SAPIENZA
UNIVERSITÀ DI ROMA

M.Sc. Data Science

Machine Unlearning

Statistical Learning

Professor:
Pierpaolo Brutti

Students:
Castaldo Mattia - 1837100
Migliarini Matteo - 1886186
Murawat Nemish - 2056142
Martinez Javier - 2058968

Academic Year 2022/2023

Contents

1 Introduction

Removing a subset of training data in an efficient way is becoming a relevant field of research, some of its main usages are applying *GDPR* regulation (right to be forgotten) and pursue Ethical ML (forget discriminatory information). Furthermore Machine Unlearning can be applied to remove the influence of outliers, and overall it could be useful in explaining why a certain sample is predicted in a certain way.

The easiest way to perform this is of course to retrain the whole model from scratch (*exact unlearning)*, but this would be computationally expensive and time consuming, instead we try to pursue what is known as *approximate unlearning*.

1.1 Problem Formalization

The task can be formalized as such: a model M is trained on the training set D. The training set D is partitioned in a *forget set* D_f and a *retain set* D_r, such that $D = D_f \cup D_r$ and $D_f \cap D_r = \emptyset$.

Consider now the model M_r which is a model retrained from scratch on only the D_r data partition, and consider the model M_u which is the output of our unlearning process. The goal of the unlearning task is to align the performances of these two models on D_f, and D_r.

1.2 Performance Evaluation

How to evaluate the performance of our unlearning methods? We need to evaluate them on 3 different metrics:

- **Performance on** D_r: the performance on the retain set should remain stable, we cannot forget D_f at the cost of forgetting everything else! Also, it's unlikely for M_u to have the same performance on D_r as M because it's using less training data, thus less information. Our goal is for M_u to have a performance on D_r similar to M_r.

- **Performance on** D_f: as stated before, the performance should align to that of M_r, and we can suppose that this performance should also be close to the performance of M on its test set, as it should treat D_f as never seen before data.

- **Time**: the last metric is the complexity and runtime of our algorithm as of course it shouldn't take more time to run than the time to retrain the model from scratch M_r.

We could aggregate this three factors in one metric as done by Kaggle for its competition, and also we could use Membership Inference Attacks (MIAs) to test our M_u against D_f.

1.3 Class- vs Random-Subset Unlearning

In the papers we read the unlearning target wasn't always defined in the same way, sometimes papers referred to the unlearning target as whole classes sometimes as single data points. We here formalize this as:

- **Class unlearning**: refers to forgetting a subset of classes $\{C_1, ..., C_n\}$ previously learned by a classification model. For example removing the class "Dog" from a model distinguishing among pictures of animals.
- **Random-Subset unlearning**: refers to forgetting a particular subset of data points $\{x_1, ..., x_n\}$ which have already been learned by a model. For example, removing a particular users's data who has asked to be forgotten (GDPR)

The methods we test and explore are sometimes only applicable to one of these two unlearning-types.

2 Finetuning

This will be our baseline for approximate unlearning. This method is very simple as it requires only to train again the model for a few epochs but only with D_r.

In fact we're going to exploit the process known as *catastrophic forgetting*, which happens to the parameters of a DNN when they aren't trained on a specific sample for a while.

As it can be seen in 12 after many epochs the performance on D_f degrades and starts aligning withe the performance on the test set.

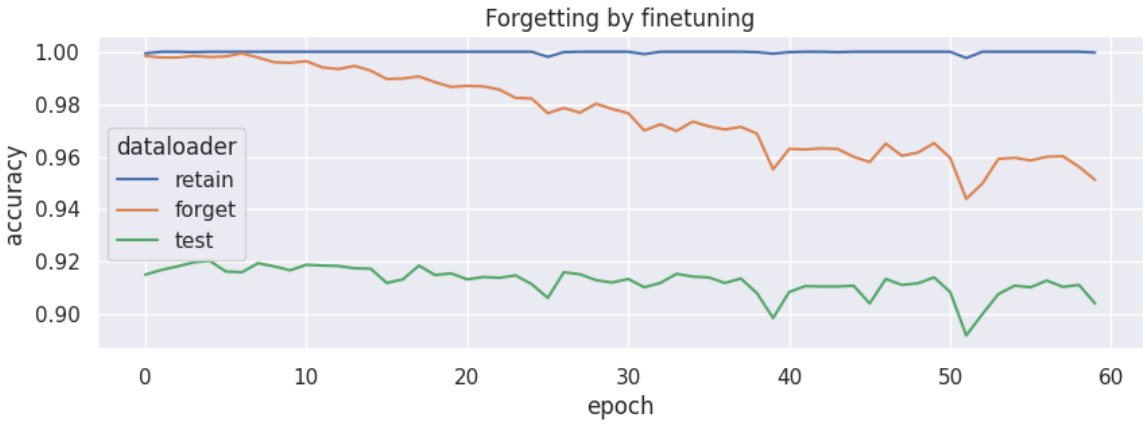


Figure 1: Finetuning on D_r. $lr = 10^{-5}, \lambda = 0.01$

This model is admittedly very simple, and it's performances are lacking due to the high number of epochs needed to forget D_f and due to the risk of overfitting the model on D_r (which can be seen on 12).

2.1 As a second-step

Still this model is used in all the next ones as a **second step**. In fact all models that we're going to evaluate from now on are composed by a:

- **Impair step**: in which the model parameters are somehow perturbed, and thus the performance is highly impacted.
- **Recovery step**: in which the model is fine-tuned on D_r, which is exactly the baseline we've considered now.

3 Random Perturbation Unlearning

This is a simple algorithm which is just a slight improvement over the standard Finetuning algorithm.

The concept is that we're going to add some Gaussian distributed noise to all the parameters $N(0, \beta\sigma)$ such that the model **parameters will be moved out of the local optima**. Note that in this case σ is the standard deviation of a neural network layer computed by considering it's learned parameters and $\beta \in (0, 1]$ is an hyperparameter that increases the strength of the noise.

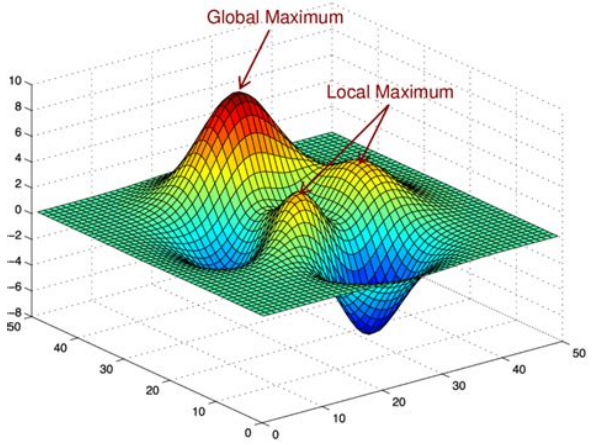


Figure 2: Local optima

Then one or two epochs of finetuning with the retain set D_r are hopefully going to bring the model to a new local optima, one in which the model hopefully forgot D_f.

One of the peculiarities of this algorithm is that it's one of the few methods capable of forgetting without the need of using the D_f, so that it's well suited for those cases in which it's not feasible to fetch the forget set.

3.1 Performance

We have to note that this algorithm is very fast, as the perturbation phase is $O(N)$ in the number of learned parameters.

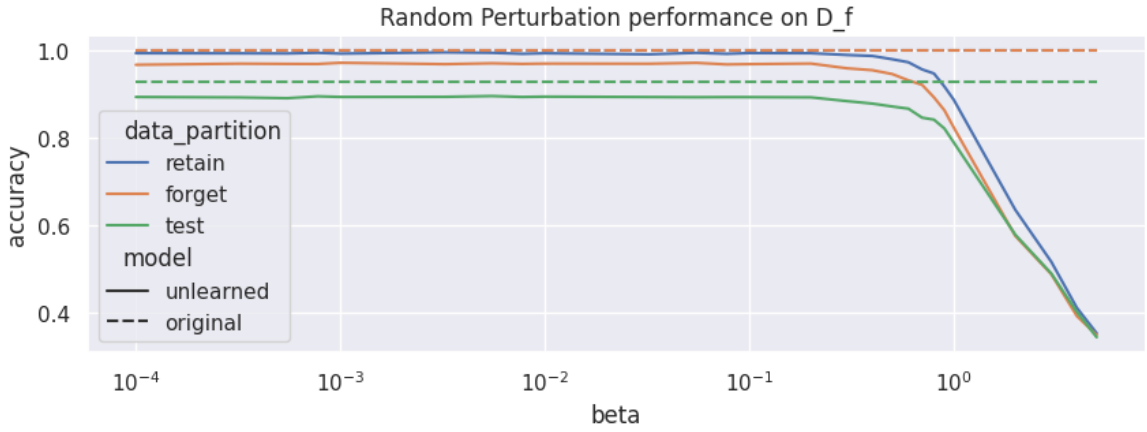


Figure 3: Performance for different β values, evaluated on 5 finetuning epochs

When considering the performance of this algorithm with varying betas we can see how the performance decreases (as expected) with the increase of β, and we also notice how the D_{test} and D_r performances remain close enough to the original performance, while the D_f performance is closer to the D_{test} performance, even though not by much.

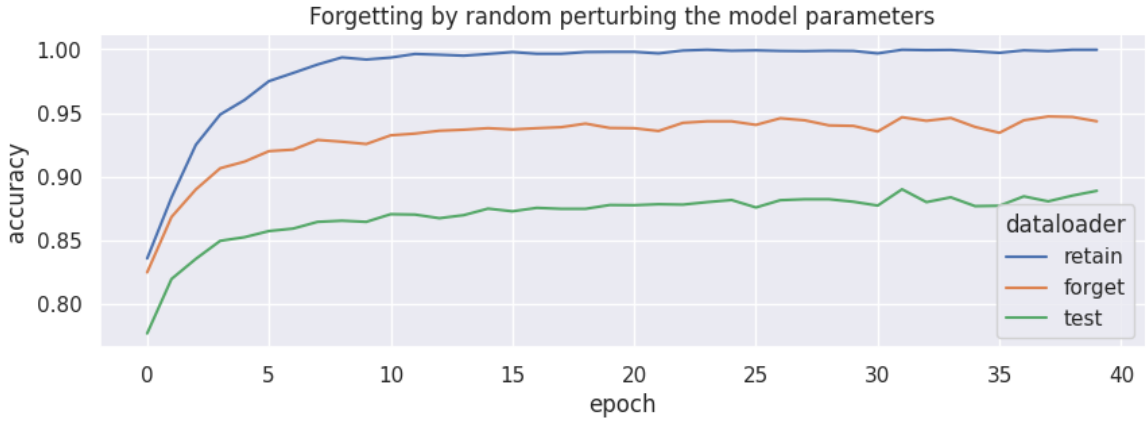


Figure 4: Accuracy for the noise-perturbed model for each retrained epoch. $\beta = 0.7, lr = 10^{-5}, \lambda = 0.01$

When considering how this method compares to the *Only-Finetuning* method we can see how the performance on D_f goes down way faster, but then it stalls, compared to 12 we conclude that this method is more efficient.

Although this algorithms is very simple, it's still able to perform quite well, and in the Kaggle competition it scores surprisingly good **??**.

Score
0,0152
0,0065

4 Stochastic Teacher Network

Based on the work of Zhang et al., 2023, [1] our initial methodology consists of two primary steps:

1. We begin by using a randomly initialized network to assist the original model in eliminating the influence from the dataset D_f. This process results in an unlearning model called M_u, which performs similarly to a random classifier when it comes to the target D_f.

2. Subsequently, we work on reconstructing the new model Mu so that it can maintain good performance on the remaining data points.

Our expectation and hope are that the unlearning model M_u will preserve the knowledge related to D_r while removing the knowledge from D_f.

4.1 Knowledge erasure

We initialize a stochastic network M_s that shares the exact architecture as the original model but does not contain any knowledge from D_f, causing its classification results to resemble those of a random classifier. We leverage the concept of knowledge distillation and utilize the weights of the original model to initialize M_u. M_s is essentially considered the teacher network, while M_u serves as the student network. Our goal is to make the probability distribution of M_u similar to that of M_s when it comes to the target data D_f. Denoted a sample as $x_i \in D_f$, we pass this data point through both models, M_s and M_u. By applying the softmax function, we obtain two probability distributions $p(x_i)$ and $q(x_i)$. Subsequently, we adjust the parameters of M_u with the aim of reducing its knowledge of D_f, achieved by minimizing the Kullback-Leibler Divergence (KLD) between the two models.:

$$L_{KL}(p(x_i)||q(x_i)) = \sum_{i=1}^{n} p(x_i) log \frac{p(x_i)}{q(x_i)}$$

4.2 Model reconstruction

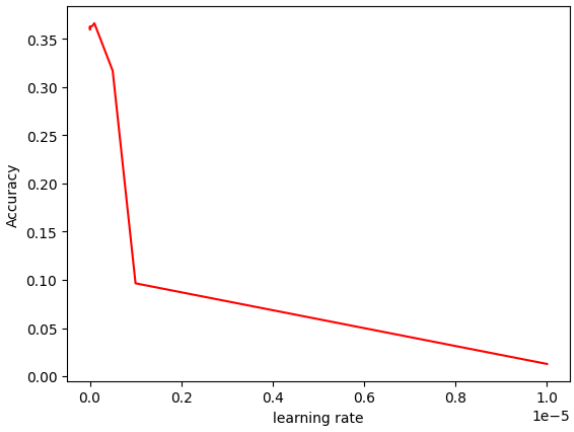
Since we modified the parameters of M_u in the previous step, this may have an impact on M_u's classification performance regarding D_r. To address this, we utilize the data from D_r and the original model that was trained on the combined dataset $D_r \cup D_f$. The original model, M_d, is considered a teacher network, while M_u remains the student network. Now given a sample $x_i \in D_r$ we compute the KLD between M_d and M_u and at the same time the cross entropy loss of M_u on D_r. This results in a final loss function:

$$L = L_{CE} + \alpha L_{KL}$$

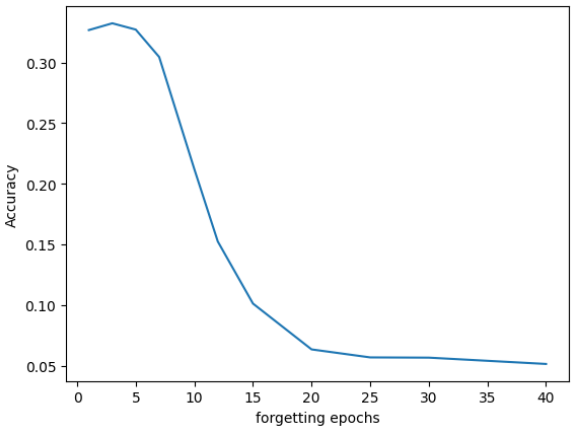
Here, α is a hyperparameter that needs to be tuned. Over a few epochs, this combined loss function should help M_u regain its performance on D_r.

4.3 Evaluation

Since this method is only valid for the Class Unlearning task, we focus on unlearning just one single class from the original model. We have to take into account two important parameters for this procedure: the number of forgetting epochs performed and the learning rate at the first stage.



(a) Accuracy decay as increasing the lr on D_f



(b) Accuracy decay as increasing the epochs on D_f

As we can see from the figures , the model becomes a random classifier with respect to the forgotten class only with a very small learning rate, around 10^{-8}, and after performing a considerable number of epochs (15). These two values can vary significantly depending on the number of samples to forget and the difficulty of the task itself. It is safe to say that increasing the learning rate can reduce the number of epochs required to achieve randomness for that class, but it also introduces numerical instability to the procedure. As the number of epochs and the learning rate increase, the accuracy of the unlearned model for the forgotten class approaches 0. This behavior is a result of the nature of the Kullback-Leibler Divergence (KLD) loss. Continuing the unlearning, the model learns to make poor predictions for data associated with that specific class, this aspect needs to be prevented since the goal is to eliminate any knowledge from the original model rather than making the model incapable of recognizing samples from that class.

For what concern the reconstruction step, the only parameter that significantly impacts the process is α. It's noticeable that as alpha increases, the retraining on D_r becomes a bit faster.

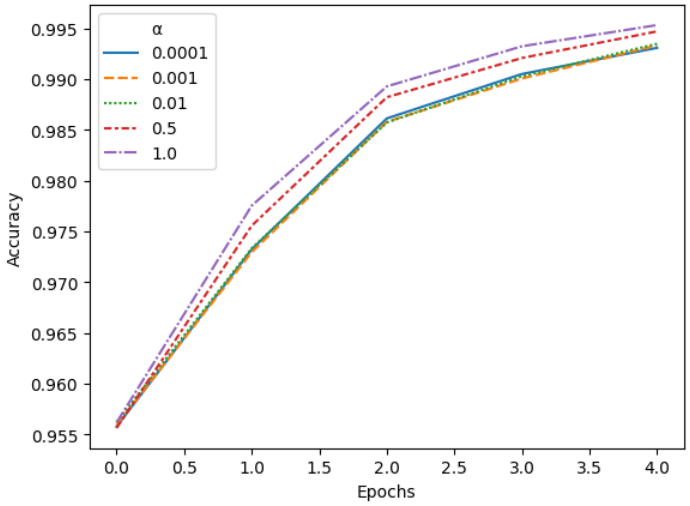


Figure 6: Reconstruction stage for different α values. $lr = 10^{-6}$, $epochs = 15$

After choosing $lr = 10^{-6}$ and $epochs_{forget} = 15$ the unlearned model is able to reach its original performance on D_r and test data.

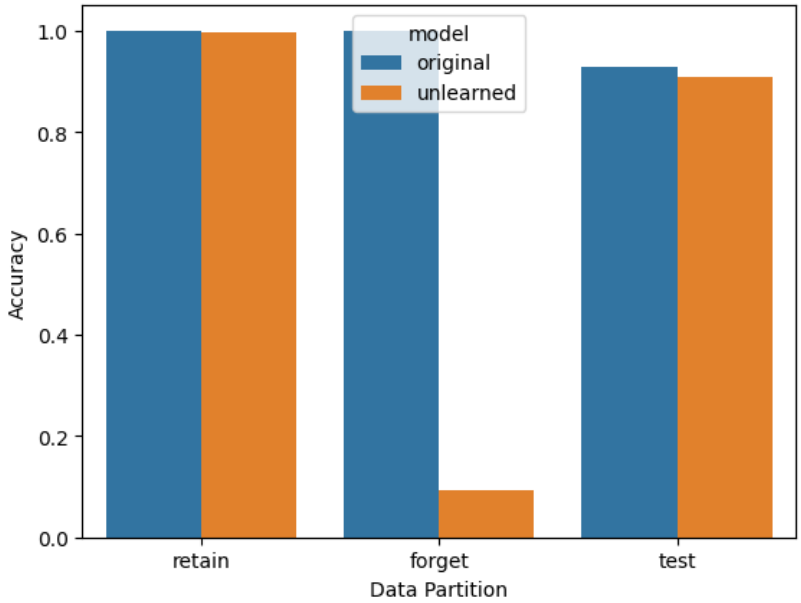


Figure 7: Comparison between accuracy of unlearned model and original model for each data partition. $\alpha = 0.5$, $lr = 10^{-6}$, $epochs = 15$

5 Fisher Masking

The limitation of the fine-tuning is not just slow unlearning of the classes as evident in the 8 but also the incapability to remove class-specific information from the learned model. In the figure below presented in the works of [2] we can clearly see that the fine-tuning method is not able to remove the information of the forget class. It still maintains an accuracy of 40 % on the Forget Dataset.

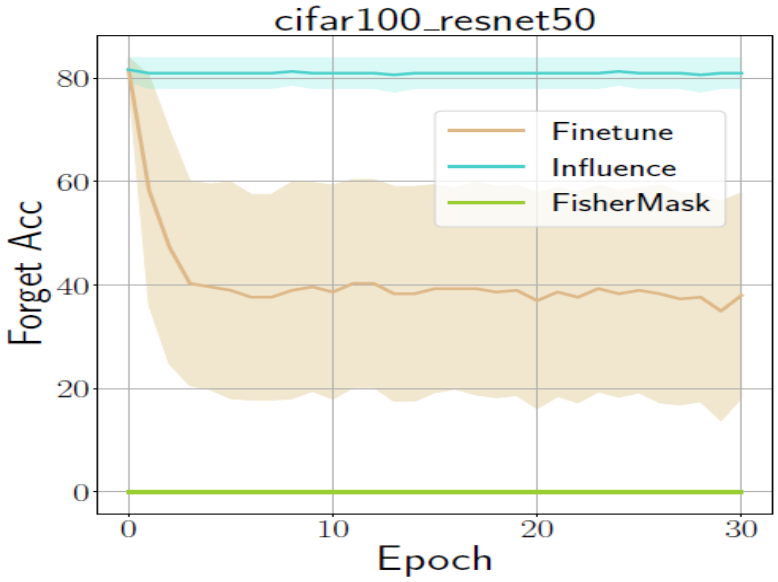


Figure 8: Finetuning Ineffectiveness on Resnet-50 trained on CIFAR-100

The main motivation in the works of [2] is that instead of random perturbation, the perturbations should be biased towards the task of removing the **Forget data** artefacts in the network. For this, they employed the strategy of computing the Fisher-Information matrix for the D_f and D_r. This helps us to compare the importance of the parameters between the two datasets.

5.1 Fisher Masking Strategy

For a distribution $p(y|x, w)$, the Fisher information matrix and it's empirical form is given by:

$$F \triangleq \mathbb{E}_{x,y}\left[\nabla_w \log p(y|x,w) \nabla \log p(y|x,w)^T\right] \approx \frac{1}{|D|}\sum_{i=1}^{|D|} \nabla_w \log p(y_i|x_i,w) \nabla \log p(y_i|x_i,w)^T$$

When we move towards the regime of Deep neural networks computing the whole matrix for the entire network is infeasible. Instead, in the prior works of [3] the full **F** have been estimated with its $Diag(F)$. In such case, the **F** is given by the expectation

of the second derivative of loss.

$$F = -\mathbb{E}_{x,y}\left[\nabla_w^2 \log p(y|x,w)\right]$$

Note we don't have to explicitly compute the second derivative, we can leverage the first-order gradients stored in the trained network and employ the empirical form to compute **F**.

5.1.1 Masking Steps

1. Compute the two Fisher Information Matrix(FIM) F_r and F_f using D_f and D_r datasets.

2. Intuitively, the masking strategy should perturb the parameters which have more information with respect to the Forget Set rather than the Retain set. For this, we compute $F_{f,jj} - F_{r,jj}$, where $F_{f,jj}$ represents how j^{th} parameter gives information with respect to forget Data.

3. We select the top R % parameters and zero them out.

   We want to point out that R here will be considered a Hyperparameter as this would depend on the difficulty of the Dataset and the model used.

5.2 Evaluation

5.2.1 Class Unlearning

In the class Unlearning task the goal is to makes Trained Model Forget about a specific class of concern.

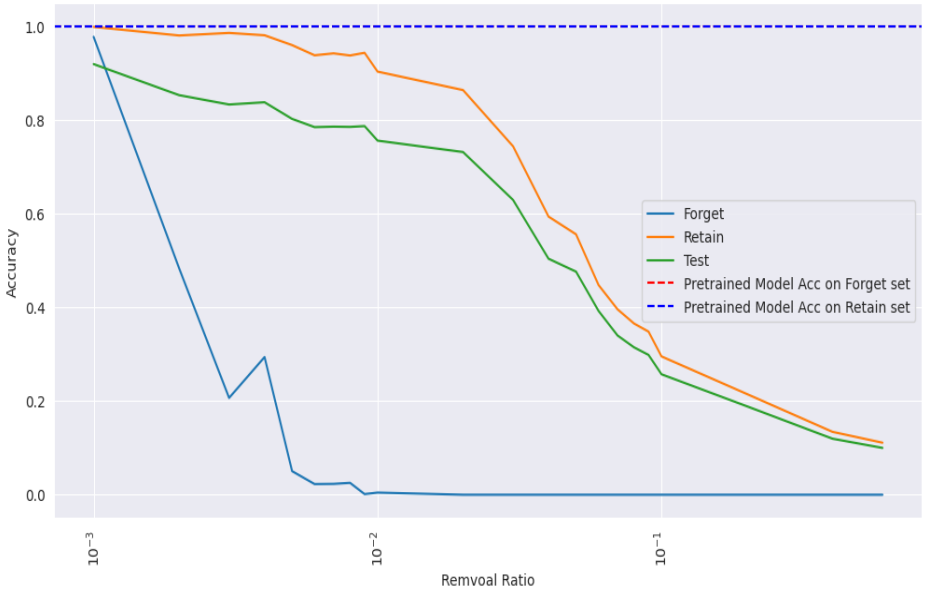


Figure 9: Effect of R on the performance of the Unlearned Model on three datasets

We can see from R=0.01 that the model is able to completely forget the samples from the **forgot class**. Thought important thing to note is the slight drop in performance on the retain and test dataset.

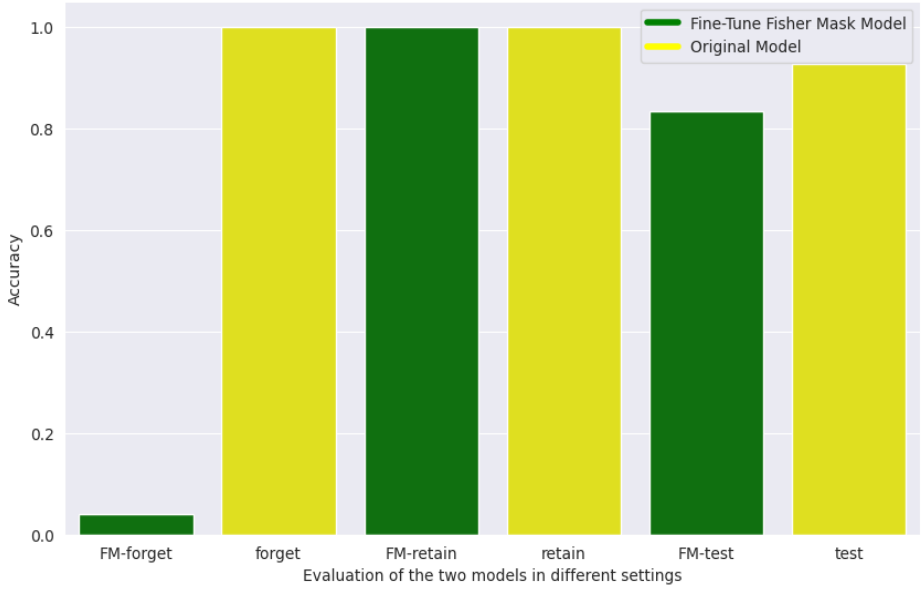


Figure 10: Comparison between Unlearn and Original Unlearned Model Finetune for 5 epochs with Lr=1e-5,L2=0.01

After choosing R=0.01 and fine-tuning for 5 epochs on the retain dataset, the unlearned model is able to reach its original performance on Retain and Test. We would like to point out the efficiency of the method in comparison to fine-tuning. Note, that a drop of around 10% in Test performance is expected due to the unlearning of a class.

Figure 11: Activation Maps obtained from Original and Unlearned Model on a Test sample Predictions- Original:Automobile Unlearned:Truck

We can see the activation maps visualised for the forget sample using a Pre-trained model and an Unlearned model is completely different. The activation map coming

from the unlearned model does not focus on the object of concern. This highlights the capacity of the unlearning model to forget representation for the forget Class.

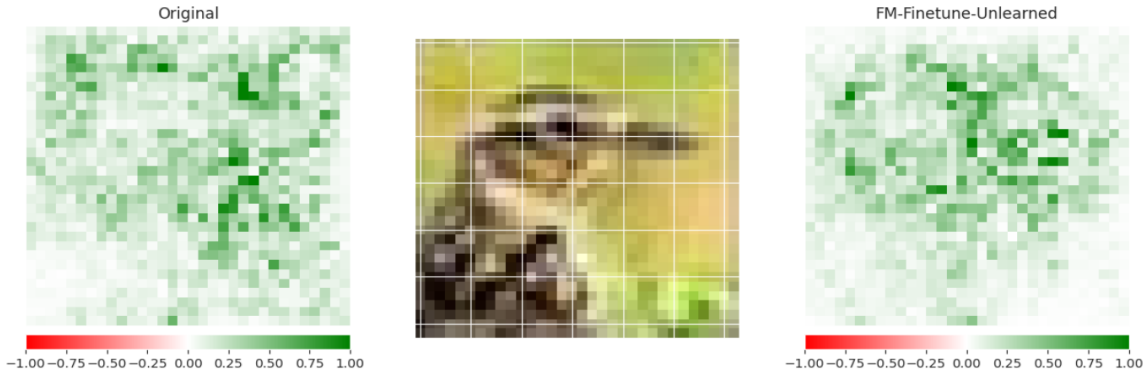
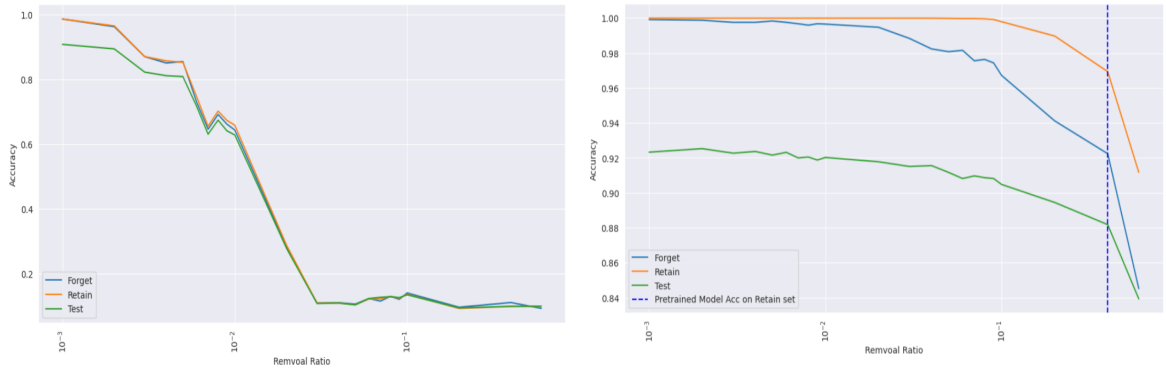


Figure 12: Activation Maps obtained from Original and Unlearned Model on a Test sample Predictions- Original:Bird Unlearned:Bird

We can see the activation maps visualised for the retain sample using a Pre-trained model and an Unlearned model is quite similar. The higher activations are received by the pixels inside and close to the frog. This highlights the capacity of unlearning model to retain representation for Retain Class.

5.2.2 Random Sample Unlearning

Random Sample Unlearning becomes a more challenging task when compared to Class Unlearning. The goal of this evaluation would be that the properties of the **Unlearned Model** should be close to the retrained model. In search of appropriate Hyperparameters can also become tricky when one abstains from using methods such as Membership Inference attacks(in our case due to computational infeasibility).



(a) D_r, D_f and D_{Test} performance of FM for Random Samples Unlearning No fine tuning Performed

(b) D_r, D_f and D_{Test} performance of FM for Random Samples Unlearning 5 epochs fine tuning Performed

Figure 13: Selection of Appropriate R

In order to select Appropriate R we investigated two plots given in 13. The ideal unlearned model should have close performance on **forget Set** when compared with **Test Set**. The Performance should be more optimistic on **Retain Dataset**. Upon investigation, 13a R values greater than 0.01 seem optimistic as they have similar performance on all three sets. Since we do 5 epochs of fine-tuning this observation changes and we can see from the accuracy curves. R values less than 0.1 still shows dominating performance on the forget-set indicating towards inefficient unlearning.

Finally, in this case, we chose $R = 0.4$ as Hyperparamter setting.Finally we use two measures for evaluating the effectiveness of Unlearning.

1. Activation distance: This is an average of the L2-distance between the unlearned model and retrained model's predicted probabilities on the forget set. A lesser activation distance represents better unlearning.

2. JS-Divergence: JSDivergence between the predictions of the unlearned and retrained model

Method	Activation Distance(L2 Average)	JS Divergence
Fisher Masking	0.11	0.07

Table 1: Unlearning Performance

6 Selective Synaptic Dampening

This approach follows the main principle of utilizing the Fisher Information matrix as presented in the section 5. The selection criterion as argued by the [4] is too strong as it leads to the degradation of performance on D_r. In the fisher masking approach we also remove parameters which have high overlap of importance across the two datasets (D_f and D_r).

In order to alleviate the weakness in the selection of parameters such that these have high importance for D_f and less importance for D_r they first computed the Fisher Importance Diagonal Matrix [] and then modified the selection criterion in the following way:

$$\beta = \min\left(\frac{\lambda \cdot D_{r,i}}{D_{f,i}}, 1\right)$$

$$\theta_i = \begin{cases} \beta\theta_i, & \text{if } []D_{f,i} > \alpha[]D_{r,i} \\ \theta_i, & \text{if } []D_{f,i} \leq \alpha[]D_{r,i} \end{cases} \text{ for all } i \in [0, |\theta|]$$

There are mainly two modifications in comparison to the Fisher Masking criterion. Firstly instead of making the weights zero, they dampen the original weights by a factor of β. The dampening factor is given by the ratio between Fisher's importance for the retain dataset and the forget dataset. Here λ is a hyper-parameter to control the level of dampening. Intuitively, we chose to go with a value of one such that $\beta < 1$ for all parameters that are specialized towards D_f Therefore, $\beta \to 0$ as a parameter becomes more specialized for D_f.

Secondly, instead of arbitrarily selecting top R % parameters they employ a threshold determined by the factor α. This step facilitates the identification of parameters that are highly specialized towards samples in the forget set, with α dictating how specialized they must be to be dampened.

6.1 Algorithm

Algorithm 1: Selective Synaptic Dampening

Input: Input: ϕ_θ, D_r, D_f; optional to skip 1.: $[]D_r$

Data: Parameter: α, λ

Output: Output: $\phi_{\theta'}$

Calculate and store $[]D_r$.

Calculate $[]D_f$ **for** *i in range* $|\theta|$ **do**

  if $[]D_f, i > \alpha[]D_r, i$ **then**

    $\beta = \min\left(\frac{\lambda \cdot D_{r,i}}{D_{f,i}}, 1\right)$;

    $\theta'_i = \beta\theta_i$

return $\phi_{\theta'}$;

6.2 Evaluation

6.2.1 Class Unlearning

In the class Unlearning task the goal is to makes Trained Model Forget about a specific class of concern.

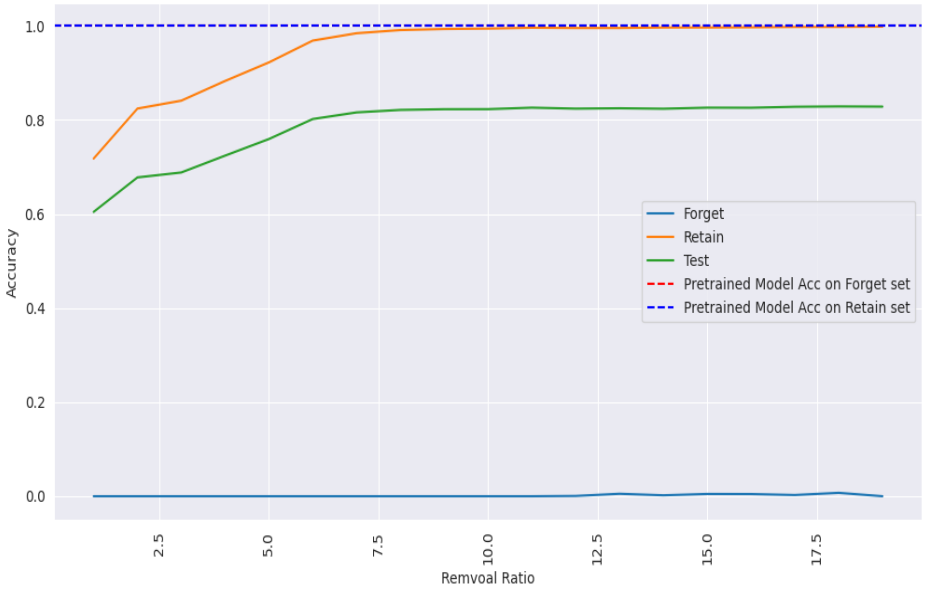


Figure 14: Effect of Alpha on the performance of the Unlearned Model on three datasets

We can see from $\alpha = 10$ that the model is able to completely forget the samples from the **forgot class**. In comparison to **Fisher Masking** strategy, we can clearly see the performance of the unlearned model is maintained on the retained set and the generalisation on the test. Note, that a drop of around 10% in Test performance is expected due to the unlearning of a class.

6.2.2 Unlearning Interpretation by Saliency Maps

Saliency maps are a tool that uses that use backpropagation and neuron activation in order to highlight which parts of an image are the most important for the prediction of a certain label. We compared such activations between the original and the unlearned model and we will present this as a heatmap.

Figure 15: Activation Maps obtained from Original and Unlearned Model on a Test sample Predictions- Original:Automobile Unlearned:Truck

We can see the activation maps visualised for the forget sample using a original model and an Unlearned model is completely different. The activation map coming from the unlearned model does not focus on the object of concern. This highlights the capacity of the unlearning model to forget representation for the forget Class.

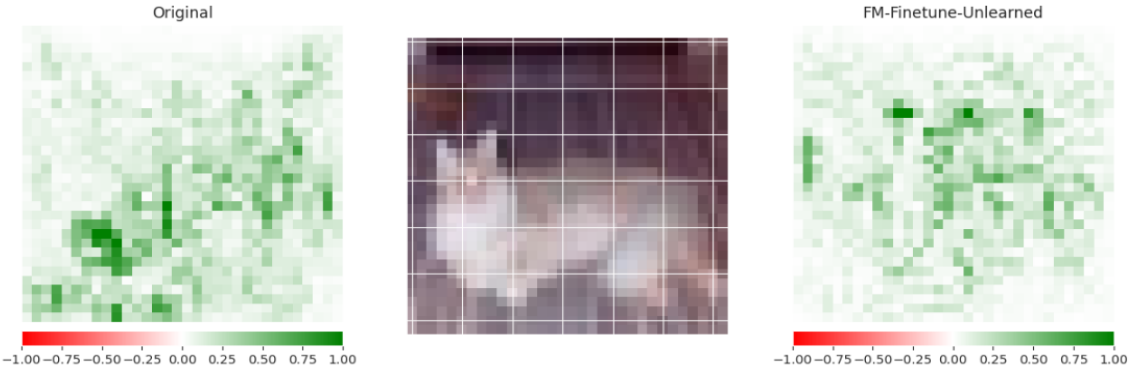
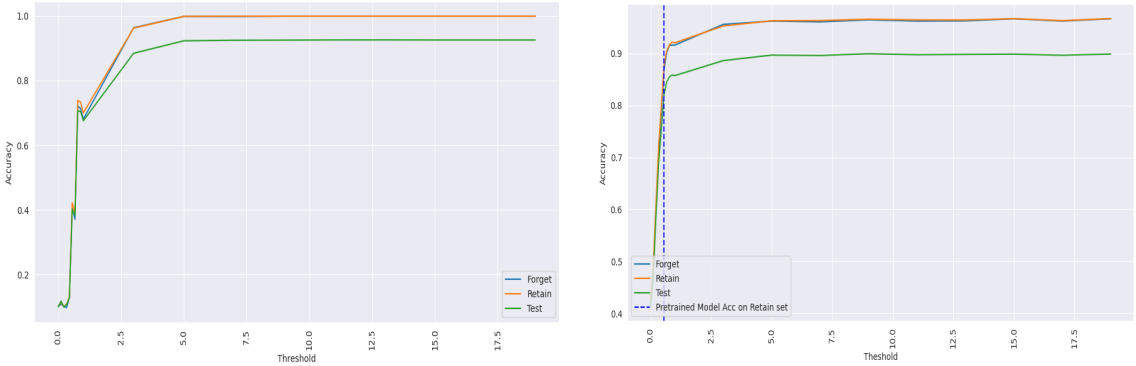


Figure 16: Activation Maps obtained from Original and Unlearned Model on a Test sample Predictions- Original:Cat Unlearned:Cat

We can see the activation maps visualised for the retain sample using a original model and an Unlearned model is quite similar. The higher activations are received by the pixels inside and close to the cat. This highlights the capacity of unlearning model to retain representation for Retain Class.

6.2.3 Random Sample Unlearning

Random Sample Unlearning becomes a more challenging task when compared to Class Unlearning. The goal of this evaluation would be that the properties of the **Unlearned Model** should be close to the retrained model.



(a) D_r, D_f and D_{Test} performance of FM for Random Samples Unlearning No fine tuning Performed

(b) D_r, D_f and D_{Test} performance of FM for Random Samples Unlearning 5 epochs fine tuning Performed

Figure 17: Selection of Appropriate R

In order to select Appropriate Alpha we investigated two plots given in **??**. The ideal unlearned model should have close performance on **forget Set** when compared with **Test Set**. The Performance should be more optimistic on **Retain Dataset**. Upon investigation, 17a alpha values less than 1 seem optimistic as they have similar performance on all three sets. Since we do 5 epochs of fine-tuning these observation changes a bit and we can see from the accuracy curves. Alpha values greater than 0.55 still shows dominating performance on the forget-set indicating towards inefficient unlearning.

Finally, in this case, we chose $alpha = 0.5$ as Hyperparamter setting.Finally we use two measures for evaluating the effectiveness of Unlearning.

1. Activation distance: This is an average of the L2-distance between the unlearned model and retrained model's predicted probabilities on the forget set. A lesser activation distance represents better unlearning.

2. JS-Divergence: JSDivergence between the predictions of the unlearned and retrained model. A less JS Divergence tells us the distribution of outputs for the Unlearned and Retrained Models are close.

Method	Activation Distance(L2 Average)	JS Divergence
SSD	0.19	0.11

Table 2: Unlearning Performance

7 Conclusions

In this study, we have explored and compared several methods for machine unlearning. The main goal of our comparative work was to find the most effective method for mitigating the influence of both specific data points or classes within a machine learning model without compromising its performance on the retained data. Through a comprehensive analysis, we implemented and evaluated different unlearning techniques, including, and measured their impact on the model's accuracy and training time.

The most simple methods are for sure the Finetuning2 and the Random Perturbation3. While the first one is ineffective, if not after a long number of epochs of retraining, the second one has some surprisingly good results when considering the simplicity of the method. Although naive, the Random Perturbation is still able to somehow forget D_f both in class- and random-subset unlearning settings. In fact it can even be competitive in the Kaggle competition**??**, thanks to the speed of its perturbation phase, which doesn't require the computation of any gradient.

Since the Stochastic Teacher Network4 has no guarantees for random-subsample unlearning, it is only possible to evaluate it for the Class Unlearning task. The procedure achieves good results on the forget set (10%) and manages to reach a performance quite similar to that of the original model on the retain set. The most relevant limitation of this method is related to the number of epochs performed at the first stage of the algorithm: as the size of the forget set increases, the number of epochs necessary to achieve a random classifier for that class also increases. This is not a problem for datasets like CIFAR-10 because they contain only 50,000 samples (5,000 per class). However, when dealing with larger datasets that have millions of samples, this algorithm becomes computationally too expensive. Apart from computational issues of this method, it is important to say that this model cannot provide any guarantees regarding random sample unlearning. The goal of unlearning tasks is to make the probability distributions of samples from the retrained model from scratch and the unlearned one close. However, in the last step, we are adding the Kullback-Leibler Divergence (KLD) between the original model and the erased one. No one guarantees to us that the two distributions on the retained set of the retrained model and the original model are the same. That's the reason why we think that this method strongly depends on the dataset and the difficulty of the learning task.

On the other hand, Fisher Masking5 showed some impressive improvements with respect to previous methods. For the case of class unlearning, without fine-tuning the model still maintains a good generalization ability which is brought to the original level by 5 epochs of fine-tuning. In the case of Synaptic Systematic Dampening6, without

doing any fine-tuning epochs, the model is able to maintain its original generalisation level.

In the case of Random Sample Unlearning, both FM and SSD methods are able to erase the influence of the forgotten dataset and are quite close to the retain model based on Activation Distance and JS-Divergence metric. Further FM, seems to perform better than SSD based on the given metrics. This relation might not hold true, we need a better metric such as Membership Inference Attacks(MIA) accuracy to verify it. Due to high computational requirements, we weren't able to perform it.

Overall, this study humbly contributes to the growing field of machine unlearning by providing a systematic evaluation of various techniques. As machine learning models become increasingly integral to various applications, the importance of effectively handling unlearning becomes paramount.

References

[1] Xulong Zhang, Jianzong Wang, Ning Cheng, Yifu Sun, Chuanyao Zhang, and Jing Xiao. Machine unlearning methodology base on stochastic teacher network, 2023.

[2] Yufang Liu, Changzhi Sun, Yuanbin Wu, and Aimin Zhou. Unlearning with fisher masking, 2023.

[3] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. *CoRR*, abs/1911.04933, 2019.

[4] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening, 2023.

[5] Leslie Lamport. *LaTeX: a Document Preparation System*. Addison Wesley, Massachusetts, 2 edition, 1994.

[6] Yongjing Zhang, Zhaobo Lu, Feng Zhang, Hao Wang, and Shaojing Li. Machine unlearning by reversing the continual learning. *Applied Sciences*, 13(16):9341, Aug 2023.