Machine Unlearning: SCRUBbing the information

Anonymous Authors[1]

Abstract

Machine Unlearning in deep neural networks is the process of 'removing' information from a trained model. Unlearning is crucial for addressing biases, resolving confusion, upholding user privacy and in various other aspects of model improvement and ethical AI. In this paper we look at some of the state-of-the-art unlearning algorithms like SCRUB. This paper presents a novel approach within the SCRUB framework, introducing feature-based distillation as an alternative to the existing response-based distillation. Additionally, we use saliency maps to gain insights into the process of unlearning and also understand the impact of unlearning on model behaviour.

1. Introduction

In an era where our personal data permeates every aspect of our digital lives, the right to control one's information is important. Individuals have the right to opt out of contributing their data to the training models of companies and organizations. Individuals may wish to invoke their right to be forgotten, necessitating a mechanism for scrubbing their identity from trained neural networks. This becomes particularly crucial in the context of data protection regulations such as the General Data Protection Regulation (GDPR) established by the European Union, which grants individuals the right to be forgotten. While this issue is recent, legal experts emphasize the considerable and potentially detrimental costs associated with GDPR compliance for businesses and entities. If a malicious entity or attacker successfully exploits a trained model to access private information of an individual who has exercised the right to be forgotten, the model owner may face liability for not implementing adequate protective measures.

However, the conventional approach of solely deleting training data proves inadequate, as neural networks remain susceptible to privacy breaches via attacks like Model Inversion and Membership Inference. The straightforward resolution would be to discard the trained model entirely and initiate new training from scratch. Unfortunately, training machine learning models is known to be both expensive and time-consuming. Consequently, model owners may be hesitant to repeatedly undergo the resource-intensive process of training new models for compliance every time an individual exercises the right to be forgotten. Thus, it is in their best interest to adopt methods that efficiently eliminate learned data, addressing not only compliance but also mitigating the potential economic harm associated with the constant replacement of trained models. This underscores the need for advanced techniques, like the one proposed in this research, that selectively forget information without compromising model's performance on retained datasets, allowing companies to respect individuals' privacy rights while maintaining operational efficiency.

2. Related Work

2.1. Retraining

As discussed previously, the most obvious solution to machine unlearning and our baseline model is the naive method of simply erasing the sensitive data from the train set and training the model from scratch. Thus, the retrained DNN is the optimal unlearned model.

2.2. Fine Tuning

As a starting point, we examine the simplistic approach of retraining the model without the forget data. The concept of catastrophic interference (Sharkey and Sharkey 1995) warns us that when a neural network encounters new data, there is a likelihood of losing previously acquired information. In the case of a network initially trained on a dataset D and then retrained on the same dataset minus a subset of data (D S), there is a substantial probability of information loss about S over time. Although this method carries a high probability of causing the neural network to forget the intended information, it lacks specifics regarding the duration of this process. Despite its limitations, it serves as a valuable benchmark because the straightforward removal of data and subsequent retraining offers a simple solution.

2.3. NegGrad

In Negative Gradient (NegGrad) fine-tune on the model by moving in the direction of increasing loss for samples in D_f , which is equivalent to using a negative gradient for the samples to forget. This aims to damage features predicting

D_f correctly.

$$w = w + \epsilon \cdot \delta_w L(x_f, w) \quad (1)$$

2.4. NegGrad+

Similar to Finetuning, starts from the original model and finetunes it both on data from the retain and forget sets, negating the gradient for the latter. NegGrad considers a weaker baseline that only trains on the forget set, with a negative gradient. We tune the stronger NegGrad+ to achieve a good balance between the two objectives

$$w = w + \epsilon \cdot \delta_w L(x_f, w) - \alpha \cdot \delta_w L(x_r, w) \quad (2)$$

2.5. SCRUB

It is an unlearning algorithm based on a novel casting of the unlearning problem into a teacher-student framework. We consider the original model w^o as the 'teacher' and train a 'student' w^u that selectively obeys that teacher - w^o should be obeyed when teaching about D_r , but disobeyed when teaching about D_f. It gives the optimization objective as

$$\min_{w^u} \frac{1}{N_r} \sum_{x_r \in D_r} d(x_r, w^u) - \frac{1}{N_f} \sum_{x_f \in D_f} d(x_f, w^u) \quad (3)$$

where $d(x; w_u)$ is the KL-divergence between the student and teacher output distributions (softmax probabilities) for the example x

$$d(x; w_u) = \text{DKL}\left(p(f(x; w_o)) \,\|\, p(f(x; w_u))\right) \quad (4)$$

Furthermore, SCRUB also simultaneously optimizes for the task loss on the retain set, to further strengthen the incentive to perform well there. Our final training objective is then the following:

$$\min_{w^u} \frac{1}{N_r} \sum_{x_r \in D_r} d(x_r, w^u) - \frac{1}{N_f} \sum_{x_f \in D_f} d(x_f, w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in D_r} l(f(x_r, w^u), y_r) \quad (5)$$

where l stands for the cross-entropy loss and the and are scalars that we treat as hyperparameters. In practice, optimizing this objective is challenging as we are trying to simultaneously satisfy two objectives, which may interfere with each other, namely moving close to the teacher on some data points while moving away from it on others. To address this, SCRUB provides a practical recipe for optimization, reminiscent of 'tricks' used in other min-max-like problems like Generative Adversarial Networks where the discriminator is trained for several steps before each update to the generator.

Specifically, SCRUB iterates between performing an epoch of updates on the forget set (the maxstep) followed by an epoch of updates on the retain set (the min-step), in an alternating fashion. Guarding against hurting retain performance due to this alteration of min-steps and max-steps, SCRUB also performs a sequence of additional min-steps at the end of the sequence to restore the retain performance in the event that it was harmed. SCRUB training stops when the forget error has increased without harming the retain set error. We find in practice this point can be reached with a small number of epochs.

The following is the pseudo-code for SCRUB

Algorithm 1 SCRUB

Require: Teacher weights w_o
Require: Total max steps *MAX-STEPS*
Require: Total steps *STEPS*
Require: Learning rate ϵ
$w_u \leftarrow w_o$
$i \leftarrow 0$
repeat
  if $i <$ *MAX-STEPS* **then**
    $w_u \leftarrow$ DO-MAX-EPOCH(w_u)
  end if
  $w_u \leftarrow$ DO-MIN-EPOCH(w_u)
  $i \leftarrow i + 1$
until $i <$ *STEPS* =0

Algorithm 2 DO-MAX-EPOCH

Require: Student weights w_u
Require: Learning rate ϵ
Require: Batch size B
Require: Forget set D_f
Require: Procedure NEXT-BATCH
$b \leftarrow$ NEXT-BATCH(D_f, B)
repeat
  $w_u \leftarrow w_u + \epsilon \nabla w_u \frac{1}{|b|} \sum_{xf \in b} d(x_f; w_u)$
  $b \leftarrow$ NEXT-BATCH(D_f, B)
until b =0

Algorithm 3 DO-MIN-EPOCH

Require: Student weights w_u
Require: Learning rate ϵ
Require: Batch size B
Require: Retain set D_r
Require: Procedure NEXT-BATCH
$b \leftarrow$ NEXT-BATCH(D_r, B)
repeat
$w_u \leftarrow w_u - \epsilon \nabla_{w_u} \frac{1}{|b|} \sum_{(x_r, y_r) \in b} \alpha d(x_r; w_u)$
$+\gamma l(f(x_r; w_u), y_r)$
$b \leftarrow$ NEXT-BATCH(D_r, B)
until b =0

3. Proposed Method

In SCRUB we have used response based knowledge distillation to obey D_f and disobey D_r. We propose a modified version of SCRUB where we use both KL-Divergence based knowledge distillation and Feature based Knowledge Distillation.

Therefore, we propose a modified maximization loss function given by

$$\frac{1}{N_f} \sum_{layers} MSE\left(\frac{F_{i_{w^o}}}{||F_{i_{w^o}}||}, \frac{F_{i_{w^u}}}{||F_{i_{w^u}}||}\right) + \frac{1}{N_f} \sum_{x_f \in D_f} d(x_f, w^u) \quad (6)$$

and a minimization loss

$$\frac{1}{N_r} \sum_{layers} MSE\left(\frac{F_{i_{w^o}}}{||F_{i_{w^o}}||}, \frac{F_{i_{w^u}}}{||F_{i_{w^u}}||}\right) + \frac{1}{N_r} \sum_{x_r \in D_r} d(x_r, w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in D_r} l(f(x_r, w^u), y_r) \quad (7)$$

Using this we define two types of SCRUB:

1. Attention Based: We use multiple feature layers for knowledge distillation which we called as our attention map.

2. Similarity Based: We use the topmost convolution layer for knowledge distillation. This is done with the idea being that the higher layers will contain all the relevant information.

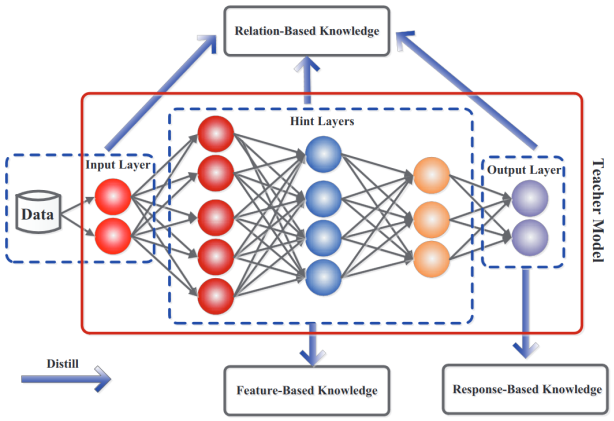

Figure 1. Image showing different types of knowledge distillation

4. Experiments

4.1. Experimental Setup

Our algorithms are implemented in Python 3.7 and use the PyTorch library. All the experiments were conducted on T4 GPU on Google Colab. Our experiments were performed on the Resnet18 convolutional neural network, a state-of-the-art residual learning architecture. We run each experiment with 3 random seeds and report the mean values.
There can be two kinds of unlearning scenarious - *Selective Unlearning* and *Class Unlearning* where any subset of training data and an entire class of training data respectively are forgotten. In our experiments, we consider the most common scenario of selective unlearning and they can be extended to the class unlearning also.

4.2. Unlearning Algorithms

We compare against state-of-the-art approaches as well as various baselines:

1. *Retrain*: Retraining from scratch without the forget set. Assumed to be impractical, included for reference.

2. *Finetuning*: Finetunes the original model on data from the retain set D_r.

3. *NegGrad+*: Similar to Finetuning but also finetunes on data from the forget set D_f, negating the gradient for the latter.

4. *SCRUB Vanilla*: A student-teacher formulation with the student obeying the teacher on D_r and disobeying the teacher on D_f.

4.3. Unlearning for Removing Bias (RB)

In this section, we study unlearning for removing bias in trained models, addressing scenarios such as outdated or in-

correct information. We aim to achieve the highest possible error on D_f without compromising D_r.

4.3.1. Dataset Selection

We utilize the CIFAR-10 dataset for this purpose. The images have 3 colour channels and have a resolution of 32x32 pixels each. From it, we randomly selected 5000 images for training and 1000 images for validation. For selective unlearning, forget set D_f is created using 10% of the training data uniformly chosen across all classes.

4.3.2. Metric

We want our model to achieve high error on D_f without loosing it's utility. Therefore, we use accuracy on D_f, D_r and test errors as our metrics.

4.3.3. Results

The following table gives accuracy on retain set D_r, forget set D_f and test set for different unlearning algorithms

Table 1. Experimental Results

Method	Retain	Forget	Test
Retrain	97.01%	**65.20%**	67.8%
Finetuning	96.82%	93%	64.3%
NegGrad+	96.83%	83.6%	65.2%
Vanilla SCRUB	99.75%	80.12%	65%
Attention SCRUB	**99.81%**	77.20%	66%

From the above table, we can see that all the unlearning algorithms have successfully retained the model's performance on test dataset. However, our proposed model and Vanilla SCRUB perform much better in forgetting the dataset.

4.4. Unlearning for Resolving Confusion (RC)

In this section, we study unlearning for resolving confusion between two classes that the original model suffers from due to a part of its training set being mislabelled.

4.4.1. Dataset Selection

We utilize the tiny-ImageNet dataset, a subset of ImageNet, for this purpose. The images have 3 colour channels and have a resolution of 64x64 pixels each. From the ImageNet dataset, we selected first 20 classes and 500 images each. 90% images for training and 10% images for validation. We mislabelled 100 images each between the classes 0 and 1. Since we want to avoid learning from the mislabelled dataset, this is considered as forget set.

4.4.2. Metric

Given a trained model, we want to forget the mislabelled dataset without compromising performance on D_r and test set. Therefore, just like before we use accuracy on D_f, D_r and test errors as our metrics.

4.4.3. Results

The following table gives accuracy on retain set D_r, forget set D_f and test set for different unlearning algorithms

Table 2. Experimental Results

Method	Retain	Forget	Test
Original	91.16%	51.50%	52.90%
Retrain	93.06%	**0.5%**	54.90%
Vanilla SCRUB	99.82%	43.2%	65.3%
Attention SCRUB	**99.88%**	41.20%	64%
Similarity SCRUB	99.83%	38%	**66.4%**

From the above table, we can see that our unlearning algorithms not only forgets the mislabelled samples, the text accuracy of theses algorithms is much better than retrain or original model.

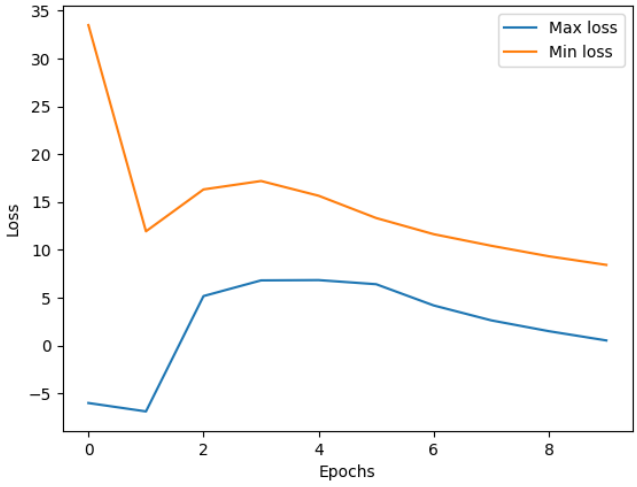

Figure 2. Graph showing the decrease of Min loss and increase in Max loss with epochs during SCRUB unlearning. After a few epochs, we see that the loss functions converge.

4.4.4. EXPLAINABLE RESULTS IN RC

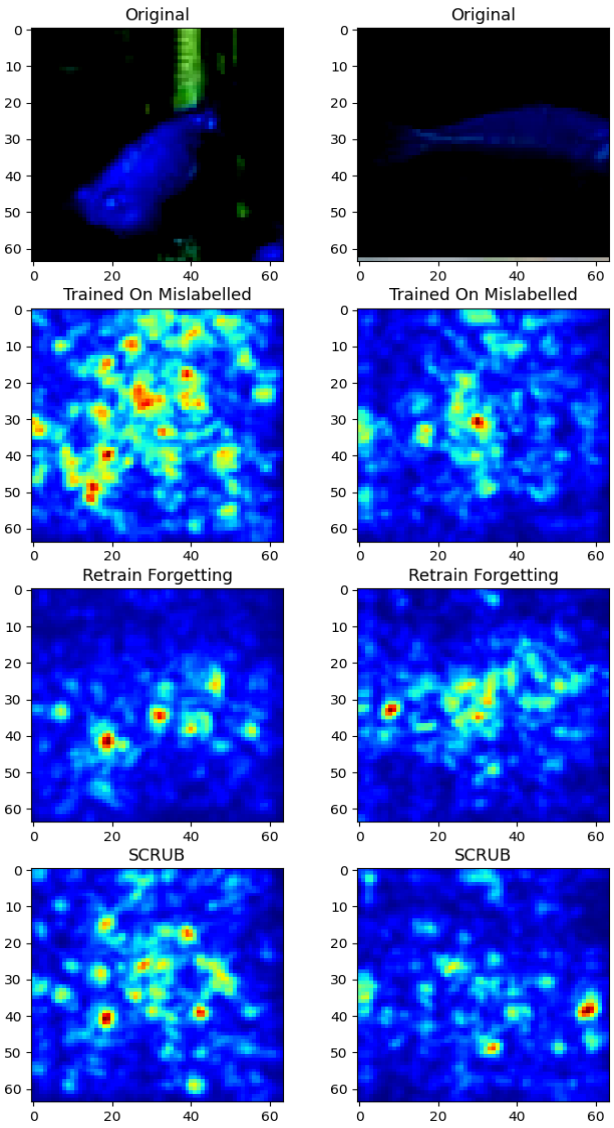

Figure 3. Image showing different types of knowledge distillation

Upon investigating the saliency maps of our proposed methods against the original model on mislabelled classes, we found out that our proposed method focuses on more explainable features. This shows that our model successfully avoids poor knowledge from mislabelled and retains knowledge from correctly labeled data.

5. Conclusion

Unlearning techniques such as SCRUB and modified-SCRUB (Attention and Similarity) successfully avoid the knowledge from the forget set while retaining the knowledge from retain set. Our experiments on resolving confusion show that our proposed methods not only successfully forget the mislabelled dataset but also use the negative knowledge from those points to better generalize the model

Although there is no direct association of unlearning with explainability, we can use various unlearning algorithms to understand the effects of individual classes or data points on the model's knowledge.

6. Future Work

The concept of 'forgetting' varies based on the application, while unlearning holds different priorities and goals across scenarios. For example, in a privacy-focused application, 'forgetting' user data upon deletion requests takes precedence, even if it compromises model performance to protect against membership attacks. On the other hand, in scenarios like removing bias, preserving performance during 'cleanup' operations might be important while erasing traces of old data becomes less significant. We could decide on a notion that satisfies all the needs to the maximum possible extent.

Although there exist various evaluation metrics like classification accuracy, distance to the fully re-trained model eg. KL divergence, error rate of data attacks eg. Membership Inference Attacks, there is not a standardized evaluation metric for machine unlearning.

References

1. Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, Eleni Triantafillou: Towards Unbounded Machine Unlearning

2. Jianping Gou, Baosheng Yu, Stephen J. Maybank, Dacheng Tao: Knowledge Distillation: A Survey

3. Shen Lin, Xiaoyu Zhang, Chenyang Chen, Xiaofeng Chen, Willy Susilo: ERM-KTP: Knowledge-level Machine Unlearning via Knowledge Transfer

4. Laura Graves, Vineel Nagisetty, Vijay Ganesh: Amnesiac Machine Learning

5. Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, Chen Wang: Boundary Unlearning: Rapid Forgetting of Deep Networks via Shifting the Decision Boundary

Preliminary Report: PDF