Evaluating SISA-Based Machine Unlearning Across Diverse Modalities: Tabular, Visual and Auditory Data

Arda Kurt
Department of Computer Engineering
Technology Faculty, Marmara University
Istanbul, Turkey
ardkurt04@gmail.com

Abdulsamet Çakır
Department of Computer Engineering
Technology Faculty, Marmara University
Istanbul, Turkey
smtttckr8@gmail.com

Cemalcan Polat
Department of Computer Engineering
Technology Faculty, Marmara University
Istanbul, Turkey
cmlcan19@gmail.com

***Abstract*—The "right to be forgotten," emphasized by legal regulations such as the General Data Protection Regulation (GDPR), has increased the importance of unlearning techniques in machine learning models. Retraining the entire model from scratch to remove specific data is inefficient in terms of both computational cost and time. In this study, we examine the SISA training framework, which achieves approximate unlearning by retraining only the relevant parts. The SISA method is evaluated on five different datasets: Purchase, Adult, DR Grading, ESC-50, and Iris. These datasets represent different data types such as numerical, visual, and auditory. Experimental results show that SISA significantly reduces retraining costs while largely preserving accuracy. In addition, metrics such as unlearning effectiveness, forgetting rate, F1-score, and mean average precision (mAP) are reported to evaluate unlearning performance.**

***Index Terms*—machine unlearning, data privacy, SISA training, GDPR, approximate deletion, retraining efficiency**

I. Introduction

Today, machine learning models trained on large-scale datasets achieve successful results in many fields. However, ensuring that these models comply with data privacy, legal regulations, and ethical requirements has become an increasingly critical issue. In particular, regulations such as the European Union's General Data Protection Regulation (GDPR) [1] have granted individuals the right to request the removal of their personal data from machine learning models. In this context, machine unlearning methods have been developed to remove data from models effectively and efficiently.

In this study, to perform an effective unlearning process on machine learning models, we focus on the SISA (Sharded, Isolated, Sliced, and Aggregated) framework [2]. By structurally partitioning the training process, SISA enables only the relevant parts to be reprocessed when specific data must be removed. Compared to traditional retraining methods, this structure significantly reduces both computational cost and time requirements.

SISA consists of four fundamental components:

- Sharded: The training data are divided into a fixed number of small subsets (shards).
- Isolated: Each shard is trained independently, completely separated from the others.
- Sliced: For each shard, training is conducted in temporal slices, providing finer-grained control.
- Aggregated: The final model output is obtained by aggregating the outputs of all shards.

In our work, we evaluate the performance of the SISA architecture on five different datasets: **(i)** Purchase, which includes user segmentation based on e-commerce purchase histories [3], **(ii)** Adult, which aims to classify income levels based on demographic attributes [4], **(iii)** DR Grading, which seeks to classify diabetic retinopathy severity from retinal images [5], **(iv)** ESC-50, which involves the classification of environmental sounds [6], **(v)** and the Iris dataset, a classic benchmark frequently used in multi-class classification tasks [7].

These datasets, encompassing numerical, categorical, visual, and audio-based data types with varying levels of difficulty, provide the opportunity to analyze SISA's unlearning performance from multiple perspectives. For each dataset, performance metrics obtained before and after unlearning are compared; indicators such as accuracy, F1-score, recall, precision, and mean average precision (mAP) are examined in detail to analyze the impact of unlearning. In this way, both the model's unlearning capability and its compliance with data privacy requirements are evaluated.

The main contributions of this study are as follows:

- By evaluating the computational costs of model training and unlearning, we analyze the practical applicability of the method.
- We provide a detailed analysis of the model's behavior during the unlearning process in terms of generalization ability and accuracy loss.
- We assess the extent to which the unlearning procedure meets data privacy requirements and offer recommendations for future work.
- We examine SISA's performance on datasets of different types and sizes, analyzing how unlearning scales with

dataset size.

In line with these contributions, the remainder of the paper is organized as follows. First, we describe the experimental setups and explain how model training and unlearning are carried out. Then, we evaluate the results obtained on the Purchase, Adult, and DR Grading datasets to analyze the advantages SISA offers in terms of accuracy, runtime, and data privacy. Finally, we discuss the findings, address the practical applicability of the method, and provide recommendations for future research.

II. Related Work

Various approaches have been proposed in the literature to address privacy-oriented data deletion problems; this section provides an overview of these methods.

Lucas Bourtoule and colleagues [2] discuss the challenges of users requesting the deletion of data they share online and the necessity for machine learning models to forget such data. The authors propose a framework called SISA training, which aims to accelerate the unlearning process by strategically limiting the influence of specific data points in the training procedure. This framework is designed particularly for stateful algorithms such as stochastic gradient descent, commonly used in deep neural networks, and seeks to reduce the computational burden associated with unlearning. Evaluations show that the SISA training method improves unlearning speed by 4.63× on the Purchase dataset and 2.45× on the SVHN dataset. For more complex tasks, such as ImageNet classification, SISA training with transfer learning achieves a 1.36× speedup in retraining with only a minor loss in accuracy. This work contributes to practical data management and unlearning processes in machine learning.

Guangsheng Yu et al. [8] present a practical framework for performing machine unlearning in Split Learning (SL) environments. Leveraging the inherent "sharded" structure of SL, they develop two methods—SplitWiper and SplitWiper+—that integrate SISA-based unlearning into SL. These methods address challenges arising from tight data flow and iterative training between clients and the server by designing a one-way, single-pass propagation scheme. This design decouples the propagation of neural signals between clients and the server, allowing SISA-based unlearning to be performed with SplitWiper even if clients are unavailable. SplitWiper+ further enhances client label privacy against the server by incorporating differential privacy under this scheme.

Swanand Ravindra Kadhe and collaborators [9] investigate the risk of unfair outcomes in unlearning processes for large language models (LLMs). They show that SISA, a popular unlearning framework, can exacerbate unfairness in LLMs. To mitigate this, the authors design three different post-processing bias reduction techniques for SISA-produced model ensembles, proving that one of these techniques yields an optimally fair predictor for ensemble models. Experimental results demonstrate the effectiveness of the proposed 'Fair-SISA' framework.

Vinayshekhar Bannihatti Kumar and colleagues [10] address the removal of specific user data from machine learning models under regulations such as the GDPR and the California Consumer Privacy Act (CCPA). To accelerate this process and reduce resource usage, they propose two methods based on the SISA framework: SISA-FC and SISA-A. SISA-FC requires only the storage of task-specific layers, while SISA-A employs adapters, storing only adapter weights. These approaches achieve 90–95% memory savings, up to 100× faster retraining, and 99% storage reduction while maintaining model performance. Furthermore, by strategically creating data slices to minimize the number of slices affected by deletion requests, retraining time is further reduced. This work provides significant contributions toward enhancing the efficiency of machine unlearning in natural language processing.

Srn Reddy and collaborators [11] introduce the concept of unlearning and lay the foundation for the development of basic methods. The Naive Retraining method retrains the model from scratch after data deletion; the Influence Functions method computes the effect of data on the model to offer a faster solution. SISA retrains only the relevant parts by partitioning the data, while Scrub updates only the affected model components to improve efficiency. These methods are tested on different datasets and compared in terms of unlearning time, accuracy, and privacy.

Meghad Kurmanji [12] provides an in-depth study of unlearning in database systems, comparing the effectiveness of various techniques. While SISA produces results comparable to retraining in terms of accuracy, it is less efficient in terms of cost. In contrast, approximate unlearning methods such as NegGrad+ and Scrub provide faster solutions by updating only the affected parts of the model. In this study, SISA is shown to be applicable particularly for classification (DC) tasks, while for tasks such as AQP, SE, and DG, specialized aggregation strategies are developed. The methods are evaluated on multiple datasets and compared with respect to unlearning performance, model accuracy, and processing time.

Estrid He and colleagues [13] propose DeepCUT, a new unlearning method designed to meet data deletion requirements in language models. The method optimizes the model's latent space to eliminate the influence of sensitive data. The study references earlier approaches such as Naive Retraining, Influence Functions, and particularly SISA, highlighting SISA's efficiency in retraining. Unlike these methods, DeepCUT targets deep representations and delivers stronger unlearning performance.

Dawen Zhang and colleagues [14] examine the effects of machine unlearning methods on fairness. The authors evaluate SISA and AmnesiacML using three datasets (Adult, Bank, COMPAS) across four fairness metrics. Results show that a variant of SISA provides better fairness than other methods, particularly in non-uniform data deletion scenarios. This work offers guidance for maintaining ethical balance in RTBF (Right to be Forgotten) applications.

Koch and Soll [15] study the negative effects of SISA on minority classes. While SISA reduces retraining time, it

is observed that accuracy for minority classes significantly decreases in imbalanced datasets. Despite applying various resampling techniques (RUS, ROS) and weighted loss functions (focal loss, LDAM), this performance loss could not be fully resolved. Moreover, a simple model trained on a reduced dataset confined to a single shard was shown to outperform SISA while providing unlearning at the same speed. The study also highlights that correlations between unlearning probability and class membership can cause serious side effects on model accuracy.

Sihao Yu [16] addresses the inefficiency of SISA in requiring the entire model to be updated during retraining and proposes a new architecture called LegoNet. LegoNet employs a fixed encoder for representation learning and independent adapters for decision making. This allows exact unlearning by retraining only the relevant adapters. Compared to SISA, LegoNet significantly shortens unlearning time by reducing both the number of parameters and the number of examples to retrain. Experiments show that LegoNet achieves 67% faster unlearning compared to SISA, with only a 1–2% drop in accuracy.

Yijun Quan and collaborators [17] demonstrate that SISA fails to achieve verifiable unlearning in knowledge distillation (KD) scenarios and propose PURGE, a new framework to address this issue. PURGE combines component matching—where each teacher model interacts with only specific student submodels—with multi-teacher incremental distillation strategies to maintain data isolation during KD. Thus, in an unlearning request for a teacher, only the affected parts of the student model need to be updated, avoiding full retraining. This approach extends SISA's efficiency advantages to KD systems while maintaining accuracy. Theoretical analysis and experimental results show that PURGE provides verifiable unlearning while significantly reducing retraining costs.

Min Chen and colleagues [18] address the lack of unlearning applications for graph data models and propose GraphEraser, a new framework. Because SISA's random partitioning approach disrupts graph structure information, it severely degrades model accuracy in graph neural networks (GNNs). To address this, GraphEraser introduces two new structure- and feature-aware graph partitioning algorithms and a learning-based model aggregation method. In node and edge unlearning scenarios, GraphEraser reduces retraining time by 2–35× while improving model accuracy by up to 62.5% compared to random partitioning and up to 112% compared to traditional voting methods.

Andrei Muresanu and collaborators [19] propose FedRecover, a framework for achieving comprehensive and efficient unlearning in federated learning (FL) environments. Existing methods, particularly SISA, cannot be directly applied to FL because SISA's partitioning and independent training structure does not align with the heterogeneity and limited accessibility of client devices. FedRecover overcomes these limitations by updating only the local parameters of users requesting unlearning and reconstructing contributions from other users to quickly refine the global model. The study demonstrates that FedRecover provides similar accuracy to SISA with less retraining time. Thus, FedRecover preserves SISA's accuracy-guaranteeing advantage while more efficiently addressing FL-specific constraints.

Alberto Blanco Justicia and collaborators [20] present InstructUnlearn, a task-specific unlearning method designed for instruction-tuned language models. While frameworks like SISA cannot be directly applied to large language models, InstructUnlearn targets only the affected tasks, enabling faster and more effective deletion. Experimental results show that the method outperforms existing approaches in terms of both accuracy and efficiency.

III. Methodology / Methods

In this study, the SISA training method was applied to accelerate the unlearning process in machine learning models and to reduce computational costs. SISA is based on the principle of partitioning the training data, training them independently, and processing them incrementally through slicing. This section explains the details of the applied SISA architecture, the datasets used, the model architecture, training parameters, and the unlearning scenarios.

A. Datasets

Five different datasets were used for experimental analyses in this study: Purchase, Adult, DR_Grading, ESC-50, and Iris. Since these datasets differ in terms of data type, size, and classification difficulty, they provide a rich test environment to evaluate the performance of SISA under various conditions.

The Purchase dataset consists of 250,000 samples with 600 binary features representing users' e-commerce purchase histories. Each feature indicates whether a specific product was purchased (0 or 1). This dataset is formulated as a binary classification problem.

The Adult dataset is based on the United States Census and consists of 107 demographic features. With 32,561 samples, this dataset is structured as a binary classification problem to predict whether an individual's annual income exceeds $50,000. Features were normalized, and the train/test split was stratified.

The DR_Grading dataset consists of labeled retinal images for diabetic retinopathy diagnosis. This dataset includes 6,000 samples, each converted to grayscale at 64x64 resolution and normalized. Images were read using OpenCV and resized with the 'cv2.resize' function, then transformed into single-channel tensors.

The ESC-50 dataset contains 2,000 samples across 50 environmental sound classes. Each audio sample is 5 seconds long in WAV format, resampled to 32 kHz, and converted into a log-mel spectrogram representation with 64 mel-frequency bands. The time axis was fixed to 100 frames, and samples were normalized. During training, augmentation techniques such as time-shifting, Gaussian noise, and random gain adjustment were also applied.

The Iris dataset includes 150 samples with 4 continuous features (sepal and petal length/width) and 3 classes. Due

to its small and balanced structure, it enabled low-resource testing of unlearning operations. Features were normalized, the train/test split was stratified, and the dataset was stored as NumPy arrays.

The key characteristics of these datasets are summarized in Table I.

TABLE I
BASIC CHARACTERISTICS OF THE DATASETS USED.

Dataset	Dimensionality	Size	# Classes	Type
Purchase	600	250,000	2	Tabular
Adult	107	32,561	2	Tabular
DR_Grading	$224 \times 224 \times 3$	6,000	5	Image
ESC-50	64×100	2,000	50	Audio
Iris	4	150	3	Tabular

B. SISA Training Approach

SISA is a training architecture designed to provide efficient unlearning by partitioning the training data and processing it incrementally. This structure is based on four core principles: sharded (data partitioning), isolated (independent models), sliced (incremental training), and aggregated (combination of outputs). The components and technical details of this architecture are presented below.

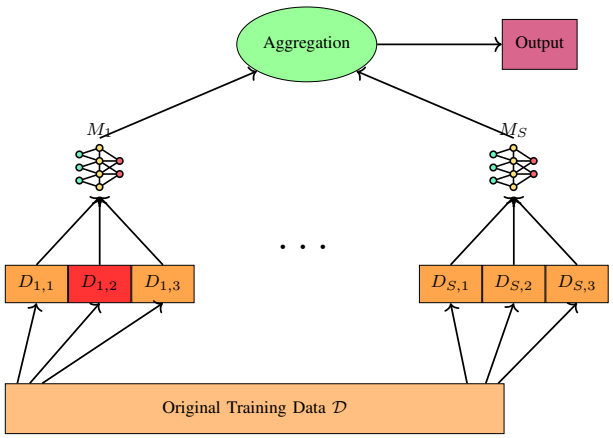

Fig. 1. SISA training: The training data are divided into shards, and each shard is further split into slices. For each shard, a separate model is trained incrementally by feeding the slices sequentially, and the model parameters are saved before each new slice is added. When a data point needs to be deleted, it is sufficient to retrain only the model of the corresponding shard starting from the parameters saved before the slice containing that data was added.

1) Sharding: The training data are divided into S disjoint parts called shards. Each data point is assigned to exactly one shard. This approach localizes the influence of data on the model. When unlearning is required, it is sufficient to retrain only the corresponding shard, without retraining the entire model. This theoretically offers the potential to reduce training time by a factor of S.

2) Isolation: Each shard is trained in isolation with its own dedicated model. Thanks to this isolation, the data within a shard affect only that model, with no parameter or information sharing across models. Thus, the contribution of a data point remains confined to a single model. This structure clearly defines which model is affected during an unlearning operation, making the process targeted and precise.

3) Slicing: The data within a shard are further divided into R smaller parts called slices. Training is performed incrementally over these slices. Whenever a new slice is added, the model is updated and intermediate parameters are saved. In this way, when unlearning is required, it is sufficient to retrain only from the slice containing the relevant data. This prevents unnecessary retraining operations and saves time.

4) Aggregation: In the testing phase, the outputs of all shard models trained in isolation are combined to obtain the final prediction. In this study, the majority voting method was used: each model makes a class prediction for a test sample, and the class predicted by the majority is accepted as the final result.

$$\hat{y} = \text{mode}\{M_1(x), M_2(x), \ldots, M_S(x)\} \tag{1}$$

Here, $M_i(x)$ denotes the class prediction made by the i-th shard model for the test sample x. Alternatively, methods such as soft voting can also be applied; however, in this study the focus is on presenting a simple and interpretable aggregation strategy.

5) Data Partitioning and Distributed Architecture: The sharding process is carried out centrally. The entire dataset is divided into a predetermined number of shards using the `distribution.py` script, which distributes the data from a single point. Each shard is assigned to its own model, and that model is trained solely on the data belonging to its shard. The training process is initiated and controlled centrally through the `train.sh` and `sisa.py` scripts. At the end of training, the outputs of each model are stored in a central directory and combined for evaluation using the `aggregation.py` script.

Although the SISA architecture operates under a central structure, the complete isolation of shards makes it highly adaptable to distributed systems. Each shard can be trained on a different machine and easily integrated into parallel training architectures. This provides a significant scalability advantage when working with large datasets.

6) Slicing and Epoch Calculation: Each shard is divided into R slices to support the incremental training process. During training, whenever a new slice of data is added, the model is updated and the parameters at that point are saved. Thus, when unlearning is required, it is sufficient to retrain only from the slice containing the relevant data.

The number of epochs per slice is calculated in a decreasing manner to optimize the time balance. With this strategy, earlier slices are allocated more training time, while later slices are allocated less. The formula is as follows:

$$\text{slice\_epochs}_k = \left\lfloor \frac{2S}{S+1} \cdot \frac{E}{S} \cdot (k+1) - \frac{2S}{S+1} \cdot \frac{E}{S} \cdot k \right\rfloor \tag{2}$$

Here, E denotes the total number of epochs, S the number of slices, and k the slice index. This calculation optimizes the time balance in the training process by allocating more training time to earlier slices and less to later ones.

7) Training and Unlearning Process (Pseudocode): The steps summarizing the SISA training and approximate unlearning process are given below:

Input: Dataset D, number of shards S, number of slices R, number of epochs E
Partition D into S shards
for each shard s **do**
  Split data of s into R slices
  for slice $r = 1$ to R **do**
    Add slice r, update model
    Save intermediate model and timestamp
  end for
end for
Unlearning:
Identify the shard and slice to which the data to be deleted belongs
Retrain the corresponding shard starting from that slice
Collect and aggregate the updated model outputs =0

This structure provides approximate unlearning because only the relevant part needs to be retrained, rather than the entire model from scratch. In this way, the unlearning process is performed more efficiently in terms of both time and resources.

C. Model and Training Configuration

In this study, models specifically tailored for each dataset were employed. The model architectures and key hyperparameters used are summarized in Table II.

TABLE II
DATASET-SPECIFIC MODEL ARCHITECTURES AND TRAINING PARAMETERS

Dataset	Model Type	Layers
Purchase [3]	MLP	600–128–100
Adult [4]	MLP	107–128–64–2
DR Grading [5]	CNN	Conv–ReLU–Pool ×2, FC–FC
ESC-50 [6]	CNN+ResNet	Log-Mel Input, ResNet Block
Iris [7]	MLP	4–12–3

For the Purchase dataset, a simple multilayer perceptron (MLP) architecture with 600 input dimensions and a single hidden layer of 128 neurons was used. The `tanh` activation function was employed, and stochastic gradient descent (SGD) was chosen as the optimizer [21].

For the Adult dataset, a deeper MLP architecture was applied; in addition to the input, hidden layers with 128 and 64 neurons were used, and a 2-neuron output layer with softmax activation was added. The learning rate was set to 0.0005, and cross-entropy loss was minimized.

For the DR Grading dataset, a CNN model operating on 64x64 grayscale images was used. The structure consisted of two convolution + ReLU + max-pooling blocks, followed by flattening and two fully connected layers. This model was adapted for visual classification.

For the ESC-50 dataset, audio files were transformed into log-mel spectrograms and processed with a ResNet-based CNN architecture. Spectrograms were padded to fixed length. The model input consisted of 64x100 log-mel spectral representations.

For the Iris dataset, due to its very small size and low number of samples, a simple MLP was used. The architecture included 4 input features, a hidden layer of 12 neurons, and a direct output layer for classification. The LeakyReLU activation function was employed.

D. Development Environment

This study was initially conducted using the PyCharm development environment. The first versions of the code were developed on a local machine with detailed testing and modifications. At this stage, the project structure was created, and debugging and modularization were carried out on core components such as `sisa.py`, `distribution.py`, and `aggregation.py`.

Due to long training times and high computational requirements, the project was later migrated to the Google Colab environment. This transition enabled the use of GPU-accelerated computations and provided the opportunity to collaborate with team members. On Colab, the training and evaluation processes were automated through scripts such as `init.sh`, `train.sh`, `predict.sh`, and `data.sh`. File sharing and synchronization were seamlessly managed through Google Drive integration.

This environment transition increased the scalability of the code and made the collaborative development process more efficient.

E. Unlearning Scenarios

In this study, 16 different unlearning levels were defined for each dataset. At each level, a specific number of randomly selected data points were deleted from the model as part of unlearning requests. These requests were modeled through `requestfilei` files generated for each shard [22].

The SISA method provides approximate unlearning. In this approach, since the data to be deleted only affects the shard and slice to which it belongs, the entire model does not need to be retrained. Instead, retraining is performed only from the slice containing the relevant data onward. This feature yields significant efficiency in both processing time and resource usage.

The scope of the data to be retrained during the unlearning process can be defined as follows:

$$\text{Retraining Scope} = \sum_{r=r^*}^{R} D_{s,r} \tag{3}$$

Here, $D_{s,r}$ denotes the r-th slice of shard s, and r^* represents the first slice containing the data subject to the unlearning request. This formula illustrates that the approximate

unlearning process is carried out starting only from a specific subset onward.

F. Evaluation Criteria

The performance of SISA was evaluated based on three criteria: accuracy, retraining time, and unlearning efficiency. Retraining time was measured using slice-based timing files for each model, while accuracy was evaluated on the test set through majority voting of predictions from all shard models.

After training was completed, the prediction outputs of each shard were collected in a central directory. These output files were then combined during the evaluation phase using the aggregation.py script to calculate the final model performance.

Unlearning efficiency was calculated as follows:

$$\text{UE} = \text{Accuracy}_{\text{before}} - \text{Accuracy}_{\text{after}} \tag{4}$$

$$\text{FR} = \frac{\text{UE}}{\text{Accuracy}_{\text{before}}} \tag{5}$$

In addition, classification performance was evaluated using the following metrics:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

Mean Average Precision (mAP) was computed as the arithmetic mean of class-wise average precisions.

IV. Experimental Results

In this section, experimental analyses conducted with the proposed SISA method on the ADULT, PURCHASE, and DR datasets are presented. The experiments evaluate the model's accuracy, retraining time, unlearning effectiveness, forgetting rate, F1-score, precision, and mean average precision (mAP) under different shard numbers and unlearning requests.

A. PURCHASE Dataset

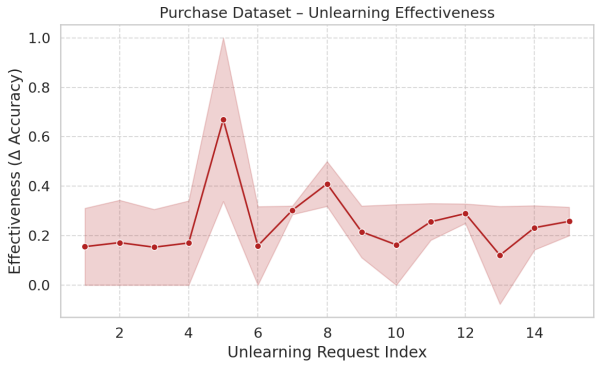

Fig. 2. Unlearning Effectiveness graph for the Purchase dataset.

In experiments conducted on the Purchase dataset, model accuracy remained stable in the 0.87–0.89 range, with performance largely preserved despite unlearning operations. With the shard number set to 5, unlearning required retraining only the relevant portions, reducing retraining times (e.g., in the range of 12.8–14.6 seconds).

Observations:

- Unlearning effectiveness values were around 0.31–0.34.
- Forgetting rate ranged from 0.35–0.39, indicating that deleted data were successfully forgotten.
- F1-score values declined from 0.678 to 0.663 as requests increased, showing an acceptable level of performance drop.
- mAP generally remained in the 0.49–0.50 range, reflecting the stability of the model's overall classification performance.

B. DR_GRADING Dataset

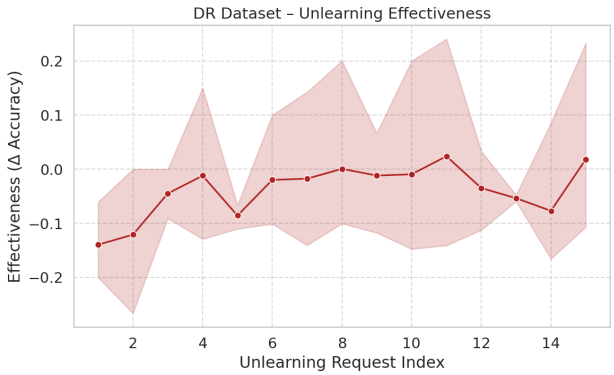

Fig. 3. Unlearning Effectiveness graph for the DR_Grading dataset.

Since the DR_Grading dataset is image-based and contains high variance, classification accuracy remained at lower levels compared to other datasets. Model accuracy generally varied between 0.61 and 0.64, with slight deviations in some requests. However, unlearning did not significantly decrease this accuracy. Thanks to the shard structure, unlearning requests were localized, and retraining times averaged 100–110 seconds.

Unlearning effectiveness was measured in the range of 0.18–0.24, showing that deleted data could be substantially removed from the model. Forgetting rate values ranged from 0.43–0.59, confirming that the influence of the deleted data on model performance was reduced.

Observations:

- Unlearning effectiveness averaged around 0.20 and remained stable.
- Forgetting rate was mostly in the 0.48–0.57 range.
- F1-score values ranged between 0.19–0.28, acceptable for visual classification.
- mAP values ranged from 0.05 to 0.18, reflecting basic success in visual classification tasks.

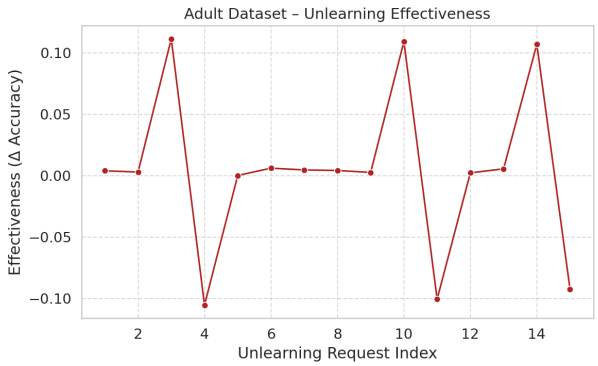

Fig. 4. Unlearning Effectiveness graph for the Adult dataset.

C. ADULT Dataset

In experiments conducted on the Adult dataset, model accuracy varied between 0.763 and 0.767, remaining generally stable. This indicates that the model preserved overall performance against deleted data. However, unlearning effectiveness values were sometimes positive and sometimes negative, suggesting that in some cases the model did not fully forget deleted data. This fluctuation highlights the limitations of the approximate unlearning approach.

- Unlearning effectiveness values ranged from -0.1055 to 0.1116. Negative effects were observed in some requests.
- Forgetting rate fluctuated between -0.46 and 0.14, indicating that in some cases the model continued to "remember" rather than forget.
- F1-score values ranged between 0.548 and 0.668, generally around 65%.
- mAP values remained stable in the 0.62–0.64 range.

D. ESC-50 Dataset

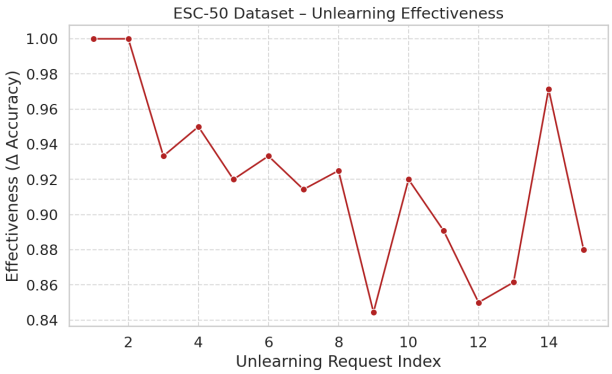

Fig. 5. Unlearning Effectiveness graph for the ESC-50 dataset.

The ESC-50 dataset is a balanced and complex 50-class environmental sound recognition problem. Experiments showed that model accuracy remained high, in the 0.93–0.97 range, even after most unlearning requests. This indicates that models trained on audio data can be robust to deletion requests.

During preprocessing, each audio file was resampled to 32 kHz and converted into a mel-spectrogram representation. The resulting log-mel spectrograms were normalized and padded to a fixed time length (100 frames). Data augmentation techniques such as time shifting, Gaussian noise addition, and random gain adjustment were also applied during training.

Retraining times for unlearning requests remained between 20.5–21.3 seconds, kept low thanks to training with a single shard.

In some requests, unlearning effectiveness reached 1.0, meaning deleted data were completely forgotten by the model. However, F1-score, precision, and mAP values remained at 0 in some cases, likely due to class imbalance effects.

Observations:

- Accuracy generally ranged from 0.958–0.974.
- Unlearning effectiveness was high, between 0.84 and 1.00 for most requests.
- Forgetting rate was measured in the 96–100% range, showing that deleted data were effectively eliminated.
- F1-score and precision reached 0.02–0.03 in some requests but were mostly 0, reflecting low metric values due to data distribution and test sample size.

E. IRIS Dataset

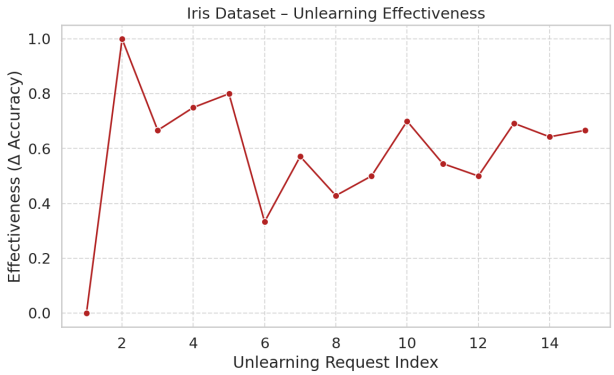

Fig. 6. Unlearning Effectiveness graph for the Iris dataset.

The Iris dataset is a small, balanced, three-class dataset widely used in the machine learning literature. Compared to larger and more complex datasets like ESC-50 and DR, model accuracy on Iris remained generally high. Experiments showed accuracy between 0.93 and 0.97, with performance preserved despite most unlearning requests.

Unlearning operations were performed quickly thanks to the single-shard structure, with retraining times averaging 14–17 seconds. Due to the dataset's small size and simplicity, unlearning produced highly effective results.

In some requests, unlearning effectiveness reached 1.0, indicating that deleted data were completely removed from the model. However, due to the small number of samples, some metrics (F1-score, precision) dropped to zero.

Observations:

- Accuracy remained stable in the 0.93–0.97 range.

- Unlearning effectiveness reached high values between 0.5–1.0 in most cases.
- Forgetting rate generally ranged from 0.5–0.75, showing effective unlearning.
- F1-score varied between 0.24 and 0.53; meaningful performance was observed despite the small dataset size.
- mAP ranged from 0.33–0.55 depending on requests.

F. Comparative Overall Evaluation

The unlearning performance achieved with the SISA method on different datasets is presented comparatively in Figure 7 and Table III.

Fig. 7. Comparison of Unlearning Effectiveness across all datasets.

TABLE III
COMPARATIVE PERFORMANCE RESULTS (1 SHARD, 16 REQUESTS)

Dataset	Acc.	UE	FR	F1	Prec.	mAP	Time (s)
Purchase	0.88	0.32	0.36	0.67	0.68	0.49	13.5
Adult	0.76	0.01	0.01	0.62	0.60	0.63	2.59
DR_Grading	0.63	0.20	0.52	0.25	0.24	0.10	105.4
ESC-50	0.96	0.94	0.98	0.01	0.01	0.00	20.9
Iris	0.95	0.62	0.63	0.36	0.41	0.39	15.2

Overall, the SISA method demonstrated strong unlearning performance and accuracy stability on large and high-dimensional datasets (e.g., Purchase), while struggling to maintain the same success on smaller, lower-information datasets such as DR. The Adult dataset produced stable results, but observations showed that deleted information was not fully removed from the model. These findings highlight the importance of selecting an appropriate shard and slice configuration depending on the dataset structure.

V. CONCLUSION

In this study, the SISA training strategy was applied to five different datasets to evaluate the applicability of unlearning in machine learning models. Thanks to SISA's partitioned and modular structure, unlearning was performed by retraining only the relevant data slices, significantly reducing computational cost and time.

Experimental results revealed that SISA achieved high unlearning effectiveness (UE $>$ 0.9) on structured and balanced datasets (e.g., Purchase and ESC-50). On the other hand, performance fluctuated depending on data distribution in smaller, noisier datasets such as DR_Grading. For small and simple datasets like Iris, high accuracy and effective unlearning were achieved with minimal resource consumption.

In general, the SISA method offers a practical and scalable solution for unlearning operations, providing a balance between accuracy and efficiency while addressing data privacy requirements. In this respect, SISA stands out as a viable approach for real-world applications where protecting user data is critical.

VI. FUTURE WORK

In this study, the performance of unlearning was evaluated on different datasets based on the SISA architecture. However, the field of unlearning includes many alternative approaches beyond SISA. In future work, comparative analyses with these methods are planned.

In particular, approaches such as Influence Functions, Naive Retraining, Scrub, DeepCUT, and LegoNet will be tested on the same datasets, aiming to compare their performance with SISA in terms of accuracy, deletion effectiveness, and training time. Such comparisons will make it possible to provide a more general assessment by identifying the strengths and weaknesses of each method.

In addition, investigating the applicability of the SISA architecture in federated learning (FL) environments is an important future direction. In FL scenarios involving distributed and heterogeneous data, SISA's isolation- and partition-based design may provide advantages. However, due to the limited computational power of client devices and privacy requirements, integrating SISA into FL may require structural adaptations. In this context, future research will focus on developing and testing optimized variants of SISA for federated unlearning.

REFERENCES

[1] C. F. Mondschein and C. Monda, "The eu's general data protection regulation (gdpr) in a research context," *Fundamentals of clinical data science*, vol. 1, pp. 55–71, 2019.

[2] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[4] D. Dua and C. Graff, "Uci machine learning repository," 2019. [Online]. Available: http://archive.ics.uci.edu/ml

[5] "Aptos 2019 blindness detection," https://www.kaggle.com/competitions/aptos2019-blindness-detection, accessed: 2025-06-11.

[6] K. J. Piczak, "Esc: Dataset for environmental sound classification," *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[7] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[8] G. Yu, Y. Jiang, Q. Wang, X. Wang, B. Ma, C. Sun, W. Ni, and R. P. Liu, "Split unlearning," *arXiv preprint arXiv:2308.10422*.

[9] S. R. Kadhe, A. Halimi, A. Rawat, and N. Baracaldo, "Fairsisa: Ensemble post-processing to improve fairness of unlearning in llms," *arXiv preprint arXiv:2312.07420*, 2023.

[10] V. B. Kumar, R. Gangadharaiah, and D. Roth, "Privacy adhering machine un-learning in nlp," *arXiv preprint arXiv:2212.09573*, 2022.

[11] S. Reddy, R. Anand *et al.*, "Comparison of model adaptation techniques with machine unlearning," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2024, pp. 1–11.
[12] M. Kurmanji, E. Triantafillou, and P. Triantafillou, "Machine unlearning in learned databases: An experimental analysis," *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–26, 2024.
[13] E. He, T. Sarwar, I. Khalil, X. Yi, and K. Wang, "Deep contrastive unlearning for language models," *arXiv preprint arXiv:2503.14900*, 2025.
[14] D. Zhang, S. Pan, T. Hoang, Z. Xing, M. Staples, X. Xu, L. Yao, Q. Lu, and L. Zhu, "To be forgotten or to be fair: Unveiling fairness implications of machine unlearning methods," *AI and Ethics*, vol. 4, no. 1, pp. 83–93, 2024.
[15] K. Koch and M. Soll, "No matter how you slice it: Machine unlearning with sisa comes at the expense of minority classes," in *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 2023, pp. 622–637.
[16] S. Yu, F. Sun, J. Guo, R. Zhang, and X. Cheng, "Legonet: A fast and exact unlearning architecture," *arXiv preprint arXiv:2210.16023*, 2022.
[17] Y. Quan, Z. Li, and G. Montana, "Efficient verified machine unlearning for distillation," *arXiv preprint arXiv:2503.22539*, 2025.
[18] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, 2022, pp. 499–513.
[19] A. Muresanu, A. Thudi, M. R. Zhang, and N. Papernot, "Unlearnable algorithms for in-context learning," *arXiv preprint arXiv:2402.00751*, 2024.
[20] R. Dilworth, "Privacy preservation through practical machine unlearning," *arXiv preprint arXiv:2502.10635*, 2025.
[21] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.
[22] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.

article tikz amsmath