated for each task that EWC can avoid the model parameters moving too far away from the optimal parameters of the target tasks.

The results show that the samples crafted to target forgetting are effective at reducing the performance; however, we need to show that the samples are difficult to detect. We demonstrate the stealthiness of the samples by visualizing the data using t-distributed stochastic neighbor embeddings (t-SNE) [36]. The benign samples and poison samples are mapped into a two-dimensional space, as shown in Figure 5. The poison samples are close to the benign samples in the low dimensional manifold, however, they are able to significantly force the defender to forget the learned knowledge. In addition to the adversarial samples being difficult to detect, we can also show that generating the samples has very little overhead. During the poisoning procedure, a batch poisoning strategy is adopted that for each run of Algorithm 2, 500 images are sampled as



(a) MNIST Fellowship (b) Rotated MNIST

Fig. 4. Performance of EWC, EWC (Online) and SI on the MNIST Fellowship and Rotated MNIST dataset with different levels of poisoning data added into the training set. The results are reported as the error and the error bars represent a 95% confidence interval. Note that the key finding here is not to determine if one algorithm is performing better or worse than another, rather, we show that the proposed targeted forgetting increases the error on the targeted task.



(a) Benign Samples (b) Adversarial Samples

Fig. 5. t-SNE visualization (a) benign and (b) adversarial samples generated from the MNIST dataset. For each sample in (a), we generate an adversarial sample using the proposed algorithm and are shown in (b). One of the key findings is that the adversarial samples are extremely close the the benign samples on the low dimensional manifold; however, these samples can significantly degrade the performance of the continual learning models. Thus, the performance reduction is observed in the tables, and this figure shows that the adversarial samples are difficulty to detect.

a single batch from the training data to be poisoned. Based on our experiments, one batch of 500 poison samples costs around 1.8s on GPU. This low time complexity enables our approach to be practical in continual learning settings where the model is learning from steaming data.

V. Ethical Considerations

The experiments conducted in this work include the artificially produced catastrophic forgetting of continual learning algorithms. It is worth noting that concept forgetting can be valuable in many incremental learning tasks where the concepts change over time; however, there are ethical implications of the proposed approach because the adversarial data can be used to target forgetting in a way that violates fairness in machine learning [37]. This notion of fairness is particularly important when we take into account the concepts or tasks that are targeted for forgetting. Therefore, we urge that care be taken when these models and techniques are used in practice, and the experiments in this work are limited specific types of concept forgetting.

VI. Conclusion

In summary, this work proposed a *task targeted poisoning attack* against continual learning algorithms that aim to degrade the performance of the neural network on specific learned tasks. We demonstrate the vulnerability to data poisoning attacks of three commonly-used regularization-based continual learning approaches. Further, we have shown the continual learning approaches vulnerable to data poisoning attacks in both task and domain incremental learning scenarios. The presented method starts from naïve label flipping attacks and then derives less detectable clean label attacks by approximation of flipped label gradients. The poison samples generated by the proposed method inject misinformation into the neural network, making the network learn a false memory about the target task. While data poisoning attacks have been widely studied in offline learning paradigms under i.i.d assumption, the data poisoning methodology in continual learning among non-i.i.d data still needs to be focused on. The primary purpose of this paper is to raise the community's awareness to focus on continual learning in adversarial and malicious environments. Our future work includes defense mechanisms to reduce the impact of adversarial attacks against continual learning algorithms.

Acknowledgements

This work was supported by grants from the Department of Energy #DE-NA0003946, and the National Science Foundation CAREER #1943552.

References

[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[2] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Adaptive strategies for learning in nonstationary environments: a survey," *Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[3] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, no. 1, pp. 17–61, 1988.

[4] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 4, pp. 497–508, 2001.

[5] D. Schwartz, T. W. Sawyer, N. Thurston, J. Barton, and G. Ditzler, "Early Ovarian Cancer Detection in Mice," *Neural Computing and Applications*, 2021.

[6] P. Ruvolo and E. Eaton, "{ELLA}: An efficient lifelong learning algorithm," in *International Conference on Machine Learning*, 2013.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2014.

[8] K. Sadeghi, A. Banerjee, and S. Gupta, "A System-Driven Taxonomy of Attacks and Defenses in Adversarial Machine Learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 450–467, 2020.

[9] C. Frederickson, M. Moore, G. Dawson, and R. Polikar, "Attack Strength vs. Detectability Dilemma in Adversarial Machine Learning," in *IEEE/INNS International Joint Conference on Neural Networks*, 2018.

[10] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[11] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," *arXiv preprint arXiv:2009.02276*, 2020.

[12] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *arXiv preprint arXiv:1804.00792*, 2018.

[13] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7614–7623.

[14] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 27–38.

[15] M. Umer, G. Dawson, and R. Polikar, "Targeted forgetting and false memory formation in continual learners through adversarial backdoor attacks," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[16] M. Umer and R. Polikar, "Adversarial targeted forgetting in regularization and generative based continual learning models," *arXiv preprint arXiv:2102.08355*, 2021.

[17] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, pp. 6467–6476, 2017.

[18] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[19] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *arXiv preprint arXiv:1705.08690*, 2017.

[20] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[21] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3366–3375.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[23] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4528–4537.

[24] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[25] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[28] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[30] H. Huang, X. Ma, S. M. Erfani, J. Bailey, and Y. Wang, "Unlearnable examples: Making personal data unexploitable," *arXiv preprint arXiv:2101.04898*, 2021.

[31] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[34] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[35] A. Douillard and T. Lesort, "Continuum: Simple management of complex continual learning scenarios," 2021.

[36] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

[37] S. Verma and J. Rubin, "Fairness definitions explained," in *Proceedings of the International Workshop on Software Fairness*. New York, NY, USA: ACM, 5 2018.