Disposable Transfer Learning for Selective Source Task Unlearning

Seunghee Koh[1] Hyounguk Shon[1] Janghyeon Lee[2] Hyeong Gwon Hong[1] Junmo Kim[1]
[1] Korea Advanced Institute of Science and Technology, South Korea
[2] LG AI Research, South Korea
{seunghee1215, hyounguk.shon, honggudrnjs, junmo.kim}@kaist.ac.kr
janghyeon.lee@lgresearch.ai

Abstract

Transfer learning is widely used for training deep neural networks (DNN) for building a powerful representation. Even after the pre-trained model is adapted for the target task, the representation performance of the feature extractor is retained to some extent. As the performance of the pre-trained model can be considered the private property of the owner, it is natural to seek the exclusive right of the generalized performance of the pre-trained weight. To address this issue, we suggest a new paradigm of transfer learning called disposable transfer learning (DTL), which disposes of only the source task without degrading the performance of the target task. To achieve knowledge disposal, we propose a novel loss named Gradient Collision loss (GC loss). GC loss selectively unlearns the source knowledge by leading the gradient vectors of mini-batches in different directions. Whether the model successfully unlearns the source task is measured by piggyback learning accuracy (PL accuracy). PL accuracy estimates the vulnerability of knowledge leakage by retraining the scrubbed model on a subset of source data or new downstream data. We demonstrate that GC loss is an effective approach to the DTL problem by showing that the model trained with GC loss retains the performance on the target task with a significantly reduced PL accuracy.

1. Introduction

Transfer learning [26] (TL) is one of the bedrocks in the success of deep neural networks (DNNs). The core idea of TL is to build a strong generic model that can adapt to a broad range of downstream tasks with much less amount of data. The scale of data collection for pre-training becomes the key objective to build a model with competitive performance on their target tasks [14]. Nowadays, a lot of organizations are interested in collecting an internal dataset to build their own generic model.

For example, JFT-300M [6, 24, 25], a large-scale private dataset exclusively available to Google, is used for pre-training to reach the state-of-the-art performance of downstream target tasks [12, 27] with relatively small target datasets. IG-3.5B-17k [18], an internal Facebook AI Research dataset, is also used to train a weakly supervised model. Such datasets and trained models have a high economic value in that collecting data and training a model is time-consuming and costly, and a proficient model is readily adaptable to various commercial services.

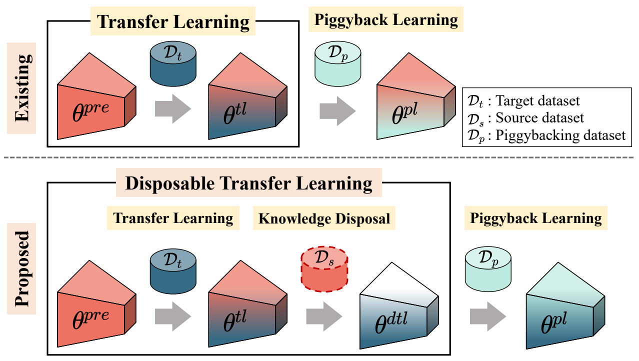


Figure 1: Illustration of the proposed Disposable Transfer Learning (DTL) framework. DTL extends the existing Transfer Learning (TL) paradigm with an additional knowledge disposal stage that scrubs off the prior knowledge irrelevant to the target task. The goal of DTL is to prevent Piggyback Learning (PL) which maliciously exploits the representation performance of a pre-trained model for a piggyback task by simply performing an extra fine-tuning step on top of the published transfer-learned model.

However, those private properties are exposed to unauthorized customizing when released. Specifically, we denote piggyback learning (PL) (Figure 1) as a kind of extra fine-tuning on other downstream tasks for leveraging the benefits of the transfer-learned model with much less effort. As shown in Figure 2, the performance of TL (blue) and PL (green) on the downstream tasks is comparable, and is considerably improved over the model trained from scratch (red). In other words, it enables anyone to exploit the pretext knowledge even when one does not have access to the pre-trained model by accessing the released transfer-learned model. PL is profitable to those free riders, so they may launch a new service or product by just exploiting the proficient model. It conflicts with the model owner's interest.

To alleviate this potential risk, we propose a novel TL paradigm that temporarily utilizes and then *disposes of* the source task knowledge after transfer learning, coined *Disposable Transfer Learning* (DTL). DTL aims to protect the exclusive license of generic performance on the internal pre-training data while achieving a powerful downstream performance.

To address the DTL problem, we propose a novel loss function that scrubs the source task knowledge, named *Gradient Collision loss* (GC loss). GC loss guides the model towards abnormal convergence on the source task by minimizing inner-products between sample gradients. GC loss deals with a non-typical unlearning problem where the scale of data to be unlearned is much larger and the dataset to be unlearned and the dataset to be retained are heterogeneous, whereas existing unlearning literature [2, 3, 7, 8] mainly focus on forgetting only a small portion of a single kind of training data.

After DTL, we measure the model's susceptibility to unwanted PL using *Piggyback Learning accuracy* (PL accuracy). We define the PL accuracy of a model as the test accuracy measured by learning an additional piggyback task. A low PL accuracy indicates that the model successfully unlearned the source knowledge so that it is resistant to a small portion of source re-training or fine-tuning on other downstream tasks. We will show the importance of PL accuracy as a measurement of unlearning for validating knowledge disposal.

We demonstrate that the model scrubbed with GC loss retains the target performance while effectively preventing the exploitation of the performance of the pre-trained model. To the best of our knowledge, DTL with GC loss is the first work in making transfer learning and unlearning compatible.

Our main contributions are summarized as follows:

- We propose a novel forgetting problem, DTL, in which we try to dispose of the generalization power of a pre-trained model while adapting the model only to a specific target downstream task.
- We propose GC loss, which is a novel loss that achieves knowledge disposal of the source task. We also provide an extremely efficient implementation of GC loss that also allows for distributed training.
- We propose PL accuracy as an evaluation metric that can estimate the performance of a DTL model.

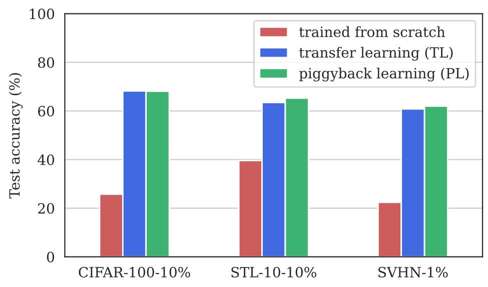


Figure 2: Performance comparison between models trained from scratch, models trained by transfer learning (TL), and models trained by piggyback learning (PL). The horizontal axis indicates the datasets used for performance evaluations. PL model (green) achieves performance comparable to TL model (blue) and both models perform much better than the models trained from scratch (red). For both TL and PL, CIFAR-100 is used as the source task. For PL, CIFAR-10-1% is additionally used as the target task before the model is piggybacked.

2. Related Work

2.1. Unlearning in deep learning

Unlearning is a training mechanism to controllably forget specific data from the knowledge of a DNN and is gaining attention in the deep learning community owing to increased public awareness of digital privacy. The existing methods come with settings to bypass restrictions of the non-convex and stochastic nature of DNN, which make the perfect unlearning almost impossible. [3] combined distributed training and ensemble training to reduce retraining time excluding samples to be unlearned. Class-wise unlearning had been attempted [2], but it only removed a class from the classifier output, not from the model parameters. [8] tried to scrub a subset of a class or an entire class with the assumption that the size of the dataset to be forgotten is much smaller than the dataset to be retained, and it uses the mean squared error loss to make the model partially convex to guarantee the perfect forgetting from the model. [9] approximated the weight difference using Neural Tangent Kernel theory [13]. [7] approximated the model to a linear model, with pre-determining a subset of the training dataset not to be unlearned and using it in pre-training.

2.2. Gradient-based learning methods

The gradient vector of the learned model on specific input data characterizes the relationship between the model and input. In representation learning, a gradient of the learned model can be used as a feature for downstream tasks [19]. In continual learning, [4, 17] take advantage of gradient vectors to reduce catastrophic forgetting of a previously learned task. Also, it has been used to select samples for episodic memory [1]. The algorithm for composing the episodic memory aims to minimize the cosine similarity of the gradient for each sample pair in candidate memory, and the authors have shown that the solution of the minimization problem is consequently maximizing the variance of the gradient of selected samples.

2.3. Readout function

Readout functions [2, 7, 8] are used to test whether a model has been successfully unlearned. Entropy, retraining time, or the success rate of membership inference attacks (MIAs) are usually used for the readout function. Entropy is used to measure the increase in uncertainty of the model after unlearning. This is based on the assumption that the prediction of a model gets less confident as it forgets the knowledge of interest. Retraining time quantifies the number of training steps required to restore the previous performance. MIA is a kind of attack method to detect whether a given sample was used in training a model, concerning the information leakage issue of training data from a trained model. Black-box MIAs only observe the input-output relationship, and white-box attacks fully exploit the model's architecture, weight, and gradient. We report the success rate of MIA against unlearned models using the strategies reported and implemented in [10]. Note that we use PL accuracy as the readout function for knowledge disposal.

3. Method

Notations Let $\mathcal{D}$ be a dataset for a classification task with input space $\mathcal{X}$ and label space $\mathcal{Y}$. We consider a DNN model $P(y|x;\theta)$ parameterized by θ that takes in an input x and predicts a categorical probability distribution. We denote a training procedure as $\theta^{out} = \textsc{Train}_{\mathcal{A}}\left(\theta^{in}, \mathcal{L}(\theta;\mathcal{D})\right)$ which represents a model learned from an initialization θ^{in}, an objective function $\mathcal{L}(\theta;\mathcal{D})$, and a training scheme $\mathcal{A}$. For example, $\mathcal{A}$ can be a stochastic gradient descent optimizer with decaying learning rate scheduling.

3.1. Disposable transfer learning

Disposable transfer learning (DTL) is a training paradigm for selective unlearning of the source task upon completion of transfer learning. As described in Figure 1, it consists of two stages: the transfer learning (TL) stage and the knowledge disposal stage of the source data.

DTL is conducted by the owner of a private source data $\mathcal{D}_s$, so $\mathcal{D}_s$ is to be unlearned and accessible during unlearning. Also, the size of the target dataset $\mathcal{D}_t$ is much smaller, *i.e.*, $|\mathcal{D}_s| \gg |\mathcal{D}_t|$, such that transfer learning is required to achieve competitive target task performance.

Transfer learning stage The transfer-learned model θ^{tl} is obtained by pre-training and fine-tuning. The model is initially pre-trained on a source dataset $\mathcal{D}_s$ from scratch,

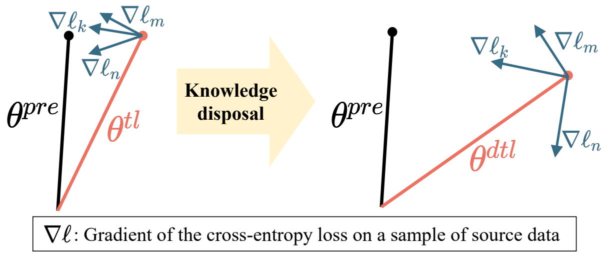


Figure 3: A conceptual diagram of gradients after transfer learning (left) and knowledge disposal (right). The gradients from the source data examples are marked as blue arrows. As depicted in Figure 1, θ^{pre}, θ^{tl} and θ^{dtl} correspond to the pre-trained model, transfer-learned model, and disposable transfer-learned model, respectively.

and the output model θ^{pre} is then easily adaptable to the target task. Then θ^{pre} is fine-tuned to get $\theta^{tl} = \textsc{Train}_{\mathcal{A}}\left(\theta^{pre}, \mathcal{L}(\theta;\mathcal{D}_t)\right)$. Due to the small scale of $\mathcal{D}_t$, fine-tuning has a reasonable training cost and the fine-tuned weight θ^{tl} is not perturbed significantly from θ^{pre}.

Knowledge disposal stage Knowledge disposal stage is the last stage of DTL for building θ^{dtl} by disposing of the source task from the transfer-learned model θ^{tl} with the DTL loss $\mathcal{L}_{DTL}$, which we denote as $\theta^{dtl} = \textsc{Train}_{\mathcal{A}}\left(\theta^{tl}, \mathcal{L}_{DTL}(\theta;\mathcal{D})\right)$.

For a successful disposable *transfer learning* where good generalization to the target task is one of the key factors, θ^{dtl} has to retain the performance on the target task of θ^{tl}. Simultaneously, it is essential to dispose of the source performance not to be recovered through PL, as discussed in Section 3.3. We formulate those factors into the retaining loss and the unlearning loss by combining them into a single objective as Equation (1). Here, λ is a scalar hyperparameter that controls the level of unlearning.

$$\mathcal{L}_{DTL}(\theta) = (1-\lambda)\cdot\mathcal{L}_{retain}(\theta) + \lambda\cdot\mathcal{L}_{unlearn}(\theta) \quad (1)$$

3.2. Retaining loss and unlearning loss

3.2.1 Retaining by knowledge distillation loss

To effectively retain the transferred knowledge on the downstream task, we adopt knowledge distillation loss (KD loss) [12]. KD loss is for transferring knowledge between different models by setting the output of the teacher model to a soft target and minimizing the KL divergence of the soft target and the output of a student model, as formulated in Equation (2). In this paper, we choose $\mathcal{D} = \mathcal{D}_s$ to prevent a risk of over-fitting due to the small size of $\mathcal{D}_t$.

$$\mathcal{L}_{retain}(\theta) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[D_{KL}\left(P(y|x;\theta^{tl})||P(y|x;\theta)\right)\right] \quad (2)$$

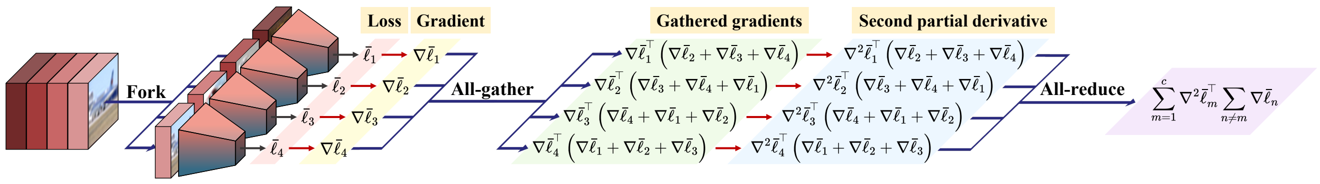

Loss
Gradient
Gathered gradients
Second partial derivative
Fork
All-gather
All-reduce
$\bar{\ell}_1 \rightarrow \nabla\bar{\ell}_1$
$\bar{\ell}_2 \rightarrow \nabla\bar{\ell}_2$
$\bar{\ell}_3 \rightarrow \nabla\bar{\ell}_3$
$\bar{\ell}_4 \rightarrow \nabla\bar{\ell}_4$
$\nabla\bar{\ell}_1^\top(\nabla\bar{\ell}_2+\nabla\bar{\ell}_3+\nabla\bar{\ell}_4) \rightarrow \nabla^2\bar{\ell}_1^\top(\nabla\bar{\ell}_2+\nabla\bar{\ell}_3+\nabla\bar{\ell}_4)$
$\nabla\bar{\ell}_2^\top(\nabla\bar{\ell}_3+\nabla\bar{\ell}_4+\nabla\bar{\ell}_1) \rightarrow \nabla^2\bar{\ell}_2^\top(\nabla\bar{\ell}_3+\nabla\bar{\ell}_4+\nabla\bar{\ell}_1)$
$\nabla\bar{\ell}_3^\top(\nabla\bar{\ell}_4+\nabla\bar{\ell}_1+\nabla\bar{\ell}_2) \rightarrow \nabla^2\bar{\ell}_3^\top(\nabla\bar{\ell}_4+\nabla\bar{\ell}_1+\nabla\bar{\ell}_2)$
$\nabla\bar{\ell}_4^\top(\nabla\bar{\ell}_1+\nabla\bar{\ell}_2+\nabla\bar{\ell}_3) \rightarrow \nabla^2\bar{\ell}_4^\top(\nabla\bar{\ell}_1+\nabla\bar{\ell}_2+\nabla\bar{\ell}_3)$
$\sum_{m=1}^{c}\nabla^2\bar{\ell}_m^\top\sum_{n\neq m}\nabla\bar{\ell}_n$

Figure 4: Illustration of distributed computation for GC loss. The computation of GC loss and its gradient can be optimized by rearranging Equation (9). The rearrangement makes the HVP computation parallelized and enables distributed processing of the backward pass. The pseudo-code is reported in Algorithm 1.

3.2.2 Unlearning by gradient collision loss

Consider a dataset of training examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. The cross-entropy loss of a model θ on an example (x_i, y_i) is denoted by $\ell(\theta; x_i, y_i)$, which is abbreviated to $\ell_i(\theta)$ when clear from the context.

In the context of unlearning, a key concept is how the value of $\ell_n(\theta)$ changes when the model weights are updated to reduce $\ell_m(\theta)$. In gradient descent, the model weight is optimized to minimize the loss function by updating θ by $\Delta\theta$ which is $\nabla\ell(\theta)$ scaled by the learning rate η:

$$\Delta\theta = -\eta\nabla\ell(\theta; x_m, y_m) \tag{3}$$

The change of $\ell_n(\theta)$ over $\Delta\theta$ is approximated by its first-order Taylor expansion,

$$\Delta\ell_n(\theta) = \ell_n(\theta + \Delta\theta) - \ell_n(\theta) \tag{4}$$
$$\simeq \nabla\ell_n(\theta)^\top\Delta\theta. \tag{5}$$

We get the following relationship between a pair of gradients on the loss by combining Equation (3) and Equation (5).

$$\Delta\ell_n(\theta) \simeq -\eta\nabla\ell_m(\theta)^\top\nabla\ell_n(\theta) \tag{6}$$

Equation (6) states that the value of $\ell_n(\theta)$ is affected by the relationship between $\nabla\ell_m$ and $\nabla\ell_n$. As illustrated in the left side of Figure 3, the gradients of a pair of examples point in similar directions, *i.e.* the inner-product of the gradients is positive, the model update for reducing $\ell_m(\theta)$ also causes a decrease in $\ell_n(\theta)$. On the other hand, as in the right side of Figure 3, the update for reducing the loss on a sample does not imply the other's learning or can even hinder the other's learning. Hence, we claim that the behavior of the model to learn the knowledge can be hindered by colliding the gradient of the examples of the dataset, which corresponds to reducing the inner product of the gradient vectors. Finally, we propose the gradient collision loss (GC loss).

Definition 3.1 (Gradient Collision loss)**.** The full gradient collision loss defined for a dataset $\mathcal{D}$ of size N is given by,

$$\mathcal{L}_{gc}(\theta, \mathcal{D}) = \frac{1}{\binom{N}{2}}\sum_{m\neq n}\nabla\ell_m(\theta)^\top\nabla\ell_n(\theta). \tag{7}$$

When the model is unlearned, each gradient vector changes direction as seen in the Figure 3 and they converge towards perpendicular angles which results in suppressed inner-product value. Note that GC loss does not have a zero minimum. The value is negative when the per-sample gradients point in opposing directions. In this case, a gradient descent step on one sample leads to a loss increase on another sample.

This ideal form aggregates every pair of possible combinations of ℓ_i, however, the computation quickly becomes intractable as the size N grows. Therefore, we calculate the inner products within a stochastically sampled mini-batch by dividing a mini-batch into c number of chunks and colliding the gradients of the chunks. We calculate the unlearning loss for every chunk pair, which are $\binom{c}{2}$ number of pairs per mini-batch.

Definition 3.2 (Stochastic Gradient Collision loss)**.** We define a variant of GC loss for mini-batch training. The stochastic GC loss is the sum of inner products between pairs of chunk-averaged gradients. Here, $\nabla\bar{\ell}_i$ is the gradient averaged over the i-th chunk in a mini-batch, and c is the number of chunks per mini-batch.

$$\mathcal{L}_{gc}(\theta, \mathcal{D}) = \frac{1}{\binom{c}{2}}\sum_{m\neq n}\nabla\bar{\ell}_m(\theta)^\top\nabla\bar{\ell}_n(\theta) \tag{8}$$

We compute GC loss on $\mathcal{D}_s$, for disposing of the source knowledge by guiding the gradient of source data colliding. We use Equation (8) as our unlearning objective along with a custom gradient computation for efficient training.

Efficient training of GC loss Training with a naive implementation of GC loss is expensive because it requires calculating the gradient of each chunk in series, summing up the inner product of all pairs of the gradient vectors, and then conducting back-propagation.

For computational efficiency, we re-formulate the derivative of GC loss to a sum of Hessian-vector products (HVPs), as in Equation (10).

Algorithm 1 Pseudocode for GC loss, PyTorch-like

```
1 # Inputs: net, input, target, lr
2 ce_loss = cross_entropy(net(input), target)
3 ce_grad = grad(ce_loss, net.parameters())
4 chunk_grad = all_gather(ce_grad.detach()).sum(0)
5 gc_loss = dot_prod(ce_grad, chunk_grad)
6 gc_grad = grad(gc_loss, net.parameters())
7 gc_grad = all_reduce(gc_grad)
8 optimizer_step(net.parameters(), gc_grad, lr)
```

$$\nabla \mathcal{L}_{gc} = \frac{1}{\binom{c}{2}} \sum_{m \neq n} \nabla \left(\nabla \bar{\ell}_m{}^\top \nabla \bar{\ell}_n \right) \tag{9}$$

$$= \frac{1}{\binom{c}{2}} \sum_{m=1}^{c} \nabla^2 \bar{\ell}_m{}^\top \sum_{m \neq n} \nabla \bar{\ell}_n \tag{10}$$

This reduces repeated computation of intermediate vectors and parallelizes computing the derivative of each chunk and each HVP along the chunk axis for distributed processing across multiple GPUs as depicted in Figure 4.

A detailed description of the algorithm is provided in Algorithm 1. We use `DistributedDataParallel` [22] primitives of PyTorch which provide communication for multiprocessing. First, each process calculates the chunk-wise gradient of cross-entropy loss (Line 3). The gradients are aggregated across processes using `gather` operation to compute the GC loss (Line 4). `detach` allows for calculating the partial derivative of gradient which is required for computing the HVP. Finally, the GC loss is then partially back-propagated, and we `reduce` the values to obtain the gradient of GC loss as Equation (10) to update the model parameters.

While a naive combinatorial implementation for GC loss has $\mathcal{O}(c^2)$ complexity, our re-formulation greatly reduces the cost to $\mathcal{O}(c)$. Note that this matches the complexity of typical loss functions. This is because our method only needs products between individual vectors against an averaged value, not pair-wise products, which enables data parallelism for multiprocessing. An extra cost is a backward-on-backward step for propagating the gradient of GC loss in Line 6. Fortunately, this only requires a similar computational cost and memory footprint to a typical backpropagation.

3.2.3 Baseline unlearning methods

As baselines, we consider three additional naive unlearning losses adopted from [8]. Random target fooling loss updates the model using cross-entropy loss with $\tilde{\mathcal{D}}$, a dataset constructed from $\mathcal{D}$ with random target labels. This leads the model to memorize wrong answers so that the model unlearns the related knowledge.

$$\mathcal{L}_{rand}(\theta; \mathcal{D}) = \mathop{\mathbb{E}}_{(x,y)\sim\tilde{\mathcal{D}}} [-\log P(y|x;\theta)] \tag{11}$$

Uniform target fooling loss guides the model to output uniform distribution, therefore the model loses the ability to make any prediction. Here, $\mathcal{U}(y)$ is an uniform distribution over $\mathcal{Y}$.

$$\mathcal{L}_{unif}(\theta; \mathcal{D}) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} [D_{KL}\left(\mathcal{U}(y) || P(y|x;\theta)\right)] \tag{12}$$

Negative cross-entropy loss flips the learning signal by increasing the cross-entropy. The concept is that increasing loss through gradient ascent steps lets a model be forgotten.

$$\mathcal{L}_{neg}(\theta; \mathcal{D}) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} [\log P(y|x;\theta)] \tag{13}$$

We will compare the proposed unlearning loss $\mathcal{L}_{gc}$ against $\mathcal{L}_{rand}$, $\mathcal{L}_{unif}$, and $\mathcal{L}_{neg}$ on $\mathcal{D}_s$ in Section 4.

3.3. Evaluation by piggyback learning accuracy

In this section, we establish an evaluation protocol for DTL performance using piggyback learning accuracy (PL accuracy). While measuring the source task accuracy may seem like the most direct approach for evaluating knowledge disposal, we have found that it is less effective measure due to trivial solutions such as last-layer fooling. Specifically, if the source task accuracy is used as an unlearning metric, a trivial solution is to simply collapse the source classifier since there are separate classifiers for the unlearned task and the retained task in DTL. Designing a proper evaluation protocol for DTL is crucial, as the ultimate goal is to prevent the model from learning an unknown piggyback task after transfer learning is completed. To address these challenges, we propose to benchmark DTL using PL accuracy.

Definition 3.3 (Piggyback learning and piggyback learning accuracy)**.** We define Piggyback Learning (PL) as adapting a model θ^0 on a piggyback task $\mathcal{D}_p$ using a fine-tuning scheme $\mathcal{A}$.

$$\theta^{pl} = \textsc{Train}_{\mathcal{A}}\left(\theta^0, \mathcal{L}(\theta; \mathcal{D}_p^{train})\right) \tag{14}$$

where $\mathcal{D}_p^{train}$ is the train split of $\mathcal{D}_p$. Piggyback learning accuracy $Acc_{pl}(\theta^0)$ of a model θ^0 is the test accuracy of θ^{pl} on the piggyback task. θ^0 is set to either θ^{tl} or θ^{dtl} in our main experiments.

We use multiple datasets for measuring PL accuracy. Unlike a typical test accuracy, which is measured on a certain dataset, our protocol estimates the performance by fine-tuning the model obtained from DTL and testing the performance on multiple datasets.

The PL accuracy quantifies the susceptibility of a model to various kinds of possible downstream tasks. In other words, it measures the model's transferability as a pre-trained weight.

Model	CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1%					CIFAR-100 $\xrightarrow{DTL}$ STL-10-10%					TinyImageNet$\xrightarrow{DTL}$CIFAR-100-10%				
	ΔAcc_t vs TGT ↑	ΔAcc_t vs TL ↑	ΔAcc_{pl} vs TL ↓			ΔAcc_t vs TGT ↑	ΔAcc_t vs TL ↑	ΔAcc_{pl} vs TL ↓			ΔAcc_t vs TGT ↑	ΔAcc_t vs TL ↑	ΔAcc_{pl} vs TL ↓		
			C100-10%	S10-10%	SV-1%			C100-10%	C10-1%	SV-1%			TIN-6%	S10-10%	C10-1%
RAND	+33.88	-1.93	-14.16	-3.00	+4.01	+22.89	-0.98	-11.27	-1.91	+6.42	+28.63	-2.20	-9.98	-1.90	-4.68
UNIF	+36.29	+0.48	-13.04	-1.17	+7.66	+23.46	-0.41	-12.48	-2.55	+6.40	+29.63	-1.20	-11.17	-3.50	-3.48
NEG	+36.05	+0.24	-7.18	-0.09	+9.13	+21.28	-2.59	-10.40	-4.15	+9.65	+28.80	-2.03	-11.98	-2.77	-3.87
GC (Ours)	+33.84	-1.97	**-24.53**	**-7.47**	**-6.10**	+21.95	-1.92	**-22.99**	**-10.93**	**-4.19**	+29.19	-1.64	**-12.79**	**-4.88**	**-7.90**
Reference	35.12	70.93	68.10	65.25	61.97	39.58	63.45	68.15	72.12	60.82	25.16	55.99	39.17	67.71	72.08

Table 1: The transition of source, target, and piggyback learning accuracy for each DTL stage in CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1%, CIFAR-100 $\xrightarrow{DTL}$ STL-10-10%, and TinyImageNet $\xrightarrow{DTL}$ CIFAR-100-10% experiments. The values indicate the difference between DTL and TL counterparts as percentage points. The proposed GC loss significantly outperforms unlearning baselines in achieving DTL. This provides control over selectively removing pretext knowledge from the model after learning the target task. *Reference* indicates the base accuracy (%) used to obtain the difference. CIFAR-10/100, STL-10, SVHN, and TinyImageNet are abbreviated as C10/100, S10, SV, and TIN, respectively.

When $\mathcal{D}_p$ is substituted with the source dataset $\mathcal{D}_s$, PL accuracy can be also used to estimate the recoverability of the scrubbed model. Lower PL accuracy implies a lower risk of catastrophic leakage of generic performance because the difficulty of relearning depends on the amount of knowledge remaining in the unlearned model.

4. Experiment

4.1. Experimental setting

We denote a DTL training scheme as $\mathcal{D}_s \xrightarrow{DTL} \mathcal{D}_t$ where $\mathcal{D}_s$ is the source task and $\mathcal{D}_t$ is the target task. We used CIFAR-10/100, STL-10, SVHN, and TinyImageNet [5, 15, 16, 21] to construct the benchmarks. Additionally, we reduced the scale of the target training datasets by subsampling. The training sets are reduced from the original size by class-balanced random sampling. Sub-sampled datasets have their name suffixed by "-γ%", where γ is the sampling ratio. Reducing the dataset size sets heavier emphasis on transfer learning as TL brings more impact to the target task performance (Acc_t). The unlearned models are evaluated by the target accuracy and PL accuracy (Acc_{pl}). For measuring the PL accuracy, the datasets are subsampled. Columns marked with an up-arrow symbol (↑) indicate that higher is better, while a down-arrow symbol (↓) indicates the opposite. We used ResNet-18 [11] architecture for all experiments.

4.2. Baseline methods

The most naive baseline is TGT model, which is trained only with $\mathcal{D}_t$ from scratch. In addition, we compared the effect of the baseline unlearning losses in Section 3.2.3 by conducting the knowledge disposal stage from the transfer-learned model (TL). We named each model following the name of unlearning loss. The model unlearned using the proposed GC loss (Equation (8)) is marked with GC. We use random target fooling loss (RAND, Equation (11)), uniform target fooling loss (UNIF, Equation (12)), and negative cross-entropy loss (NEG, Equation (13)). Refer to the supplementary materials for the hyperparameter λ in Equation (1) used in our main experiments. For a fair comparison of unlearning methods, we adjusted λ so that the differently unlearned models are compared with similar target accuracy.

4.3. Piggyback learning accuracy of unlearned models

Table 1 shows the comparison of four DTL models, which have similar target accuracy (Acc_t). Acc_t is improved significantly in CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1% experiment than TGT model, with negligible performance degradation compared to TL model. A similar trend is observed in CIFAR-100 $\xrightarrow{DTL}$ STL-10-10% case, where Acc_t is significantly higher than the TGT model with only a negligible penalty. Likewise, GC loss is superior to the others in TinyImageNet $\xrightarrow{DTL}$ CIFAR-100 experiment.

The model with lower PL accuracy (Acc_{pl}) successfully disposes of the source knowledge so is less susceptible to other piggyback tasks. Our experiments show that the proposed method (GC model) is the most effective in preventing piggyback learning. Notably, when the pre-training task is CIFAR-100 and the piggyback task is CIFAR-100-10%, the GC model successfully unlearns the source task knowledge by a significant gap in PL accuracy with other baselines. The PL accuracy of the GC model showed a decrease of 24.53 percentage points compared to the TL model, whereas the PL accuracy of the best-performing baseline (RAND) was only 14.16 percentage points lower than that of the TL model.

In addition, when the SVHN-1% dataset is piggybacked, only the GC model successfully prevents the full recov-

Model	MIA strategy ↓				
	Adv. Dist	†Grad w	†Grad x	†WB	Avg.
TL	63.63	65.33	65.09	65.29	64.84
RAND	50.23	50.90	50.58	51.49	50.80
UNIF	50.40	51.10	50.64	53.14	51.32
NEG	**50.00**	51.91	50.74	56.93	52.40
GC	50.11	**50.69**	**50.45**	**50.52**	**50.44**

Table 2: Membership inference attack (MIA) accuracy on CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10 experiment. A lower value indicates better robustness to MIA, therefore more success in unlearning. MIA strategies used are [10, 20, 23]. † additionally involves training an attacker model.

Model	CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1%			CIFAR-100 $\xrightarrow{DTL}$ STL-10-10%		
	Acc_s	Acc_t	Acc_{pl}	Acc_s	Acc_t	Acc_{pl}
TL	67.46	70.93	68.10	65.41	63.45	68.15
RAND	1.64	69.00	53.94	2.02	62.47	56.88
UNIF	2.74	71.41	55.06	7.03	63.04	55.67
NEG	0.02	71.17	60.92	0.02	60.86	57.75
GC	2.41	68.96	43.57	3.14	61.53	45.16

Table 3: Comparison of source task accuracy (Acc_s) and PL accuracy (Acc_{pl}) after DTL. Acc_s does not serve as a proxy for estimating the degree of knowledge disposal (Acc_{pl}).

ery of the PL accuracy of the TL model whereas others surpass the PL accuracy of the TL model. The proposed GC model significantly outperforms the baselines all while achieving target task performance comparable with the TL model across all benchmarks including the TinyImageNet experiments.

4.4. Robustness to membership inference attacks

We investigated the potential security issue of membership information leakage on the private source data by MIA strategies, as mentioned in Section 2.3. The success rate for white-box MIAs is reported in Table 2 to evaluate the robustness of the DTL models. The result shows that all unlearned models are significantly more robust to MIAs than the TL model which has not been unlearned. Among them, the GC model is the most resistant to MIA attacks, demonstrating the lowest success rate across most attack strategies. Notably, in the case of WB attack [20] which utilizes intermediate features for fitting the attack model, the GC model remains considerably robust while other baselines are shown more vulnerable. It is because while other baselines focus on perturbing the output layer, GC loss directly perturbs the hidden layer representations through the gradient vectors, which results in significantly better resilience against multiple MIA-based privacy attacks.

4.5. Effectiveness of piggyback learning accuracy

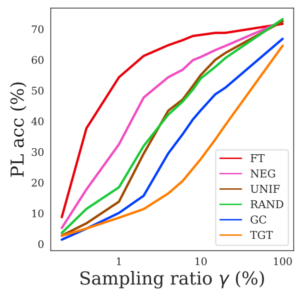
We inspected the effectiveness of PL accuracy as an evaluation metric of knowledge disposal from two perspectives. First, our results show that the source accuracy of the DTL model cannot be used as a representative measure for estimating knowledge disposal. As shown in Table 3, the UNIF model and GC model exhibit similar source accuracy (Acc_s), yet their PL accuracy on source data (Acc_{pl}) differs significantly in both CIFAR-100$\xrightarrow{DTL}$CIFAR-10-1% and CIFAR-100$\xrightarrow{DTL}$STL-10-10% experiments. This is consistent with our observations that trivial unlearning can occur by simply choosing to disrupt the classification layer, resulting in degraded performance while the feature extractor remains intact.

Furthermore, we have observed that PL accuracy stays an effective metric across an arbitrary size of the piggyback dataset. In Figure 5, we examined the PL accuracy on the source data, CIFAR-100 (Figure 5a), and new downstream data, SVHN (Figure 5b), in CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1% experiment. We simulated with an arbitrary size of the piggyback dataset by sampling γ% of the original training data.

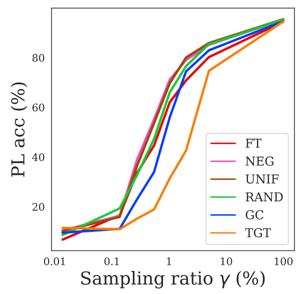
Interestingly, we found that the ranking of the PL accuracy on source data stays consistent across all ranges of sampling ratio (Figure 5a), which means that the data size has little effect on the validation of knowledge disposal. Measuring PL accuracy on another downstream task (Figure 5b) has a similar effect on measuring the vulnerability of unlearned models on extra fine-tuning. In both experiments, the TGT model (orange) shows the weakest transfer for PL since it has not learned $\mathcal{D}_s$. The GC model (blue) behaves relatively similarly to TGT while the TL model is much more susceptible to PL. This is because the TL does not forget the source task knowledge. Note that when the sampling ratio γ approaches 100%, the PL accuracy metrics converge and become less discriminative.

4.6. Sensitivity analysis for λ

In this section, we discuss how the trade-off between model performance on the target accuracy and PL accuracy behaves across varying values of λ in Equation (1) and characterizes the models. In Figure 6, we show the PL accuracy vs. target accuracy curve by varying λ. In Figure 6a, we vary the unlearning losses while the KD loss is fixed as the source knowledge retaining loss. In Figure 6b, the unlearning loss is fixed to GC loss while we vary the knowledge retaining loss. Both experiments are conducted on the CIFAR-100 $\xrightarrow{DTL}$ CIFAR-10-1% experiment with CIFAR-100-10% as the piggyback dataset.

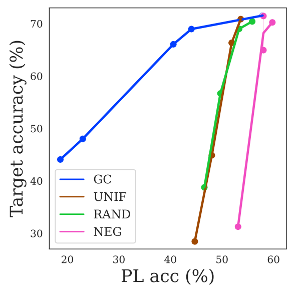


(a) CIFAR-100-$\gamma\%$

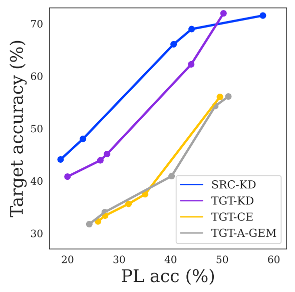


(b) SVHN-$\gamma\%$

Figure 5: PL accuracy in CIFAR-100$\xrightarrow{DTL}$CIFAR-10-1% experiment under varying dataset sampling rate γ.



(a) Unlearning loss



(b) Knowledge retaining loss

Figure 6: Trade-off of target accuracy and PL accuracy with varying $\lambda \in [0, 1]$. We experimented with different knowledge-retaining losses and unlearning losses.

Our results show that the powerful DTL performance of the combined loss of KD loss on the source data and GC loss is not a coincidence of a single value of λ, but it works for different levels of target and PL accuracy. In both experiments, the model which fits the unlearning goal of DTL is with high target accuracy and low PL accuracy, as it is plotted on the upper left region. As λ increases, the target accuracy and PL accuracy decrease as the unlearning loss becomes dominant. In Figure 6a, it is observed that GC loss (blue) performs much better on the target task with the same level of PL accuracy. The GC loss retains target performance with significantly lower PL accuracy compared to the others which show catastrophically degraded target performance, even at a high PL accuracy.

Meanwhile, retaining the target task knowledge is another important factor in successful knowledge disposal. In Figure 6b, we compare different knowledge retaining losses. We compare the adopted KD loss with $\mathcal{D}_s$ and the fine-tuned model for retaining the target knowledge (SRC-KD) as in Equation (2) against three other knowledge retaining baselines: KD loss with the target dataset (TGT-KD), training the model jointly with the typical cross-entropy loss with the target dataset (TGT-CE), cross-entropy loss replaced with A-GEM (TGT-A-GEM) [4, 17], which is a constrained optimization technique used for continual learning. See Section C.1 in the supplementary materials for details on TGT-A-GEM.

We found that KD-based knowledge retaining losses (SRC-KD, TGT-KD) outperform the non-KD methods. They show significantly higher target accuracy than other baselines with the same level of PL accuracy. This is because KD not only retains the target knowledge but also transfers dark knowledge from the TL model. The results provide justifications for using SRC-KD over TGT-KD in knowledge retaining. As discussed in Section 3.1 and Section 3.2.1, the amount of the target data is insufficient to represent the whole target distribution, whereas the larger source data facilitates better knowledge distillation from the TL model.

5. Conclusion

We propose a novel transfer learning scheme, named disposable transfer learning (DTL), which is designed to address the risk of piggybacking on a transfer-learned model when the model is released to the public. Our results highlight the promising potential of DTL towards preventing unauthorized exploitation of pre-trained weight for performance gain once the target task is adapted through transfer learning. To selectively dispose of the source knowledge, we propose a novel unlearning loss, coined gradient collision loss (GC loss). We have demonstrated that a combination of KD loss and GC loss successfully achieves DTL. Further, we propose an evaluation protocol named piggyback learning accuracy (PL accuracy) which verifies the susceptibility against piggyback learning. We have demonstrated that GC loss unlearns a model to make it less susceptible to malicious piggybacking through low PL accuracy, emphasizing the effectiveness of our method.

Limitations and Future Works

Our work focused on establishing the DTL paradigm and its implementation. While our GC loss has shown superiority compared to the basic methods, there remains considerable room for improving the unlearning objective. Specifically, better integration of the knowledge retaining and unlearning objectives needs to be further explored; in our approach, we chose to simply optimize the sum of two separate objectives. Additionally, our studies were confined to relatively small-to-medium-scale datasets and model architecture. We believe that our work provides a foundation for future research for DTL. We encourage subsequent studies to design a better-integrated objective and investigate DTL within larger, more realistic contexts.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00951, Development of Uncertainty-Aware Agents Learning by Asking Questions).

References

[1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Thomas Baumhauer, Pascal Schöttle, and M. Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *ArXiv*, 2020.

[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

[5] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *PMLR*, 2011.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[7] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *CVPR*, 2021.

[8] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. *CVPR*, 2020.

[9] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *ECCV*, 2020.

[10] Ganesh Del Grosso, Hamid Jalalzai, Georg Pichler, Catuscia Palamidessi, and Pablo Piantanida. Leveraging adversarial examples to quantify membership information leakage. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10389–10399, 2022.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[13] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8580–8589, Red Hook, NY, USA, 2018.

[14] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision*, 2019.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[16] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[17] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

[18] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[19] Fangzhou Mu, Yingyu Liang, and Yin Li. Gradients as features for deep representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[20] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 739–753. IEEE, 2019.

[21] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[23] S. Rezaei and X. Liu. On the difficulty of membership inference attacks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7888–7896, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[24] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[25] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021.

[26] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

[27] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021.