CONTINUAL LEARNING AND PRIVATE UNLEARNING

Bo Liu, Qiang Liu, Peter Stone
Department of Computer Science
The University of Texas at Austin
{bliu,lqiang,pstone}@cs.utexas.edu

ABSTRACT

As intelligent agents become autonomous over longer periods of time, they may eventually become lifelong counterparts to specific people. If so, it may be common for a user to want the agent to master a task temporarily but later on to forget the task due to privacy concerns. However enabling an agent to *forget privately* what the user specified without degrading the rest of the learned knowledge is a challenging problem. With the aim of addressing this challenge, this paper formalizes this continual learning and private unlearning (CLPU) problem. The paper further introduces a straightforward but exactly private solution, CLPU-DER++, as the first step towards solving the CLPU problem, along with a set of carefully designed benchmark problems to evaluate the effectiveness of the proposed solution. The code is available at https://github.com/Cranial-XIX/Continual-Learning-Private-Unlearning.

1 INTRODUCTION

Continual learning (CL) studies how an intelligent agent can learn continually over a sequence of tasks. In particular, when the agent is learning a new task, it is generally assumed that it loses access to data from previous tasks. As a result, the goal of a successful CL algorithm is to *forget* as little as possible about previous tasks while maximally adapting past knowledge to help learn the new task.

As deep learning has become increasingly popular, it has become generally known that straightforwardly applying stochastic gradient descent (SGD) on deep architectures when learning over a sequence of tasks leads to the so-called *catastrophic forgetting* phenomenon (French, 1999), i.e., the network forgets much of what it learned previously when learning new knowledge. Thus, much CL research has focused on developing methods to mitigate forgetting. However, forgetting is *not* always bad. Besides the fact that graceful forgetting—the process of deliberately compressing useful knowledge or removing useless knowledge—can help abstract learned knowledge and leave more "space" for learning new knowledge (Bjork & Bjork, 2019), we posit that it may also become common for an agent to be required to completely remove any trace of having learned a specific task.

For example, consider a robot manufacturing company that produces service robots, whose system is continually updated by learning *novel* skills on the data collected from its customers' daily lives. From time to time, the company may be asked to expunge previously learned behaviors and/or knowledge about specific tasks that are found to raise potential fairness (Mehrabi et al., 2021), privacy or security issues (Bae et al., 2018). Looking further into the future, consider another situation in which a person is undergoing a medical treatment plan and requests that their service robot learns to assist with the treatment. However, after having recovered, when a friend is about to visit, the person may not want the robot to exhibit any evidence of their previous medical treatment. In this case, the person would like to be able to request that the robot privately remove all knowledge of the treatment plan without impairing other unrelated knowledge it may have acquired during (or before or after) the time of the treatment. Both of the above situations indicate that as personalized models that lifelong learn with and about humans become commonplace, it is important for these models to carefully unlearn knowledge when necessary. This leads to the problem of machine unlearning (MU) (Cao & Yang, 2015; Bourtoule et al., 2019). But to the authors knowledge, MU has not yet been well studied in the continual learning setting where the underlying data distribution can shift over time.

Note that even though catastrophic forgetting often happens naturally with rich parametric models such as deep neural networks, it might not be sufficient because 1) the user may want the agent to unlearn immediately (instead of unlearning over time) and 2) the unlearning must happen privately, meaning that after forgetting, it must not be possible to retrieve any information pertaining to the task, or even detect that the task has been previously learned.

With this motivation in mind, in this work, we present a novel but general CL problem setting where for each task, besides providing the task data, the user additionally provides a learning instruction indicating whether they want the

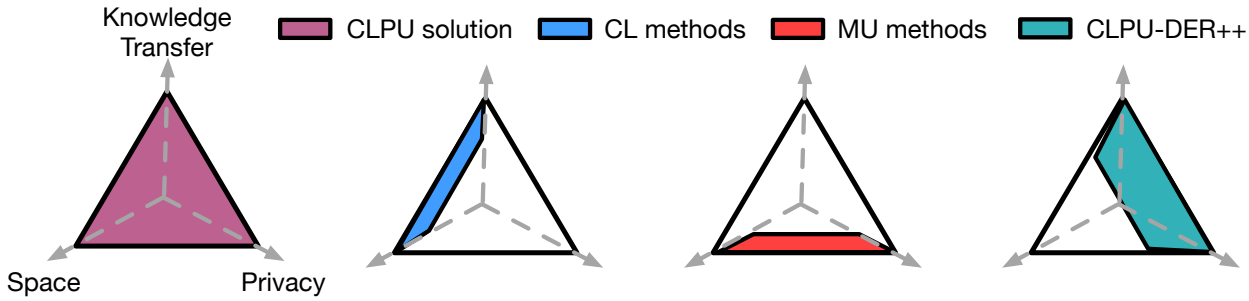

Figure 1: The CLPU problem has the Pareto front formed by good knowledge transfer ability, small model space, and no privacy leak. The ideal solution to CLPU achieves all of them simultaneously. We visualize what existing continual learning (CL) methods and machine unlearning (MU) methods achieve on the Pareto front above. CLPU-DER++ represents an initial CLPU algorithm that achieves exact unlearning and good continual learning performance, in exchange for using model space.

agent to learn and remember the task permanently, to temporarily learn the task such that later on it will either forget or permanently remember it, or to forget a certain task completely and privately. We call this novel problem *continual learning and private unlearning* (CLPU). To the best of our knowledge, only one previous paper discusses a similar problem setting pertaining to selective forgetting in continual learning (Shibata et al., 2021). However, the problem in that paper is different from CLPU as it defines forgetting as maximally degrading the performance on a task. As discussed in Sec. 5, this requirement is not privacy-preserving and can potentially leak information (e.g., that the task has been previously learned).

To address CLPU, we propose a straightforward but exact method, named CLPU-DER++, based on both the dynamic architecture approach (e.g. Rusu et al., 2016) and the rehearsal approach (e.g. Robins, 1995) from the CL literature. Furthermore, we design a set of benchmark tasks along with novel evaluation metrics for evaluating any CLPU methods. To summarize, our main contributions are:

- Formulating the continual learning and private unlearning (CLPU) problem.
- Presenting an initial solution, CLPU-DER++, to CLPU that achieves exact unlearning, and demonstrating its effectiveness on a novel set of benchmarks designed for CLPU.

2 Related Work

In this section, a brief review of continual learning and machine unlearning is provided. The relationship between CLPU and previous literature is summarized in Fig. 1.

Continual Learning Continual learning (CL) assumes a learning agent learns continually over a sequence of tasks and in general the agent loses access to previous data when learning new tasks. Due to its generality, CL has been applied to a variety of areas including computer vision (e.g. Kirkpatrick et al., 2017), reinforcement learning (e.g. Kirkpatrick et al., 2017; Riemer et al., 2018), natural language processing (e.g. Biesialska et al., 2020), and robotics (e.g. Liu et al., 2021). There exist three main approaches towards continual learning. 1) Dynamic architecture approaches study how to carefully and gradually expand the learning model to incorporate the learning of new knowledge (Rusu et al., 2016; Yoon et al., 2017; Mallya et al., 2018; Rosenfeld & Tsotsos, 2018; Mallya & Lazebnik, 2018; Hung et al., 2019b;a; Wu et al., 2020a). 2) Regularization-based methods design a regularization objective that prevents the model parameter deviating too much from the previously learned model(s) (Kirkpatrick et al., 2017; Chaudhry et al., 2018a; Schwarz et al., 2018; Aljundi et al., 2019). 3) Rehearsal methods save exemplar raw data, called episodic memory, from previously learned tasks. When learning new tasks, these methods simultaneously learn on the new task and rehearse on episodic memories to retain past knowledge (Chaudhry et al., 2019; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018b; Buzzega et al., 2020). Other than saving the raw data points, pseudo-rehearsal like training a generative model to replay past experience is also a popular approach (Shin et al., 2017). For a comprehensive survey of existing continual learning methods, we refer the reader to tow survey papers (Van de Ven & Tolias, 2019; Delange et al., 2021). In contrast to existing CL methods that focus on reducing forgetting, the CLPU problem requires the agent to deliberately forget a particular task upon request, while minimally influencing other knowledge.

Machine Unlearning Machine unlearning (MU) studies how to remove the effect of a specific training sample on a learning model per a user's request (Cao & Yang, 2015). The most straightforward approach is to retrain the model on all data except the portion that has been removed, but this approach is in general impractical if the entire training set is large. To this end, typical MU approaches consider training multiple models on different shards of data so that unlearning only requires retraining a specific model on part of the dataset (Bourtoule et al., 2019), or storing learned model parameters and their gradients for rapid retraining (Wu et al., 2020b). There is also research focusing on MU with specific model or problem assumptions, such as linear models (Guo et al., 2019), random forests (Brophy & Lowd, 2021), or k-means(Ginart et al., 2019). Based on differential privacy, Golatkar et al. introduced "scrubbing" that removes information from the weights of deep networks based on the Fisher Information Matrix (Golatkar et al., 2020). Mixed-Linear Forgetting proposes a tractable optimization problem by lineary approximating the amount of change in weights due to the addition of any training data (Golatkar et al., 2021). The above methods all consider the MU problem in general where the preserved dataset (e.g., data except the removed ones) is available, which is not the case in continual learning. Recently, a particularly relevant study first considers MU in the context of continual learning (Shibata et al., 2021). However, their problem definition aims to make the model predict as wrongly as possible on the removed data, which does not in general protect the user's privacy. For instance, if the agent has learned task B that helps improve prediction on task A, which the agent is asked to forget, then completely random prediction on task A also reveals that the agent has learned on it before but asked to unlearn it later. In fact, as we observe from experimental results (Sec. 5), simply decreasing the model's performance on removed data is not private.

3 Background

In this section, we present the notation, definitions, and necessary background information to formalize CLPU.

3.1 Continual Learning

In continual learning (CL), an agent observes and learns K tasks in a sequence. In this work, we assume each task $k \in [K]$[1] is a supervised learning task with a loss function $\ell^k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, a training dataset $D^k = \{(x_1^{k,\text{train}}, y_1^{k,\text{train}}), \ldots, (x_n^{k,\text{train}}, y_n^{k,\text{train}})\}$ and a testing dataset $D^k_{\text{test}} = \{(x_1^{k,\text{test}}, y_1^{k,\text{test}}), \ldots, (x_{n'}^{k,\text{test}}, y_{n'}^{k,\text{test}})\}$. Here, (x, y) are the raw data and labels where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assume the agent adopts a model f_θ parameterized by $\theta \in \mathbb{R}^d$. For instance, for a classification task, $\text{softmax}(f_\theta)$ produces a probability distribution over $\mathcal{Y}$. Then ℓ^k evaluates how well f_θ predicts y given x. For instance, ℓ^k can be the standard cross-entropy loss for classification tasks (e.g., $\ell^k(x, y, f_\theta) = \log \sum_{y'} \exp\big(f_\theta(x)[y']\big) - f_\theta(x)[y]$).

On learning task k, the agent loses its access to $D^{<k} = \{D^1, \ldots, D^{k-1}\}$. After learning all K tasks, the agent's objective is to achieve low loss on all test datasets $\{D^k_{\text{test}}\}_{k\in[K]}$. Assume the agent learns with a model f that is parameterized by $\theta \in \mathbb{R}^m$. Denote the agent's model after learning task k as f_{θ^k}. Then, the overarching objective of a CL agent is to optimize

$$\min_{\theta^K} \frac{1}{K} \sum_{k\in[K]} \mathbb{E}_{(x,y)\sim D^k_{\text{test}}} \Big[\ell^k(x, y, f_{\theta^K})\Big]. \tag{1}$$

However, equation 1 is hard to directly optimize using gradient-based methods (e.g., Stochastic Gradient Descent (SGD)) because of the dependency of θ^K on all previous θ^k such that $k < K$. Therefore, alternatively, from an induction point of view, the objective can be decomposed into K objectives throughout the learning process. For learning the first task, the agent just optimizes the training loss ℓ^1 on the training dataset D^1 as in standard supervised learning. For any task $k > 1$, the agent is asked to achieve low loss on task k while maintaining its performance on previously learned tasks (Lopez-Paz & Ranzato, 2017):

$$\min_{\theta\in\mathbb{R}^m} \underbrace{\mathbb{E}_{(x,y)\sim D^k}\Big[\ell^k(x, y, f_\theta)\Big]}_{\text{performance on task } k} \quad \text{s.t.} \quad \forall \tau < k, \quad \underbrace{\mathbb{E}_{(x,y)\sim D^\tau}\Big[\ell^\tau(x, y, f_\theta) - \ell^\tau(x, y, f_{\theta^{(k-1)}})\Big]}_{\text{forgetting on task } \tau} \leq 0. \tag{2}$$

Note that here we also replace the testing loss by training loss as the agent is not assumed to have access to test data during training. In practice, when the underlying K tasks share the same loss function (e.g., they are all image classification tasks), which we assume for the rest of this paper, we can elide the superscript k in ℓ^k.

The main challenge for solving CL results from losing access to $D^{<k}$. Regularization-based methods assume all information learned from $D^{<k}$ is in θ^{k-1}. Thus they aim to minimize the training loss on D^k while ensuring that θ

[1] $[K]$ denotes $\{1, 2, \ldots, K\}$.

stays close to θ^{k-1}:

$$\min_{\theta \in \mathbb{R}^m} \mathbb{E}_{(x,y)\sim D^k}\Big[\ell(x, y, f_\theta)\Big] + \alpha D(\theta, \theta^{k-1}), \tag{3}$$

where α is a hyperparameter determining the strength of regularization and $D(\theta, \theta^{k-1})$ is a divergence measure that captures how similar θ is to θ^{k-1}. Another popular and empirically more effective approach in CL is the rehearsal-based approach. These methods allow the agent to store a small number of exemplar data points, known as the *episodic memory* $\{B^\tau\}_{\tau<k}$, for maintaining the agent's learned knowledge on previous tasks. In particular, B^τ stores $b^\tau \ll |D^\tau|$ i.i.d. sampled data points from D^τ, optionally with f_{θ^τ}'s final layer output (a.k.a. the logits):

$$B^\tau = \left\{ \big(x_i, y_i, h_i = f_{\theta^\tau}(x_i)\big) \;\middle|\; (x_i, y_i) \overset{\text{i.i.d}}{\sim} D^\tau \right\}_{1\le i\le b^\tau}.$$

As a result, the general objective of rehearsal-based methods is

$$\min_{\theta \in \mathbb{R}^m} \frac{\beta}{|D^k|} \sum_{(x,y)\sim D^k} \ell(x, y, f_\theta) + (1-\beta) \sum_{\tau<k} \frac{1}{|B^\tau|} \sum_{(x',y',h')\sim B^\tau} \hat{\ell}(x', y', h', f_\theta), \tag{4}$$

where $\beta > 0$ is a hyper-parameter that weights the trade-off between learning new and preserving old knowledge. $\hat{\ell}$ can be the standard cross-entropy loss ℓ, the knowledge distillation loss $\ell_{\text{distill}}(x, h, f_\theta) = D_{\text{KL}}\big(\text{softmax}(f_\theta(x)) \;||\; h\big)$, or the mean-square-error between the predicted and saved logits $\ell_{\text{mse}}(x, h, f_\theta) = \frac{1}{2}||f_\theta(x) - h||_2^2$, which has shown strong performance at preserving past knowledge (Buzzega et al., 2020). Specifically, for the Dark Experience Replay++ (DER++) method (Buzzega et al., 2020), $\hat{\ell}$ is a linear combination of the standard cross entropy loss and mean-square-error loss: $\hat{\ell}(x', y', h', f_\theta) = \alpha_1 \ell(x', y', f_\theta) + \alpha_2 \ell_{\text{mse}}(x', h', f_\theta)$.

3.2 Machine Unlearning

Unlike continual learning which studies learning over sequential tasks, contemporary research in machine unlearning (MU) mainly focuses on single-task learning (Cao & Yang, 2015). In particular, MU is often studied in the context of supervised learning, though extensions to other types of learning are relatively straightforward. Under the standard supervised learning setting, the agent is given a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. The agent applies a (stochastic) learning algorithm A on D to learn a model f_θ parameterized by θ, such that f_θ achieves low empirical loss (e.g., $\frac{1}{n}\sum_{i=1}^n \ell(x_i, y_i, f_\theta)$ is small). Denote $A(D)$ as the distribution over the resulting model parameters θ when A is applied on D.

A user can then request that the agent unlearn part of the dataset, which we call the forget set, $D_f \subset D$. Denote $D_r = D \setminus D_f$ as the retained dataset. Machine unlearning (MU) aims to find an unlearning algorithm R_A that returns a model $\theta \sim R_A(D, A(D), D_f)$, which possesses no information about D_f while performing well on D_r. In general, it is usually assumed that $|D_f| \ll |D|$, otherwise one can directly retrain a model on D_r. If the unlearned model from $R_A(D, A(D), D_f)$ has no information about D_f, an adversary cannot differentiate the model after unlearning from a model that is retrained on D_r, and we say that (A, R_A) achieves *exact unlearning*. The formal definitions are as follows.

Definition 3.1 (Exact Unlearning)**.** *A pair of learning and unlearning algorithms (A, R_A) achieve exact unlearning if*

$$\forall D,\ D_f \subset D,\ \ A(D_r) =_d R_A(D, A(D), D_f),\ \ \textit{where}\ \ D_r = D \setminus D_f. \tag{5}$$

Here $X =_d Y$ means X and Y share the same distribution.

The definition of exact unlearning is quite restrictive and therefore can be hard to achieve in practice. As a relaxation, Ginart et al. (2019) proposed the following definition of *approximate unlearning*.

Definition 3.2 (Approximate δ-Unlearning)**.** *(A, R_A) satisfies δ-unlearning if*

$$\forall D,\ D_f \subset D,\ \textit{and}\ E \subseteq \mathbb{R}^d,\ \ P\big(R_A(D, A(D), D_f) \in E\big) \le \delta^{-1} P\big(A(D_r) \in E\big). \tag{6}$$

These definitions are based on the assumption that the adversary can directly access the model parameters θ and therefore the definitions are based on the distribution over θ. A more general assumption is that the adversary can only access the model via an output function $O(\theta, D)$, where $O : \Theta \times \mathcal{X} \longrightarrow \mathcal{O}$ and $\mathcal{X}$ denotes the space of input data. For instance, if $O(\theta, x) = f_\theta(x)$, it means the adversary only has access to the agent's prediction on any data point x. In that case, we can modify the definitions by replacing $R_A(\cdot)$ by $O \circ R_A(\cdot)$ and $A(\cdot)$ by $O \circ A(\cdot)$.

Remarks

- **MU depends on both A and R_A.** As R_A highly depends on the learning algorithm A, MU focuses on the design of both. But note that purely achieving exact unlearning without considering the model's performance is meaningless. For example, one can achieve exact unlearning trivially if A yields a constant mapping and R_A is an identity mapping. Therefore, the challenge is to maintain $R_A(D, A(D), D_f)$'s performance on D_r, while unlearning exactly.
- **MU differs from differential privacy (DP):** ϵ-DP does not divide the data into D_f and D_r and requires that *no* individual data point can significantly influence the model's prediction. But in MU (with exact unlearning), it is required that any data point $x \in D_f$ has *zero* influence on the model's prediction after the unlearning, with no restrictions on the effects of data $x \in D_r$.
- **δ-Unlearning is asymmetric** The above definitions implicitly assume that $P(\theta \mid A(D_r)) = 0 \implies P(\theta \mid R_A(D, A(D), D_f)) = 0$. However, we do not assume the converse, meaning that it is permissible for R_A to not generate models that could have been generated by $A(D_r)$.

4 Problem and Method

We start this section with a formal introduction to the continual learning and private unlearning (CLPU) problem. Then we introduce a straightforward solution to CLPU that achieves exact unlearning by saving extra models, thus sacrificing some space complexity compared to using a fixed-sized model throughout learning.

4.1 CLPU: The Continual Learning and Private Unlearning problem

In Continual Learning and Private Unlearning (CLPU), an agent receives T requests from the user sequentially and is asked to learn from a pool of K tasks in total. The t-th request $\mathcal{R}^t$ is a tuple $\mathcal{R}^t = (I^t, D^t, \rho^t)$. Here, $I^t \in [K]$ is the task ID, indicating the current task of interest. D^t is either the training dataset $\{(x_i, y_i)\}_{i=1}^{|D^t|}$ or an empty set $\emptyset$ depending on what ρ^t is. $\rho^t \in \{\mathbf{R}, \mathbf{T}, \mathbf{F}\}$ is a learning instruction:

- $\rho^t = \mathbf{R}$: the user asks the agent to learn on task i permanently.
- $\rho^t = \mathbf{T}$: the user asks the agent to temporarily learn on task i, which can be forgot in the future.
- $\rho^t = \mathbf{F}$: the user asks the agent to forget task i with exact unlearning.

The agent keeps a dictionary Ψ^t of the learned tasks' statuses, which, given $\mathcal{R}^t$, is updated by:

$$\begin{cases} \Psi^t[I^t] = (D^t, \rho^t) & \text{if } \rho^t \in \{\mathbf{R}, \mathbf{T}\} \\ \Psi^t \leftarrow \Psi^{(t-1)} \setminus \{I^t\} & \text{if } \rho^t = \mathbf{F}. \end{cases} \tag{7}$$

Here, $\Psi \setminus I$ indicates the removal of the key I as well as its corresponding values (D, ρ) from Ψ. If $I^t \in \Psi^{(t-1)}$ and $\rho^t = \mathbf{R}$, the agent is to fully memorize a task that has previously been temporarily learned with instruction $\mathbf{T}$. In both this case and the case when $\rho^t = \mathbf{F}$, we assume $D^t = \emptyset$ as there is no need for the user to provide the dataset for the same task twice.

Now we are ready to present the formulation of CLPU. Denote all requests up to the $(t-1)$-th request as $\mathcal{R}^{<t} = [\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^{(t-1)}]$.[2] A CLPU solution consists of a continual learning algorithm A and an unlearning algorithm R_A. Let f_θ be the learning model parameterized by $\theta \in \mathbb{R}^m$ and θ^t denote the model parameter after processing the t-th request. Then both A and R_A map the previous model parameters and the current request to updated model parameters. In particular, we have the following recursion:

$$\begin{cases} \theta^t \sim A(\theta^{(t-1)}, \mathcal{R}^t = (I^t, D^t, \rho^t)) & \text{if } \rho^t \in \{\mathbf{R}, \mathbf{T}\}, \\ \theta^t \sim R_A(\theta^{(t-1)}, \mathcal{R}^t = (I^t, D^t, \rho^t)) & \text{if } \rho^t = \mathbf{F}. \end{cases} \tag{8}$$

For simplicity of notation, if all requests from $\mathcal{R}^s$ to $\mathcal{R}^t$ ($\mathcal{R}^{s:t}$ for short), have $\rho \in \{\mathbf{R}, \mathbf{T}\}$, then we denote $\theta^t \sim A(\theta^{s-1}, \mathcal{R}^{s:t})$. Additionally, denote $[\tau \in \Psi^{(t-1)}]$ as all tasks with $\rho \in \{\mathbf{R}, \mathbf{T}\}$ that have been observed and not removed up to the $(t-1)$-th request. The objective of a CLPU agent when processing $\mathcal{R}^t$ is for A and R_A to output

[2] We use list notation $[\mathcal{R}^1, \mathcal{R}^2, \ldots]$ to indicate that the ordering matters.

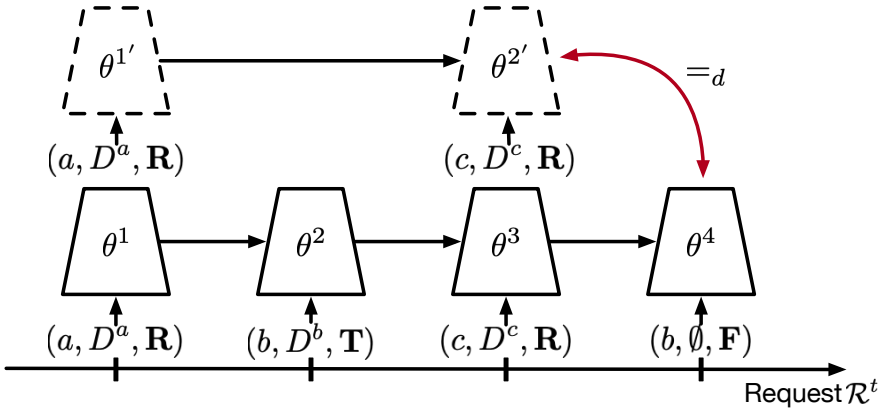

Figure 2: An illustration of the Continual Learning and private unlearning (CLPU) problem setting. After the agent has temporarily learned on task b with data D^b, if the agent is later requested to unlearn task b, the unlearned model parameters θ^4 should be indistinguishable from $\theta^{2'}$ in distribution as if the agent has never learned on task b. Except for the unlearn requests, the agent should perform continual learning over the remaining sequence of tasks.

θ^t with the following properties:

$$\theta^t \sim \begin{cases} \arg\min_\theta \mathbb{E}_{(x,y)\sim D^t}\Big[\ell(x,y,f_\theta)\Big] \ \text{ s.t} \\ \qquad \forall \tau \in \Psi^{(t-1)},\ \mathbb{E}_{(x,y)\sim D^\tau}\Big[\ell(x,y,f_\theta) - \ell(x,y,f_{\theta^{(\tau-1)}})\Big] \le 0 & \text{if } \rho^t \in \{\mathbf{R}, \mathbf{T}\}, \\ R_A(\theta^{(t-1)}, \mathcal{R}^t) \ \text{ s.t. } \ \mathcal{D}\Big(R_A(\theta^{(t-1)}, \mathcal{R}^t) \ \Big|\Big| \ A(\theta^0, \mathcal{R}^{[\tau\in\Psi^{(t-1)}]})\Big) = 0 & \text{if } \rho^t = \mathbf{F}. \end{cases} \tag{9}$$

Here, $\mathcal{D}(A \,||\, B)$ is a distance between distributions A and B. In other words, in the first case when $\rho^t \in \{\mathbf{R}, \mathbf{T}\}$, the expected loss cannot get any worse for any previously learned (but not forgotten) task. In the second case when $\rho^t = \mathbf{F}$, then the unlearned model parameters cannot be distinguished from the model parameters learned over the sequence of non-forgetting tasks with the divergence $\mathcal{D}$. The CLPU problem setting is illustrated in Fig. 2.

How does CLPU differ from CL and MU? 1) In CLPU, in addition to exhibiting knowledge transfer as in CL, the agent also needs to unlearn specific tasks while maintaining all knowledge unrelated to the forgotten tasks. 2) Unlike MU where generally the agent learns on i.i.d. samples from the entire dataset D, in CLPU the agent learns online over different tasks and hence the ordering of the sequence of tasks matters. In addition, CLPU does not in general assume the agent can keep all previous data, which makes the unlearning (or more specifically the retention of learned knowledge) more difficult.

4.2 CLPU-DER++: An Initial CLPU Algorithm

In this section, we present a straightforward method to CLPU, named CLPU-DER++ as it adapts the DER++ method (Buzzega et al., 2020) to the CLPU problem. The CLPU-DER++ method achieves exact unlearning upon request and learns continually over a sequence of tasks otherwise.

Inspired by Sharded, Isolated, Sliced, and Aggregated training (SISA) (Bourtoule et al., 2019), for each task with $\rho = \mathbf{T}$, the CLPU-DER++ agent creates an isolated temporary network with parameters $\hat{\theta}$. Based on the subsequent learning instruction for the same task, the agent either removes this isolated model ($\mathbb{F}$) or merges it with the main model ($\mathbb{R}$).

Specifically, we assume the agent maintains a main model with parameters θ_{main} and a set N of temporary models. In other words, $\theta = \{\theta_{\text{main}}\} \cup N$. Upon the t-th request $\mathcal{R}^t = (I^t, D^t, \rho^t)$, there are four possible cases. **1)** If $\rho^t = \mathbf{R}$ and $I^t \notin \Psi^{(t-1)}$, then this is the first time the agent has observed task I^t, so the agent then performs conventional CL using Dark Experience Replay++ (DER++) (Buzzega et al., 2020) and updates the main model parameters to θ^t_{main}. **2)** If $\rho^t = \mathbf{T}$, in order to benefit from prior learning experience, the agent initializes an isolated model with parameters $\hat{\theta}$ copied from $\theta^{(t-1)}_{\text{main}}$, then directly performs SGD update on $\hat{\theta}$ using the dataset D^t and includes the updated network to N (e.g., $N \leftarrow N \cup \hat{\theta}^{I^t}$). In both cases, the agent stores episodic memory B^{I^t} (See Sec. 3.1) for the task I^t. **3)** If $\rho^t = \mathbf{R}$ and $I^t \in \Psi^{(t-1)}$, this means the agent has previously learned on task I^t with a temporary network $\hat{\theta}^{I^t}$. Then

Algorithm 1 Continual Learning and private unlearning - Dark Experience Replay++ (CLPU-DER++)

Input: Initial main model parameters θ^0_{main} and temporary networks $N = \emptyset$, initial task status dictionary $\Psi^0 = \emptyset$, the total number of user requests T, and memory sizes $\{b^t\}_{t=1}^T$.
for $t = 1 : T$ **do**
  Receive request $\mathcal{R}^t = (I^t, D^t, \rho^t)$.
  Update Ψ^t by

$$\begin{cases} \Psi^t[I^t] = (D^t, \rho^t) & \text{if } \rho^t \in \{\mathbf{R}, \mathbf{T}\} \\ \Psi^t \leftarrow \Psi^{(t-1)} \setminus \{I^t\} & \text{if } \rho^t = \mathbf{F}. \end{cases}$$

  Case I: $\rho^t = \mathbf{R}$ **and** $I^t \notin \Psi^{(t-1)}$
    Perform H steps of SGD from $\theta^{(t-1)}_{\text{main}}$ by optimizing:

$$\theta^t_{\text{main}} = \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{|D^t|} \sum_{(x,y)\sim D^t} \ell(x, y, f_\theta) + \frac{1}{|\Psi^{(t-1)}|} \sum_{i \in \Psi^{(t-1)}} \frac{1}{|B^i|} \sum_{(x',h')\sim B^i} \ell_{\text{mse}}(x', h', f_\theta).$$

    Build the episodic memory B^{I^t}: $B^{I^t} = \left\{ \left(x_i, y_i, f_{\theta^t_{\text{main}}}(x_i)\right) \;\middle|\; (x_i, y_i) \overset{\text{i.i.d}}{\sim} D^t \right\}_{1 \le i \le b^t}$.
  Case II: $\rho^t = \mathbf{T}$
    Initialize $\hat{\theta}^{I^t}$ from $\theta^{(t-1)}_{\text{main}}$.
    Perform H steps of SGD on $\hat{\theta}^{I^t}$ by optimizing:

$$\hat{\theta}^{I^t} = \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{|D^t|} \sum_{(x,y)\sim D^t} \ell(x, y, f_\theta) + \frac{1}{|\Psi^{(t-1)}|} \sum_{i \in \Psi^{(t-1)}} \frac{1}{|B^i|} \sum_{(x',h')\sim B^i} \ell_{\text{mse}}(x', h', f_\theta).$$

    Store the temporary network: $N \leftarrow N \cup \{\hat{\theta}^{I^t}\}$.
    Build the episodic memory B^{I^t}: $B^{I^t} = \left\{ \left(x_i, y_i, f_{\theta^t_{\text{main}}}(x_i)\right) \;\middle|\; (x_i, y_i) \overset{\text{i.i.d}}{\sim} D^t \right\}_{1 \le i \le b^t}$.
  Case III: $\rho^t = \mathbf{R}$ and $I^t \in \Psi^{(t-1)}$
    Merge $\hat{\theta}^{I^t}$ back to θ^t_{main} by performing H step of SGD and optimize:

$$\theta^t = \arg\min_{\theta \in \mathbb{R}^m} \frac{1}{|B^{I^t}|} \sum_{(x,h)\sim B^{I^t}} \ell_{\text{mse}}(x, h, f_\theta) + \frac{1}{|\Psi^t|} \sum_{i \in \Psi^t} \frac{1}{|B^i|} \sum_{(x',h')\sim B^i} \ell_{\text{mse}}(x', h', f_\theta).$$

    Remove the temporary network: $N = N \setminus \{\hat{\theta}^{I^t}\}$.
  Case IV: $\rho^t = \mathbf{F}$
    Remove the temporary network: $N = N \setminus \{\hat{\theta}^{I^t}\}$.
end for

the agent merges the knowledge learned in $\hat{\theta}^{I^t}$ into θ^t_{main} to reduce space and encourage knowledge transfer. To do so, CLPU-DER++ performs knowledge distillation on the combined episodic memories from task I^t and the rest of the previously fully remembered tasks in Ψ^t. **4)** If $\rho^t = \mathbf{F}$, we simply remove the temporary network $\hat{\theta}^{I^t}$ from N. The details of CLPU-DER++ are presented in Alg. 1.

Remark CLPU-DER++ achieves *exact* unlearning (See Def. 3.1) by construction. For any task k that the agent has learned previously and then attempts to unlearn, the unlearn process only involves removing the relevant temporary model from N and the corresponding episodic memory: it does not influence the main model parameter θ_{main}. On the other hand, CLPU-DER++ achieves privacy at the expense of memory, as it stores a full extra model for each temporary learning task, which can be particularly important for large modern neural architectures.

5 Experimental Results

In this section, we first introduce the experiment setup and introduce how we form novel benchmarks by adapting conventional CL datasets for the CLPU problem. Then we introduce the evaluation metrics designed for measuring the agent's performance in terms of both continual learning and private unlearning. In the end, we present the evaluation results by comparing CLPU-DER++ against the following baseline methods: sequential learning (Seq), indepdent learning (Ind), Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Learning without Forgetting (LwF) (Li

& Hoiem, 2017) , Experience Replay (ER) (Chaudhry et al., 2019), Dark Experience Replay++ (DER++) (Buzzega et al., 2020), and Learning with Selective Forgetting (LSF) (Shibata et al., 2021). All the above baselines except LSF are state-of-the-art CL methods, but we adapt some of them for the CLPU setting. In particular, for sequential learning, the agent performs SGD directly over the sequence of tasks. For independent learning, the agent creates a new model for each new task, and removes a model if the user requests to unlearn the corresponding task. For ER and DER++, for an unlearning task, we remove the corresponding episodic memory and let the agent perform normal ER and DER++ updates on the remaining episodic memories and predict uniform distributions for the forgotten task to accelerate forgetting.

5.1 CLPU Experiment Setup

We consider four conventional CL benchmarks: rotation MNIST (rot-MNIST), permutation MNIST (perm-MNIST), split CIFAR-10 and split CIFAR-100. rot-MNIST and perm-MNIST datasets are formed by rotating the images and randomly permuting the pixels of the images, respectively, in the MNIST dataset. Each task is a 10-class classification task. Split CIFAR-10 and split CIFAR-100 are formed by treating the 10 classes in CIFAR-10 as five 2-class classification tasks, and the 100 classes in CIFAR-100 as five 20-class classification tasks. To be consistent, we build rot-MNIST and perm-MNIST also with 5 sequential tasks. Then, to adapt these datasets to the CLPU setting, we form the following sequence of requests:

$$\mathcal{R}^{1:8} = \big[(1, D^1, \boldsymbol{R}),\ (2, D^2, \boldsymbol{T}),\ (3, D^3, \boldsymbol{T}),\ (4, D^4, \boldsymbol{R}),\ (1, \emptyset, \boldsymbol{R}),\ (2, \emptyset, \boldsymbol{F}),\ (5, D^5, \boldsymbol{T}),\ (5, \emptyset, \boldsymbol{F})\big].$$

The corresponding sequence of requests that involve no unlearning is therefore

$$\hat{\mathcal{R}}^{1:4} = \big[(1, D^1, \boldsymbol{R}),\ (3, D^3, \boldsymbol{T}),\ (4, D^4, \boldsymbol{R}),\ (1, \emptyset, \boldsymbol{R})\big].$$

For all datasets, we use the SGD optimizer without momentum with 0.0005 weight decay. For all datasets, the learning rate is set to 0.01 and we perform 10 epochs of training for each task. When the agent is asked to unlearn a task, we also perform 10 epochs of the algorithm-specific unlearn updates. The implementations of the baseline methods are adapted from the open-source DER implementation.[3]

5.2 Evaluation Metrics

To evaluate a method on CLPU, we consider metrics both for continual learning and for private unlearning. To measure the method's performance on continual learning, we report the final average accuracy (ACC) of the model over all tasks that remain in the final task status dictionary Φ, as well as the forgetting measure (FM), which is the average drop in performance on each task, compared to the model's performance when the agent first learned these tasks. Note that all evaluations are done on holdout testing data $\{D^k_{\text{test}}\}$ for each task k. To be specific, denote a^t_s as the agent's prediction accuracy on task s's test dataset D^s_{test} after processing user's t-th request, then we define

$$\text{ACC} = \sum_{t=1}^{T} \sum_{s \in \Phi^t} a^t_s \quad \text{and} \quad \text{FM} = \sum_{t=1}^{T} \sum_{s \in \Phi^t} a^{\tau(s)}_s - a^t_s, \quad \text{where } \tau(s) = \arg\min_t \ (I^t = s). \tag{10}$$

In short, ACC measures how well the agent performs on the tasks with $\rho \in \{\mathbf{R}, \mathbf{T}\}$ after processing all requests. FM measures how much the agent forgets on the same set of tasks compared to when they were first learned.

In addition to the above two metrics for evaluating the continual learning performance, we also compare, on all tasks with $\rho = \mathbf{F}$, the divergence between the model's output distribution on that task after unlearning versus the distribution that would have resulted had the agent not learned the task. In other words, we use the output function $O(\theta, x) = f_\theta(x)$ because comparing the distributions of $f_\theta(\cdot)$ with different θs is more computationally efficient than directly comparing the distributions of different θs. Concretely, for all requests $\mathcal{R}^t$ such that $\rho^t = \mathbf{F}$, we measure how different f_{θ^t} is from $f_{\theta^{t'}}$, where $\theta^t \sim A(\theta^0, \mathcal{R}^{\leq t})$ and $\theta^{t'} \sim A(\theta^0, \mathcal{R}^{[\tau \in \Psi^t]})$. To measure the difference, we train c models using different random seeds on $\mathcal{R}^{\leq t}$ to get parameters $\{\theta^t_1, \ldots, \theta^t_c\}$, and similarly get c model parameters $\{\theta^{t'}_1, \ldots, \theta^{t'}_c\}$ by training on $\mathcal{R}^{[\tau \in \Psi^t]}$. After that, for each pair of models that are trained on $\mathcal{R}^{[\tau \in \Psi^t]}$, we calculate the in-group Jensen-Shannon (IJSD) distance between their outputs on the testing dataset $D^{I^t}_{\text{test}}$.[4] Similarly, for any model trained on $\mathcal{R}^{[\tau \in \Psi^t]}$ and any other model trained on $\mathcal{R}^{\leq t}$, we also calculate their output distributions' Jenson-Shannon distance, which we call the Across-group Jensen-Shannon Distance (AJSD). For the entire sequence of requests, we

[3] DER code from https://github.com/aimagelab/mammoth.

[4] We use Jensen-Shannon Distance because it is a symmetric divergence for comparing probability distributions.

	Perm-MNIST					
Method	ACC(↑)	FM(↓)	IJSD	AJSD	JS-ratio(↓)	IRR(↑)
Ind (Upper Bound)	95.59 ± 0.05	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.14	0.96
Seq	75.75 ± 2.44	19.97 ± 2.45	0.17 ± 0.05	0.92 ± 0.03	4.47	0.00
EWC	93.67 ± 0.25	0.45 ± 0.18	0.04 ± 0.01	0.65 ± 0.01	13.73	0.00
ER	91.83 ± 0.25	3.96 ± 0.24	0.11 ± 0.02	0.73 ± 0.01	5.92	0.00
LwF	79.09 ± 3.19	17.12 ± 3.23	0.08 ± 0.02	0.83 ± 0.03	9.41	0.00
LSF	91.18 ± 0.24	**0.44** ± 0.08	0.06 ± 0.01	0.49 ± 0.02	7.49	0.00
DER++	**93.88** ± 0.14	2.01 ± 0.14	0.07 ± 0.01	0.66 ± 0.01	8.82	0.00
CLPU-DER++ (scratch)	93.26 ± 0.25	2.33 ± 0.21	0.10 ± 0.01	0.09 ± 0.02	**0.09**	**1.00**
CLPU-DER++	93.48 ± 0.25	2.25 ± 0.30	0.10 ± 0.01	0.09 ± 0.02	0.13	0.96
	Rot-MNIST					
Method	ACC(↑)	FM(↓)	IJSD	AJSD	JS-ratio(↓)	IRR(↑)
Ind (Upper Bound)	95.53 ± 0.06	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.14	0.96
Seq	90.88 ± 0.43	5.66 ± 0.38	0.14 ± 0.02	0.80 ± 0.02	4.88	0.00
EWC	94.75 ± 0.12	**0.29** ± 0.12	0.09 ± 0.01	0.72 ± 0.01	7.50	0.00
ER	95.12 ± 0.18	1.39 ± 0.17	0.16 ± 0.02	0.79 ± 0.02	3.87	0.00
LwF	95.72 ± 0.19	0.87 ± 0.18	0.07 ± 0.01	0.76 ± 0.01	9.60	0.00
LSF	92.56 ± 0.09	0.30 ± 0.07	0.08 ± 0.01	0.65 ± 0.02	6.94	0.00
DER++	**95.94** ± 0.09	0.36 ± 0.08	0.12 ± 0.02	0.74 ± 0.02	5.38	0.00
CLPU-DER++ (scratch)	94.69 ± 0.11	1.02 ± 0.10	0.13 ± 0.02	0.11 ± 0.04	**0.14**	**1.00**
CLPU-DER++	95.37 ± 0.12	0.91 ± 0.09	0.14 ± 0.02	0.11 ± 0.04	0.17	**1.00**

Table 1: Performance of CLPU-DER++ against baseline methods on the Perm-MNIST and Rot-MNIST CLPU benchmarks. We report the mean and standard deviation for each result over 5 independent runs. The best results for each metric are bolded.

average over the number of unlearning tasks for both IJSD and AJSD. Therefore, in total we have $\frac{c(c-1)}{2}$ distances for IJSD and c^2 distances for AJSD, over the entire sequence of requests. Formally,

$$\begin{aligned}
\text{IJSD} &= \left\{ \sum_{1\le i<j\le c} \frac{1}{\left|\sum_{t\in[T],\rho^t=\mathbf{F}} 1\right|} \sum_{t\in[T],\rho^t=\mathbf{F}} \mathbb{E}_{(x,y)\sim D^{t}} \left[\text{JS}\left(f(x;\theta_i^{t'}) \,\middle\|\, f(x;\theta_j^{t'}) \right) \right] \right\}, \\
\text{AJSD} &= \left\{ \sum_{i,j\in[c]} \frac{1}{\left|\sum_{t\in[T],\rho^t=\mathbf{F}} 1\right|} \sum_{t\in[T],\rho^t=\mathbf{F}} \mathbb{E}_{(x,y)\sim D^{t}} \left[\text{JS}\left(f(x;\theta_i^{t}) \,\middle\|\, f(x;\theta_j^{t'}) \right) \right] \right\}.
\end{aligned} \tag{11}$$

After the IJSDs and AJSDs are calculated, we measure the ratio of the absolute difference between the average of IJSD and AJSD over the average of IJSD, which we call JS-ratio, and the proportion of AJSD that are smaller than the maximum of IJSD, which we call the In-Range Rate (IRR). Formally,

$$\text{JS-ratio} = \frac{\left| \frac{1}{|\text{IJSD}|} \sum_{d\in\text{IJSD}} d - \frac{1}{|\text{AJSD}|} \sum_{d\in\text{AJSD}} d \right|}{\frac{1}{|\text{IJSD}|} \sum_{d\in\text{IJSD}} d}, \quad \text{and} \quad \text{IRR} = \frac{\sum_{d\in\text{AJSD}} \mathbb{1}\left(d \le \max(\text{IJSD})\right)}{|\text{AJSD}|}. \tag{12}$$

5.3 Results

In Tab. 1-2, we report the comparison of CLPU-DER++ against the previously mentioned baseline methods on 4 benchmark datasetes. Specifically, we report the ACC, FM, JS-ratio and IRR metrics from previous section. To provide more information, we also report the mean and standard deviation of the sets IJSD and AJSD.[5]

From the table, we can see that CLPU-DER++ achieves the best JS-ratio and IRR among all methods. In contrast, all baseline methods achieve high JS-ratio and very low IRR, meaning that the unlearning indeed reveals that the model has learned on the unlearned task previously. On the other hand, in terms of the CL metrics, DER++ achieves the best

[5] In Tab. 1-2, we abuse the notations of IJSD and AJSD a bit and directly report the mean and standard deviation under them.

	Split-CIFAR10					
Method	ACC(↑)	FM(↓)	IJSD	AJSD	JS-ratio(↓)	IRR(↑)
Ind (Upper Bound)	91.84 ± 0.94	0.00 ± 0.00	0.03 ± 0.02	0.03 ± 0.01	0.12	1.00
Seq	76.73 ± 2.25	15.38 ± 2.67	0.09 ± 0.03	0.18 ± 0.02	0.95	0.04
EWC	79.43 ± 1.51	11.83 ± 2.08	0.14 ± 0.05	0.20 ± 0.03	0.40	0.76
ER	90.44 ± 0.57	0.83 ± 3.36	0.07 ± 0.02	0.16 ± 0.01	1.07	0.00
LwF	88.18 ± 1.87	4.34 ± 2.34	0.06 ± 0.02	0.16 ± 0.03	1.68	0.00
LSF	90.00 ± 2.30	2.48 ± 2.05	0.07 ± 0.02	0.14 ± 0.03	0.90	0.48
DER++	**91.26** ± 0.63	**0.73** ± 3.54	0.03 ± 0.01	0.07 ± 0.01	1.11	0.60
CLPU-DER++ (scratch)	89.52 ± 1.46	1.14 ± 2.17	0.03 ± 0.01	0.03 ± 0.02	0.04	**0.92**
CLPU-DER++	90.12 ± 1.65	1.89 ± 2.20	0.03 ± 0.01	0.03 ± 0.01	**0.00**	**0.92**
	Split-CIFAR100					
Method	ACC(↑)	FM(↓)	IJSD	AJSD	JS-ratio(↓)	IRR(↑)
Ind (Upper Bound)	63.86 ± 0.55	0.00 ± 0.00	0.17 ± 0.01	0.17 ± 0.01	0.00	0.96
Seq	44.34 ± 0.84	24.36 ± 2.44	0.44 ± 0.03	1.09 ± 0.03	1.47	0.00
EWC	45.39 ± 1.74	20.08 ± 1.42	0.63 ± 0.03	1.27 ± 0.04	1.02	0.00
ER	61.66 ± 1.27	7.69 ± 1.68	0.51 ± 0.03	1.11 ± 0.03	1.18	0.00
LwF	61.25 ± 2.73	8.60 ± 1.01	0.39 ± 0.03	1.06 ± 0.03	1.71	0.00
LSF	37.92 ± 2.14	26.88 ± 2.09	0.70 ± 0.03	1.09 ± 0.05	0.54	0.00
DER++	**66.66** ± 0.69	**2.84** ± 0.59	0.31 ± 0.03	0.70 ± 0.02	1.24	0.00
CLPU-DER++ (scratch)	61.51 ± 0.76	3.46 ± 1.18	0.21 ± 0.01	0.19 ± 0.03	**0.08**	**0.96**
CLPU-DER++	63.90 ± 0.77	3.90 ± 1.05	0.22 ± 0.01	0.21 ± 0.04	**0.08**	**0.96**

Table 2: Performance of CLPU-DER++ against baseline methods on on the Split-CIFAR10 and Split-CIFAR100 CLPU benchmarks. We report the mean and standard deviation for each result over 5 independent runs. The best results for each metric are bolded.

CL performance with CLPU-DER++ finishing a close second. The difference is due to the fact that when merging the temporary network back into the main model, the CLPU-DER++ agent essentially performs knowledge distillation to distill the knowledge from a temporarily learned task back to the main model. However, it is known that knowledge distillation often cannot fully recover the original model's performance. Lastly, we observe that when creating a temporary network, initializing from the main model (line 9 of Alg. 1) results in better performance compared to initializing from scratch (CLPU-DER++ (scratch)).

6 Conclusion and Future Work

In this work, we propose a novel continual learning and private unlearning (CLPU) problem and provide its formal formulation. In addition, we introduce a straightforward but exact unlearning method to solve CLPU, as well as novel metrics and adapted benchmark problems to evaluate any CLPU methods. There are many interesting future directions for the CLPU problem. First, as shown in Fig. 1, CLPU-DER++ is an initial solutionn that achieves exact privacy and good knowledge transfer ability. It will be interesting to extend it to the δ-unlearning setting while reducing the space complexity by saving fewer models. Second, it is important to understand theoretically what an optimal CLPU method can achieve. Note that in principle it might be impossible to reach the optima of the three objectives in Fig. 1 simultaneously. Lastly, it is also interesting to study how the performance of any CLPU method can be affected by the relationship of different tasks. Intuitively, similar tasks should encourage better continual learning performance but make privately unlearning more difficult.

7 Acknowledgement

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, FAIN-2019844), ONR (N00014-18-2243), ARO (W911NF-19-2-0333), DARPA, GM, Bosch, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

REFERENCES

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

Ho Bae, Jaehee Jang, Dahuin Jung, Hyemi Jang, Heonseok Ha, Hyungyu Lee, and Sungroh Yoon. Security and privacy issues in deep learning. *arXiv preprint arXiv:1807.11655*, 2018.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-jussà. Continual lifelong learning in natural language processing: A survey. *arXiv preprint arXiv:2012.09823*, 2020.

Robert A Bjork and Elizabeth L Bjork. Forgetting as the friend of learning: implications for teaching and self-regulated learning. *Advances in Physiology Education*, 43(2):164–167, 2019.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.

Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pp. 1092–1104. PMLR, 2021.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480. IEEE, 2015.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018a.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

Antonio Ginart, Melody Y Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012*, 2019.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 792–801, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019a.

Steven CY Hung, Jia-Hong Lee, Timmy ST Wan, Chein-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Increasingly packing multiple facial-informatics modules in a unified deep-learning model via lifelong learning. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pp. 339–343, 2019b.

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

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Bo Liu, Xuesu Xiao, and Peter Stone. A lifelong learning approach to mobile robot navigation. *IEEE Robotics and Automation Letters*, 6(2):1090–1096, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

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*, pp. 7765–7773, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

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

Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

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.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pp. 4528–4537. PMLR, 2018.

Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In *IJCAI*, volume 2, pp. 6, 2021.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: a general approach for growing neural networks. *Advances in Neural Information Processing Systems*, 33:22373–22383, 2020a.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pp. 10355–10366. PMLR, 2020b.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.