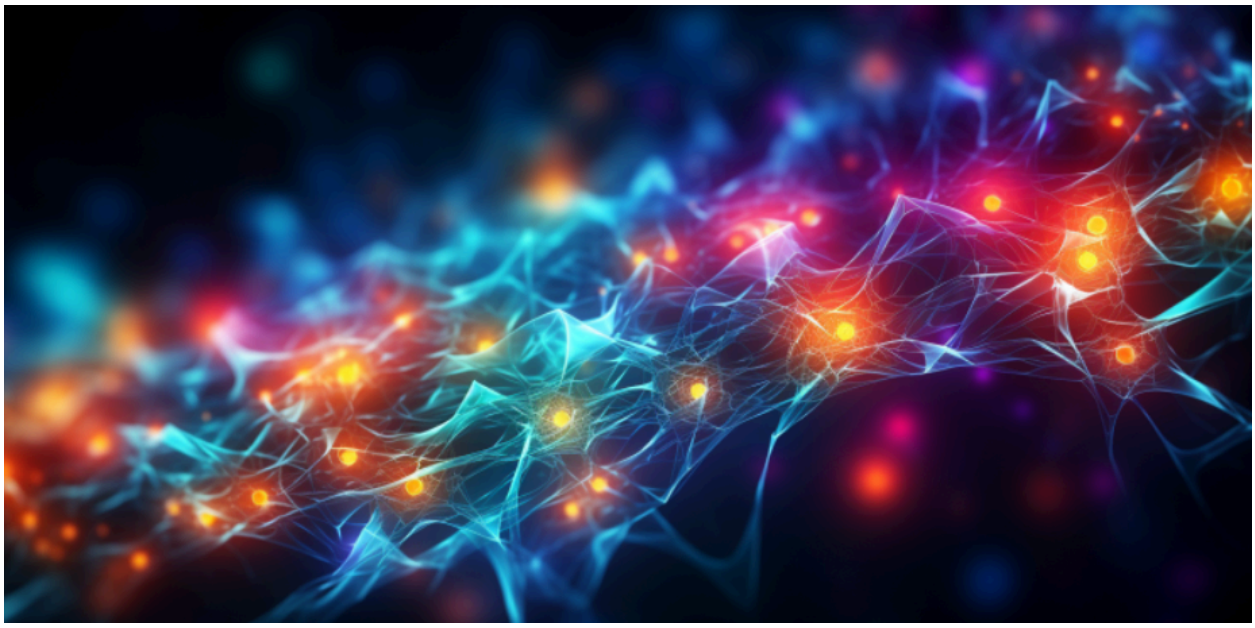
CS354 Minor Project

Machine Unlearning

End Semester Evaluation



Github - https://github.com/10isha/CS-354-Machine-Unlearning

Team Members -

Tanisha Sahu , Roll No. 210001071

Prajakta Darade , Roll No. 210001052

Summary

The project is based on the development of an efficient architecture to "unlearn" machine learning models on a certain subset of data after they have been trained on a larger dataset.The primary focus of this project is to remove image specific features which are learnt by the models for privacy reasons while keeping the generalization capabilities of the model intact. In this project we explore and develop two different kinds of unlearning algorithms, perform extensive experimentation and evaluate their "Unlearning Scores". After evaluating the algorithms on different parameters we conclude that our first architecture serves to perform better than the second with an unlearning score of **0.1616.**

Title and Problem Definition

Machine unlearning is an emerging area of research within the field of machine learning that focuses on developing methods to reverse or modify the learned behaviors of trained models. This concept is motivated by the need to address various challenges such as data privacy concerns, model adaptability in changing environments, and mitigation of biases and unfairness in AI systems. Unlike traditional machine learning, where the emphasis is on learning from data to improve model performance, unlearning involves strategies to selectively forget or update specific knowledge within a model while preserving its overall functionality.

The process of machine unlearning encompasses the design and implementation of algorithms that allow models to adaptively adjust to new data, forget outdated information, or modify learned behaviors based on evolving requirements. Techniques such as fine-tuning, regularization, and specialized loss functions are explored to achieve effective unlearning without compromising the model's performance. Challenges in this area include defining optimal unlearning strategies, ensuring minimal impact on model accuracy and generalization, and developing robust evaluation metrics to quantify the success of unlearning methods. Overall, machine unlearning represents a critical aspect of advancing AI systems towards greater adaptability, privacy, fairness, and sustainability in real-world applications.

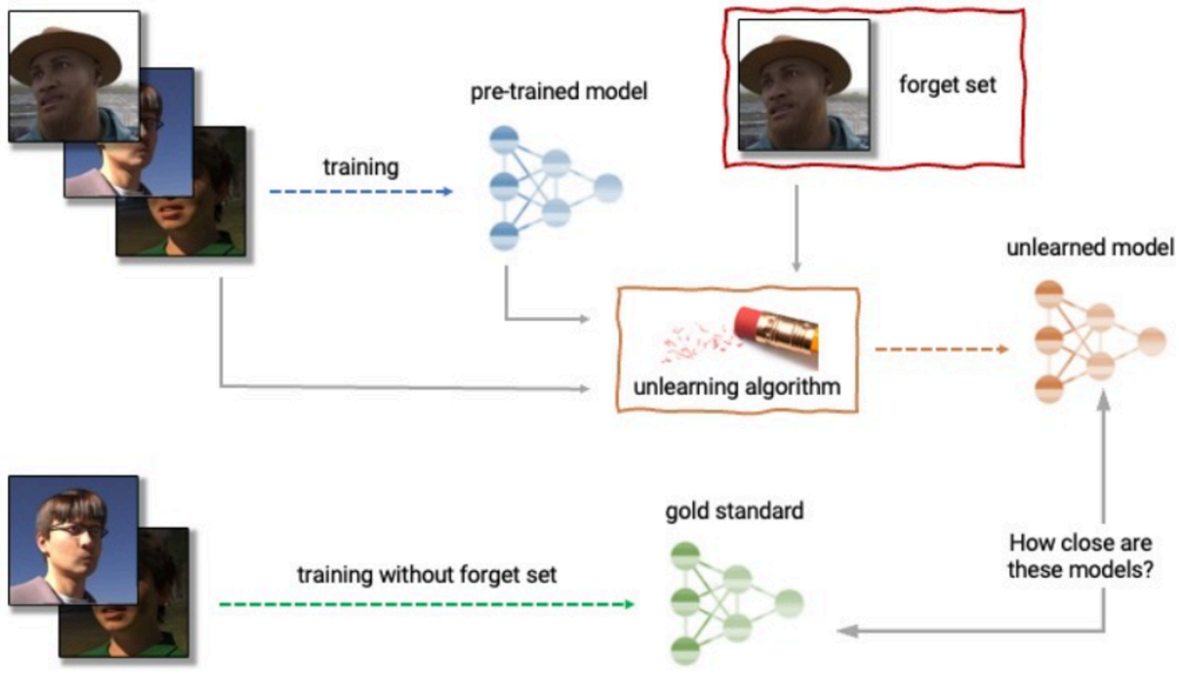
Motivation

Completely eliminating the impact of data marked for deletion presents significant challenges, as it involves more than just removing it from stored databases; it also requires nullifying its influence on other elements like trained machine learning models. Recent studies have indicated that it might be feasible to accurately infer whether a sample was used to train a machine learning model using membership inference attacks (MIAs). This discovery raises privacy concerns since it suggests that even after an individual's data is deleted from a database, it could still be possible to discern whether that individual's data contributed to training a model.

In light of these considerations, machine unlearning has emerged as a subset of machine learning dedicated to erasing the influence of specific subsets of training examples, known as the "forget set," from a trained model. An optimal unlearning algorithm would eliminate the influence of certain examples while preserving other desirable properties, such as accuracy on the remaining training set and generalization to new examples. One approach to achieve this is retraining the model on a modified training set that excludes the samples from the forget set. However, this method is not always practical due to the computational expense associated with retraining deep models. Ideally, an effective unlearning algorithm would leverage the existing trained model as a foundation and efficiently adjust it to nullify the impact of the designated data.

Analysis and Design

Project Pipeline



Initially our dataset (Cifar-10) is divided into 3 parts which are the Train set, Validation set and the Test Set with 50k Images, 5k Images and 5k Images respectively.

The Train set is then further divided into a retain set with 45k Images and a forget set with 5k Images.

The data in the train set is then used to train our base model architecture which is **resnet18**.

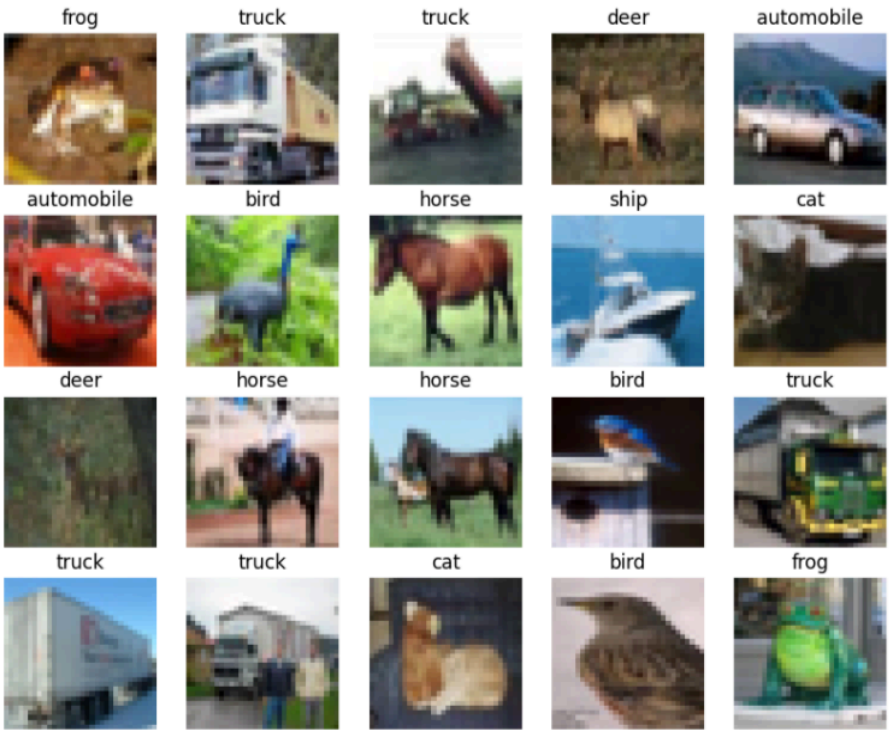
After training , we utilize our unlearning algorithm to "untrain" the model on the forget set while retaining the features it has learnt from the "retain set".

Finally after the unlearning process is complete we evaluate the model using the various metrics presented such as MIA (Membership Inference Attacks, Gold Standard, Forgetting Quality and Unlearning Score).

Data analysis and Preprocessing

Introduction to Cifar-10

The CIFAR-10 dataset is a widely used benchmark in the field of machine learning and computer vision. It consists of 60,000 color images (32x32 pixels) across 10 classes, with 6,000 images per class.



Dataset Overview

Total Images: 60,000

Image Dimensions: 32 pixels x 32 pixels x 3 channels (RGB)

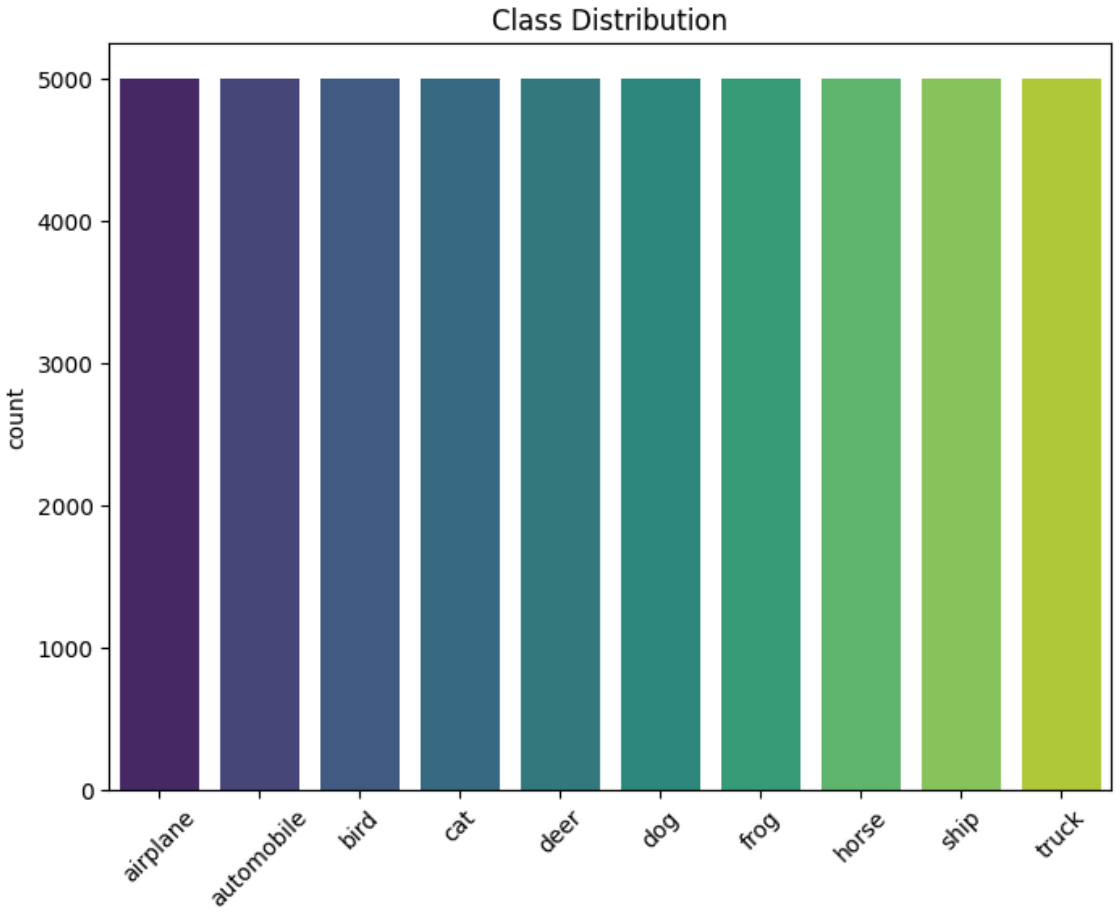
Class Distribution: The CIFAR-10 dataset is divided into 10 classes, each representing a distinct category of objects or animals. The classes in CIFAR-10 are as follows:

- Airplane
- Automobile
- Bird
- Cat
- Deer
- Frog
- Dog
- Horse
- Ship
- Truck

Number of Images per Class:

- Airplane: 6,000 images
- Automobile: 6,000 images
- Bird: 6,000 images
- Cat: 6,000 images
- Deer: 6,000 images
- Dog: 6,000 images
- Frog: 6,000 images
- Horse: 6,000 images
- Ship: 6,000 images
- Truck: 6,000 images

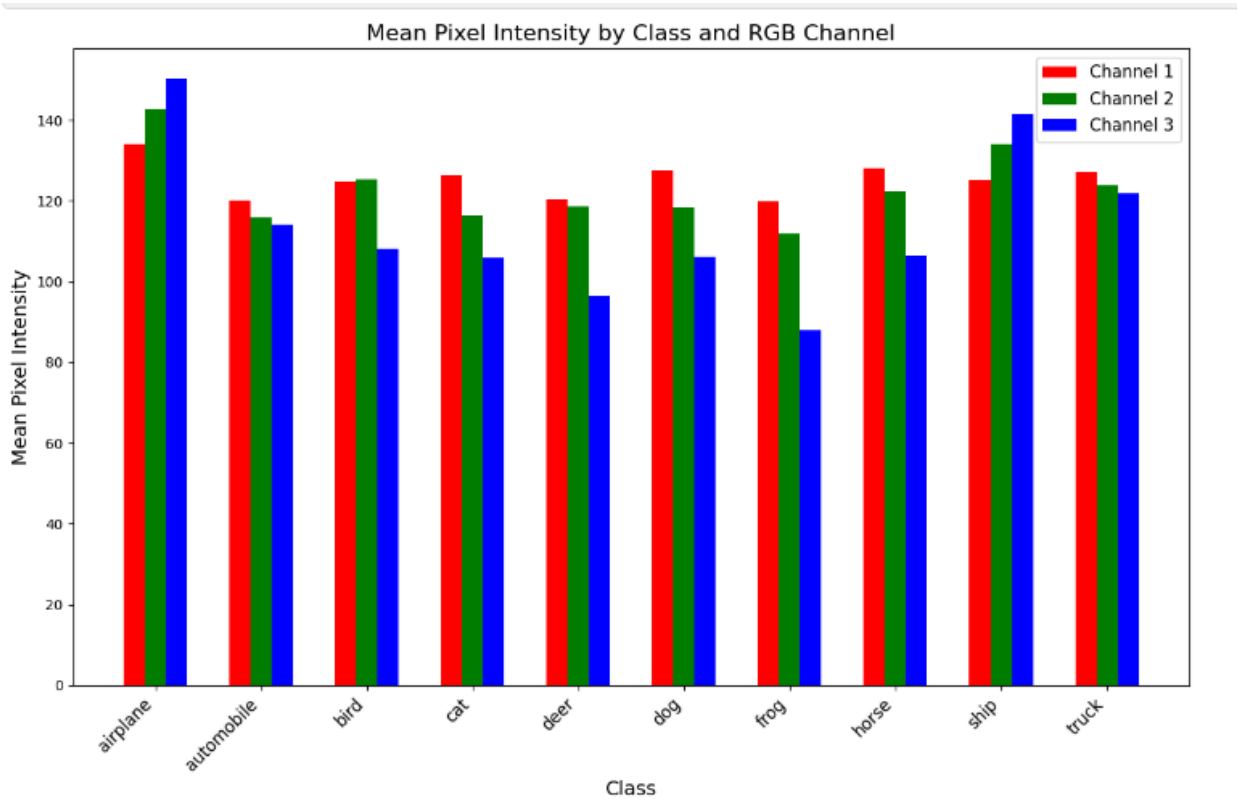
Visualizing Class Distribution:



Mean Intensity by Pixel and RGB Graph:

The mean intensity by pixel and RGB graph provides insights into the average color intensity across different pixels and color channels (Red, Green, Blue) in an RGB image.

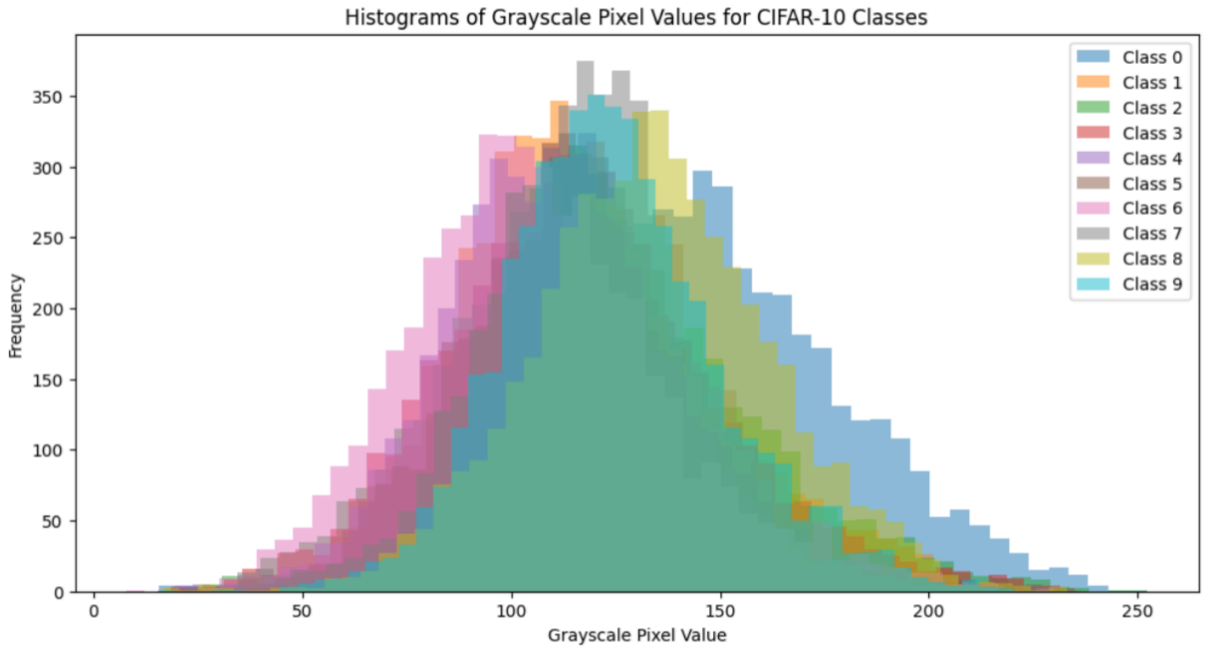
This graph helps in understanding the overall brightness and color composition of the image. Variations in the mean intensity across pixels or color channels can indicate areas of high contrast or specific color dominance within the image.



Histogram of Grayscale Pixel Values Graph:

The histogram of grayscale pixel values represents the frequency distribution of intensity levels in a grayscale image. Each pixel in a grayscale image has an intensity value ranging from 0 (black) to 255 (white), where 0 represents black and 255 represents white.

The resulting histogram typically shows a distribution of pixel intensities, indicating how common certain intensities are within the image. A peak at lower intensities (left side) suggests darker regions, while peaks at higher intensities (right side) suggest lighter regions.



Preprocessing

Before passing our data to the model, we preprocess it, using 2 methods, namely

- **Normalization** -

  Since the data which we deal with here is image data, the image pixel values lie between a range of 0-255, hence the initial step is to normalize these pixel values and bring them in a range of 0-1 by dividing the pixel values by 255.

  Also we convert our image numpy arrays to torch tensors as the framework in which the algorithms are written is pytorch.

- **Data Augmentation -**

  This technique involves applying a variety of transformations to existing data, such as rotating, flipping, and cropping images. Rotating an image introduces variations in orientation, while flipping can mirror images horizontally or vertically, providing different perspectives.

These transformations help the model generalize better by exposing it to a wider range of variations and reducing overfitting to specific patterns present in the original dataset.

Algorithm

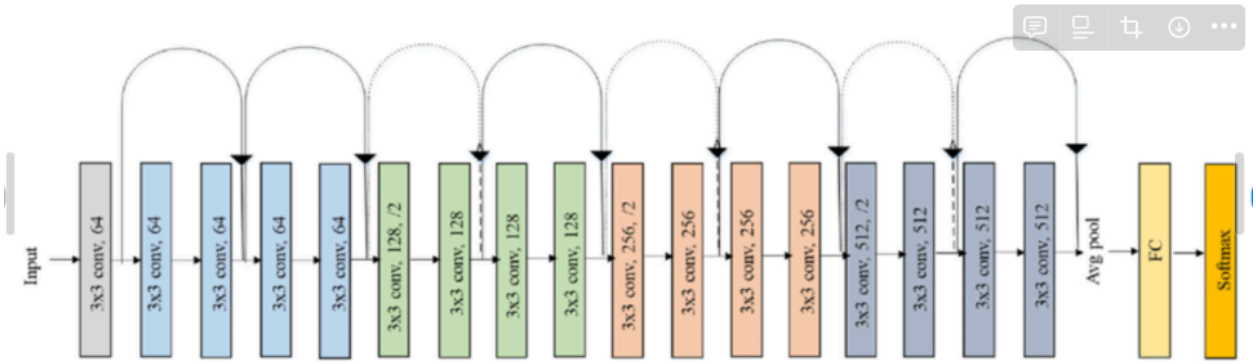
About Resnet18

Our base model architecture which we initially train on our dataset and then use an unlearning algorithm to make this trained model "unlearn" some specific images of the dataset is Resnet18.

ResNet-18 is designed with a total of 18 weight layers, including convolutional layers, pooling layers, fully connected layers, and an output layer. It consists of a series of residual blocks, which are composed of multiple convolutional layers, batch normalization, and shortcut connections. These shortcut connections allow the model to learn residual functions, making it easier to train very deep networks.

Key features of ResNet-18 include:

1. **Residual Blocks**: The main building blocks of ResNet architectures. Each residual block typically contains two convolutional layers with batch normalization and ReLU activation functions.
2. **Shortcut Connections**: These connections skip one or more layers and add the output of an earlier layer to the output of a later layer. They facilitate the flow of gradients during training, which helps to combat the vanishing gradient problem in very deep networks.
3. **Downsampling**: ResNet-18 uses max-pooling layers to reduce the spatial dimensions of feature maps in certain blocks, effectively downsampling the input.
4. **Global Average Pooling**: Instead of using fully connected layers at the end of the network, ResNet-18 employs global average pooling. This operation averages the spatial dimensions of the feature maps, resulting in a vector that represents the average activation of each feature map. This reduces the number of parameters and helps prevent overfitting.



Why resnet?

We have chosen Resnet18 as our base architecture for learning and unlearning because it is used as a standard benchmark for various training tasks.

This model has been trained using SGD with a learning rate of 0.1, momentum of 0.9 and weight decay of 5e-4.

After successfully training the model on the cifar-10 dataset we utilize our proposed unlearning algorithms to untrain these learned models.

Unlearning Algorithms

1. **Unlearning by fine tuning**

   The provided unlearning function implements a fine-tuning approach for machine unlearning. This method aims to update a pre-trained neural network (net) by training it using a specified subset of the original training data called the "retain set." The retain set contains examples that we want the model to remember or continue to perform well on, while ignoring another subset called the "forget set," which contains examples we aim to unlearn or remove the influence of. The function uses a simple training loop over a fixed number of epochs (in this case, 5 epochs) to update the model's weights based on the retain set's data. It employs a cross-entropy loss function for optimization using stochastic gradient descent (SGD) with momentum and weight decay. Additionally, a learning rate scheduler based on the cosine annealing method is utilized to adjust the learning rate during training.

Pros of this approach:

- Simplicity: Fine-tuning is straightforward to implement and understand, making it accessible for practical usage.
- Efficiency: By focusing only on the retain set, computational resources are efficiently utilized compared to retraining the entire model from scratch.
- Retains Useful Knowledge: The model retains knowledge from the retain set, ensuring continued performance on relevant tasks.

Cons of this approach:

- Limited Unlearning: This method does not directly address forgetting or removing the influence of specific examples from the forget set, potentially leading to incomplete unlearning.
- Potential Overfitting: Fine-tuning on a limited subset may lead to overfitting on that subset, reducing generalization performance on unseen data.
- Requires Domain Knowledge: Fine-tuning parameters such as learning rate, optimizer settings, and epochs require careful tuning based on domain-specific knowledge and experimentation.

2. **Unlearning as Adversarial Regularization**

The second approach to machine unlearning outlined in the provided function (unlearning) is a more complex method based on adversarial training and regularization principles. This approach aims to modify a pre-trained neural network (net) to remove or reduce the influence of specific subsets of training data, particularly the "forget set," which contains examples we wish to unlearn. The method leverages adversarial training between a discriminator (netD) and the primary model (net) to encourage the primary model to forget sensitive or unwanted features associated with the forget set.

Key components and steps of this approach include:

Adversarial Training: The discriminator (netD) is trained to distinguish between outputs (logits) of the primary model (net) derived from the forget set and those from the retain set. This adversarial setup aims to encourage the primary model to produce outputs that are less indicative of the forget set's characteristics.

Regularization via Min-Max Optimization: The primary model (net) is fine-tuned to maximize the discrimination loss computed by the discriminator (netD). This min-max optimization strategy pushes the primary model towards producing outputs that do not contain significant identifying information from the forget set.

Distillation Strategy: The primary model (net) is encouraged to match the output distribution of an original teacher model (oracle) on the retain set, aiming to retain useful knowledge while reducing sensitivity to unwanted features.

Pros of this approach:

- Effective Unlearning: Adversarial training and regularization techniques can effectively reduce the influence of specific subsets of training data, enhancing privacy and reducing bias.
- Retention of Useful Information: By distilling knowledge from an oracle model, the primary model can retain important features relevant to the retain set while discarding unwanted information.
- Flexible Optimization: The approach allows for tuning of hyperparameters to balance objectives such as discrimination loss, age classification accuracy, and output distribution matching.

Cons of this approach:

- Complexity and Hyperparameter Tuning: The method requires careful tuning of multiple hyperparameters (e.g., learning rates, strength parameters) to achieve desired outcomes, which can be challenging and time-consuming.

- Potential Performance Trade-offs: Optimizing for reduced influence from the forget set may impact the primary model's overall performance and generalization ability, particularly in tasks like age classification.

Performance Measures Used

1. **MIA (Membership Inference Attacks)**

   The MIA leverages the fact that examples that were trained on have smaller losses compared to examples that weren't. Using this fact, the simple MIA defined below will aim to infer whether the forget set was in fact part of the training set.

   This MIA takes as input the per-sample losses of the unlearned model on forget and test examples, and a membership label (0 or 1) indicating which of those two groups each sample comes from. It then returns the cross-validation accuracy of a linear model trained to distinguish between the two classes.

   Intuitively, an unlearning algorithm is successful with respect to this simple metric if the attacker isn't able to distinguish the forget set from the test set any better than it would for the ideal unlearning algorithm.

2. **Gold Standard**

   The MIA of the Retain Set is defined as the Gold Standard.

3. **Forgetting Quality**

The function below computes the forgetting quality F given only the (scalar) outputs of the retrain and unlearn models on all the forget set examples. It:

1. Iterates over each sample in S.

2. Performs the attacks specified to obtain lists of FPRs and FNRs,

3. Computes the privacy degree for each sample,

4. Computes the forgetting quality by averaging over the forget scores of all samples.

4. **Unlearning Score**

   Finally, we package everything together in a single function that takes an unlearning callable, unlearns many times and outputs a dictionary of various quantities, including:

   * The forget quality - F

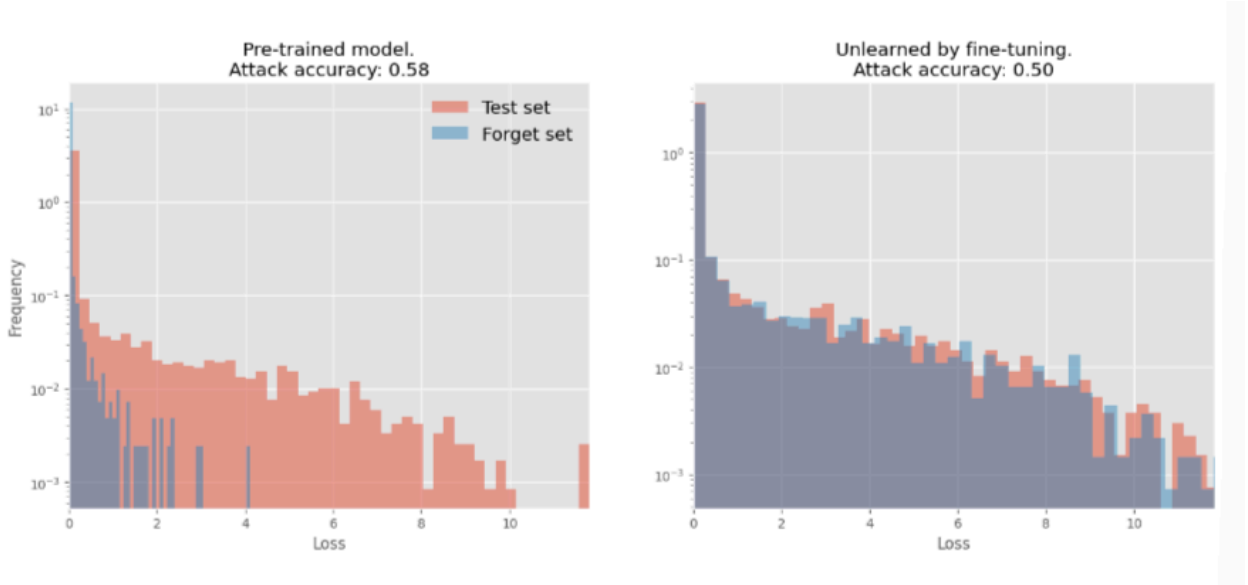
   * The various accuracies RA, TA

   * The total score = f * RA * TA

Experimentation and Results

1. **Unlearning by Fine Tuning**

SNo.	Epochs	LR	F	US	GS	MIA O	MIA U
1	5	0.1	0.1113	0.1046	0.507	0.580	0.515
2	5	0.01	0.1342	0.1351	0.502	0.579	0.559
3	5	0.001	0.0380	0.0384	0.500	0.579	0.579
4	15	0.1	0.1637	0.1551	0.502	0.579	0.503
5	**25**	**0.1**	**0.1715**	**0.1616**	**0.502**	**0.577**	**0.504**



2. Unlearning by Adversarial Training

SNo.	HS	LR	F	US	GS	MIA O	MIA U
1	16	0.0004	0.1151	0.1114	0.499	0.577	0.569
2	16	0.004	0.0854	0.0166	0.502	0.579	0.520
3	**128**	**0.0004**	**0.1271**	**0.1230**	**0.504**	**0.577**	**0.572**
4	8	0.0004	0.1153	0.1117	0.497	0.577	0.573
5	16	0.00004	0.0336	0.0330	0.501	0.577	0.575

LR = Learning Rate

F = Forgetting Quality

HS = Hidden States

US = Unlearning score

GS = Gold Standard

MIA O = MIA of Original Model

MIA U = MIA of Unlearned Model

Conclusion

We recognize and understand a newer area of research "Machine Unlearning". Our analysis is performed on the Cifar-10 dataset on which we perform data analysis and preprocessing. We use Resnet as our base architecture and perform machine unlearning using two proposed algorithms.One is based on finetuning and the other is based on adversarial training. By performing extensive evaluation and utilizing different kinds of metrics we conclude that the fine tuning algorithm performs better than the latter algorithm with an Unlearning score of **0.1616**.