ERASURE: A Modular and Extensible Framework for Machine Unlearning

ERASURE: A Modular and Extensible Framework for Machine Unlearning

Andrea D'Angelo*
andrea.dangelo6@graduate.univaq.it
University of L'Aquila
L'Aquila, Italy

Claudio Savelli*
claudio.savelli@polito.it
Polytechnic of Turin
Turin, Italy

Gabriele Tagliente
gabriele.tagliente@student.univaq.it
University of L'Aquila
L'Aquila, Italy

Flavio Giobergia
flavio.giobergia@polito.it
Polytechnic of Turin
Turin, Italy

Elena Baralis
elena.baralis@polito.it
Polytechnic of Turin
Turin, Italy

Giovanni Stilo
giovanni.stilo@univaq.it
University of L'Aquila
L'Aquila, Italy

Abstract

Machine Unlearning (MU) is an emerging research area that enables models to selectively forget specific data, a critical requirement for privacy compliance (e.g., GDPR, CCPA) and security. However, the lack of standardized benchmarks makes evaluating and developing unlearning methods difficult. To address this gap, we introduce ERASURE, a benchmarking and development framework designed to systematically assess MU techniques. ERASURE provides a modular, extensible, open-source environment with real-world datasets and standardized unlearning measures. The framework is designed with configuration-driven workflows and an inversion of control architecture, allowing integration of new datasets, models, and evaluation measures. ERASURE advances trustworthy AI research as a tool for researchers to develop and benchmark new MU methods.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Security and privacy** → **Human and societal aspects of security and privacy**.

Keywords

Machine Unlearning, Evaluation Framework, Benchmark

ACM Reference Format:
Andrea D'Angelo, Claudio Savelli, Gabriele Tagliente, Flavio Giobergia, Elena Baralis, and Giovanni Stilo. 2025. ERASURE: A Modular and Extensible Framework for Machine Unlearning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746252.3761627

1 Introduction

Artificial intelligence (AI) models are now widely used across domains to automate tasks and support decision-making. However, their use also challenges how personal data is embedded and used in those systems. A key legal requirement is the *right to be forgotten*, as stated in regulations like the General Data Protection Regulation (GDPR), which mandates the deletion of personal data upon an individual's request [22]. In such cases, the person or organization that owns the AI model is expected to release a new model that no longer uses the removed data. However, training a model from scratch for every request is impractical, especially for larger models such as Large Language Models (LLMs) [7]. These challenges have led to the emergence of Machine Unlearning (MU). The goal of MU methods is to remove the influence of specific data from a model, effectively making it behave as though the data had never been part of its training, all without retraining from scratch [27].

Despite its growing relevance, MU is still underexplored and lacks a widely accepted reference framework. As a result, researchers and practitioners often rely on ad-hoc implementations, hindering the reproducibility and comparability of results. The absence of standardized tools and benchmarks makes evaluating unlearning techniques consistently across tasks and datasets challenging. This fragmented landscape underscores the need for a unified, accessible resource to drive MU research and implementation.

To address this gap, we present **ERASURE**[1], a flexible and modular framework designed to support research in MU. ERASURE simplifies experimentation by organizing all settings within a single JSON configuration file: dataset, model, unlearning method,

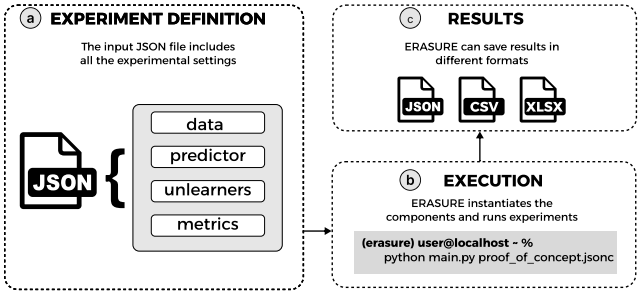

Figure 1: ERASURE in action – The experiment is defined via a single JSON configuration file that specifies required components and is executed by the main script.

*Principal authors of the work who contributed equally.

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761627

[1] Full code, and tutorial available at https://github.com/aiim-research/ERASURE

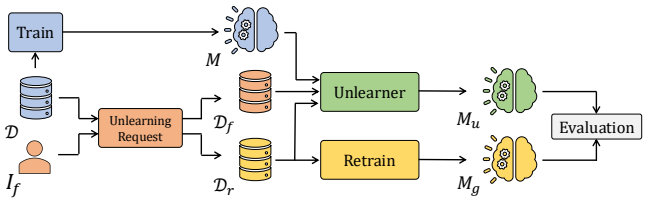

Figure 2: Machine Unlearning Workflow.

and evaluation measures, as shown in Fig. 1, which provides an overview of the experimental workflow. The framework is built upon specialized design patterns tailored to requirements of MU.

ERASURE offers implementations of many state-of-the-art unlearning techniques, ranging from retraining-based approaches to more efficient approximate methods. It supports easy access to widely used data sources such as Hugging Face [15], TorchVision [26], the UCI Repository [17], Torch Geometric [10], and supports multiple data types (image, text, tabular, graph). To assess the performance of unlearning techniques, ERASURE supports a set of evaluation measures that capture their *efficacy*, *utility*, and *efficiency* [14, 18]. It is also designed for easy customization, allowing users to add new methods, datasets, or measures with minimal setup. By bringing these components together, ERASURE aims to simplify experimentation, support reproducibility, and help the community address the growing societal need for responsible AI development.

2 Related Work

Despite increasing interest in MU, no open-source libraries have been designed to support developing, integrating, and evaluating unlearning techniques in diverse data types. Most existing codebases are limited to reproducing methods introduced in individual research papers. These implementations are often related to a particular experimental setup and lack modularity, making it difficult to adapt them to new datasets, models, or evaluation protocols.

Recent benchmark works have released partial code demonstrating the proposed unlearning strategies. For instance, [16], [21], and [25] focus on unlearning in the context of LLMs, providing code that supports basic experimentation with textual data. However, their implementations are highly specialized and not intended for reuse outside the original context. Similarly, [12], [2], and [4] target unlearning in computer vision: as such, their codebases are scoped to specific datasets and do not support other data modalities.

Broader efforts, like [3], examine multiple models and data types, but do not offer a general-purpose framework: the code is focused on evaluating a fixed set of scenarios and is not structured for extensibility or long-term reuse. In all cases, measures implementations, data loading practices, and evaluation pipelines are developed independently, resulting in duplicated effort and low reproducibility.

In contrast, ERASURE provides the first open-source, modular framework dedicated to MU. It is designed with extensibility and reproducibility at its core [8], following principles such as *Inversion of Control* and the *Factory Pattern* to promote clean separation of components. This design makes ERASURE a scalable and replicable foundation for future research in Machine Unlearning.

3 ERASURE Framework

This section introduces the design principles behind ERASURE. Section 3.1 presents the Machine Unlearning workflow that guided key decisions; Section 3.2, the core design principles; and Section 3.3, a sample configuration.

3.1 Machine Unlearning Workflow

Fig. 2 illustrates a typical workflow in MU. In this setting, an unlearning request is issued by an entity I_f, which triggers the partitioning of the original dataset $\mathcal{D}$ into a Forget Set $\mathcal{D}_f$ and a Retain Set $\mathcal{D}_r$, containing respectively the data samples to be removed and those to be preserved. An unlearning algorithm (Unlearner) operates on the original model M, which was trained on $\mathcal{D}$. Using both $\mathcal{D}_f$ and $\mathcal{D}_r$, it produces an updated model M_u in which the influence of $\mathcal{D}_f$ should be removed. The objective is to make M_u as similar as possible to the Gold Model M_g, which is trained from scratch using only $\mathcal{D}_r$. The similarity between M_u and M_g is typically assessed using a set of measures that capture utility, efficacy, and efficiency.

ERASURE implements all steps of this workflow by instantiating all the necessary objects. Specifically, the **Data Management** module of ERASURE is designed to meet the unique requirements of Machine Unlearning. All operations occur within the `DataManager` class, which orchestrates three main steps: *(i)* data loading, *(ii)* preprocessing, and *(iii)* partitioning. Partitioning the data is critical for Machine Unlearning, as it often involves more sophisticated strategies than conventional train/test splits, as shown in Fig. 2.

To support all the possible unlearning scenarios, ERASURE introduces a modular `DataSplitter` strategy that allows users to configure complex dataset partitions. Each `DataSplitter` defines a specific logic, such as selecting a percentage of the data, extracting a particular class, or filtering by a secondary label (e.g., a specific identity or topic to be forgotten). These actions can be executed sequentially, with each one optionally referencing the output of previous ones, enabling chained and hierarchical dataset splits. This design defines unlearning requests at various granularities, such as removing all samples from a particular user or forgetting only a subset of examples tied to a specific semantic label.

The **Predictor** component M is instantiated directly from the main configuration file. It is either a model that external libraries provide or a custom implementation. The configuration interface allows users to define all essential hyperparameters, such as the number of hidden layers, training epochs, and the optimizer. The optimizer is instantiated as a separate, fully configurable entity.

The **Unlearner** module represents the core abstraction for all unlearning methods within ERASURE. Each unlearning method extends this base class, encapsulating its algorithmic logic while maintaining modularity. To preserve the integrity of the original model, every Unlearner instance operates on an isolated copy of M.

Finally, the **Evaluator** module integrates all essential components for assessing the performance of unlearning techniques. To facilitate large-scale experimentation, the `Evaluator Manager` automates the setup, memory management, and sequential execution of all configured `Measures`. Results are collected through a centralized `Evaluation` object, ensuring consistency and traceability across experiments. They can be saved in a variety of file formats.

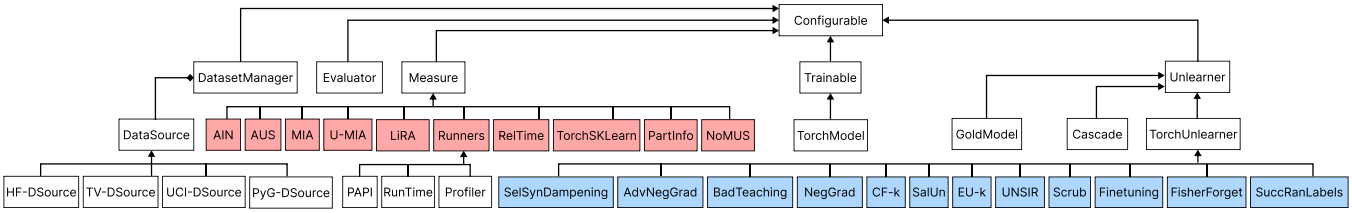

Figure 3: Overview of the main classes of the ERASURE Unlearning Framework and their hierarchical relations.

3.2 Design Principles

Inversion of Control (IoC) [20] is a software design pattern that promotes modularity and ease of maintenance by decoupling object creation and control flow from the client logic. ERASURE adopts IoC to enable dynamic and extensible dependency management, allowing components to be instantiated from standard libraries (e.g., PyTorch or scikit-learn) and custom modules, including datasets, models, unlearning methods, or evaluation measures.

Factory Pattern [23] is another core design strategy adopted by ERASURE. It handles object creation through a centralized factory, making building different types of objects easier, especially when they have many parameters or dependencies. In ERASURE, most components are implemented as `Configurable` classes (see Fig. 3, showing the classes' hierarchy, for reference), meaning their parameters can themselves be instantiated via configuration. For example, a `Predictor` may include an `Optimizer` as one of its parameters, which can be further customized (e.g., specifying the learning rate or weight decay) directly through the configuration file. This nested configuration capability is made possible by ERASURE's implementation of the Factory Pattern, enabling highly flexible and declarative experiment setups.

Beyond Reproducibility. ERASURE provides a wide selection of ready-to-use data sources, models, unlearning methods, and evaluation measures (a complete list is available on the GitHub Repository, or in Fig. 3). At the same time, ERASURE is easily extensible: developers can integrate their components by subclassing the appropriate base classes and following the interface structure.

For example, adding a custom unlearning method only requires extending the base `Unlearner` class and defining the logic for model preparation, unlearning, and optional post-processing steps. Once implemented, the custom component can be used in experiments simply by referencing its import path and configuration parameters in the Main Configuration file. Moreover, multiple Unlearners can be concatenated using ERASURE's `compose` feature, enabling exploration of method combinations beyond those in the literature.

3.3 Example of Main Configuration

The structure of the main configuration, illustrated in Listing 1, is based on four core components: `data`, `predictor`, `unlearners`, and `evaluator`. These directly correspond to the elements outlined in the Machine Unlearning workflow of Fig. 2. Each module requires a `class` field specifying its implementation and a `parameters` field defining the initialization configuration.

The `data` section (Lines 1-4) defines the dataset and how to partition it. This includes selecting a data source and applying preprocessing (e.g., filtering) via modular splitting components.

```
1 "data": {"class":"erasure.data.<NS>.DatasetManager",
2        "parameters": {
3            "DataSource": { ... },
4            "partitions": [ "p_1", "p_2", ... , "p_n" ] }},
5 "predictor": {"class": "erasure.model.<NS>.TorchModel",
6   "parameters": {
7     "optimizer": {"class": "torch.optim.Adam"},
8     "loss_fn": {"class": "torch.nn.CrossEntropyLoss"},
9     "model": {"class": "erasure.<NS>.BERTClassifier"}}},
10 "unlearners":[
11    {"compose_gold" : "configs/snippets/u_gold.json"},..,
12    {"class":"erasure.<NS>.AdvancedNegGrad",
13        "parameters": {
14             "epochs": 1,
15             "ref_data_retain": "retain",
16             "ref_data_forget": "forget",
17             "optimizer": {"class": "torch.optim.Adam","parameters": {"lr":
                    0.0001}}} }],
18 "evaluator":{
19   "class": "erasure.evaluations.<NS>.Evaluator",
20   "parameters": { "measures":[ ... ] } }
```

Listing 1: Structure of the main configuration snippet for an ERASURE experiment. $\langle NS \rangle$ **compacts sub-modules namespace.**

The `predictor` section (Lines 5-9) specifies the model and all related settings. The example shown uses a PyTorch-based model, where the optimizer and loss function can be selected directly from existing libraries, enabling easy reuse without additional code.

The `unlearners` (Lines 10-17) field is a list that defines the unlearning methods to be applied and then evaluated. Each entry includes its configuration, such as the relevant data partitions and optimization settings. All `unlearners` are executed independently to ensure isolation. The `compose_` feature, shown in this section, enables loading JSON snippets from external files for any configuration component, allowing users to easily reuse predefined setups across data sources, models, unlearners, and evaluation measures.

Lastly, the `evaluator` section (Lines 18-20) defines the measures. Although the specific measures are omitted for brevity, they are fully customizable and follow the same configuration logic.

4 ERASURE in action

To demonstrate the practical utility and flexibility of ERASURE, we performed proof-of-concept experiments with several state-of-the-art evaluation measures (red boxes in Fig. 3) on four datasets spanning the supported domains (i.e., Tabular, Images, Text, and Graph). In Table 1 we report the results for two unlearning methods selected from the 12 included in ERASURE (blue boxes in Fig. 3), which represent two of the most popular unlearning paradigms. Full configurations are included in the repository[1].

Methods. Selective Synaptic Dampening (SSD) [11] modifies model weights directly, demonstrating the framework's support for model

DS	Unlearner	$\Delta Acc_t \downarrow$	$\Delta Acc_f \downarrow$	$\Delta Acc_r \downarrow$	$\Delta AUS \downarrow$	$\Delta AIN \downarrow$	$\Delta UMIA \downarrow$	NoMUS ↑	Time (Speedup) ↓
Tabular	`Orig.`	.001 ± .000	.041 ± .002	.006 ± .003	.014 ± .004	.998 ± .000	.009 ± .003	.916 ± .001	-
	`Gold`	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.924 ± .002	461.688 ± .1 (1.0×)
	`SSD` [11]	.001 ± .000	.041 ± .002	.006 ± .003	.014 ± .004	.998 ± .000	.009 ± .004	.915 ± .001	11.889 ± .1 (38.8×)
	`SalUn` [9]	.006 ± .003	.023 ± .003	.022 ± .002	.012 ± .001	.594 ± .089	.003 ± .002	.928 ± .001	2.824 ± .0 (163.5×)
Images	`Orig.`	.003 ± .001	.034 ± .002	.003 ± .002	.035 ± .004	.885 ± .156	.031 ± .007	.918 ± .007	-
	`Gold`	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.949 ± .005	1598.795 ± 13.6 (1.0×)
	`SSD` [11]	.007 ± .003	.032 ± .005	.007 ± .005	.041 ± .005	.885 ± .156	.031 ± .006	.916 ± .001	102.026 ± .8 (15.7×)
	`SalUn` [9]	.002 ± .001	.007 ± .000	.014 ± .002	.011 ± .004	1.770 ± .951	.007 ± .004	.949 ± .003	42.110 ± .8 (38.0×)
Text	`Orig.`	.042 ± .008	.035 ± .000	.052 ± .010	.104 ± .016	.859 ± .104	.090 ± .014	.796 ± .008	-
	`Gold`	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.907 ± .006	8454.617 ± 48.6 (1.0×)
	`SSD` [11]	.036 ± .009	.033 ± .003	.045 ± .012	.093 ± .016	.568 ± .304	.071 ± .019	.817 ± .008	1736.043 ± .1 (4.9×)
	`SalUn` [9]	.004 ± .003	.020 ± .005	.011 ± .002	.037 ± .020	.221 ± .157	.019 ± .009	.887 ± .011	2713.718 ± 4.9 (3.1×)
Graph	`Orig.`	.007 ± .001	.002 ± .001	.008 ± .004	.002 ± .001	.067 ± .094	.008 ± .004	.896 ± .003	-
	`Gold`	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.000 ± .000	.897 ± .004	12.689 ± .2 (1.0×)
	`SSD` [11]	.007 ± .001	.002 ± .001	.008 ± .004	.002 ± .001	.067 ± .094	.006 ± .001	.900 ± .003	1.808 ± .0 (7.0×)
	`SalUn` [9]	.005 ± .003	.014 ± .005	.006 ± .009	.014 ± .005	.067 ± .094	.007 ± .008	.896 ± .008	1.957 ± .0 (6.5×)

Table 1: Experimental comparison of different unlearning methods across various datasets (tabular, image, textual, and graph), with 3 runs for each experiment. For each metric and dataset, the best result is in bold. `Orig.` and `Gold` models are highlighted.

editing methods. In contrast, `SalUn` [9] leverages a saliency map to identify and apply unlearning only on the weights most activated by the forget set. This exemplifies how ERASURE supports unlearning on specific model subcomponents, while its `compose` feature allows the integration of multiple unlearning methods in cascade.

Datasets and models. We selected a widely used and representative dataset for each domain: Adult [1] for Tabular, CelebA [19] for Images, AG News [13] for Text, and BBBP [24] for Graph. We employed a two-layer network for Tabular, a ResNet18 for Images, a BERT model for Text, and a two-layer GCN for the Graph domain.

Measures. We employed several state-of-the-art measures for MU evaluation. For *utility*, we report the difference in Accuracy on the test, forget, and retain sets (ΔAcc_t, $\Delta\ Acc_f$, ΔAcc_r) with respect to the `Gold` baseline. Similarly, for *efficacy*, we compute the change in the Adaptive Unlearning Score [6] (ΔAUS), Anamnesis Index [5] (ΔAIN), and Membership Inference Attack [4] (ΔUMIA) effectiveness compared to the `Gold` model. We use the Normalized Machine Unlearning Score [4] (NoMUS) as a comprehensive metric combining *utility* and *efficacy*, and adopt it to select the best unlearning configuration for each method and domain. Lastly, for *efficiency*, we report the Time and the Speedup with respect to complete model retraining (referred to as the `Gold` model in Table 1).

Outcomes. In terms of *utility*, both `SSD` and `SalUn` maintain high model performance, with accuracy degradation that is often negligible across the different domains. For *efficacy*, `SalUn` performs generally better across all domains except Graph, where the compound NoMUS score is slightly lower. The Text domain exhibits the most significant performance gap: `SalUn` severely outperforms `SSD` on Text across all *efficacy* measures. It must be noted that, since both methods are state-of-the-art, they still achieve high NoMUS scores on all domains. However, the Text domain seems the hardest to unlearn, possibly due to bigger, more complex models. Lastly, in terms of *efficiency*, `SalUn` is faster than `SSD` on all domains except Text. This is expected, as fine-tuning a large model is more time-consuming than modifying its weights directly. Nonetheless, both methods have substantial Speedups across all domains relative to the `Gold` model. This improvement is particularly critical in the context of MU, where the practicality of a technique is undermined if its computational cost exceeds that of retraining from scratch.

Highlights. While the results cover only a subset of what ERASURE supports, they provide meaningful insights. Methods like `SSD` and `SalUn` show that high predictive performance can be maintained after unlearning. However, no single approach consistently excels across all evaluation criteria, highlighting the inherent trade-offs between *utility*, *efficacy*, and *efficiency*. However, the significant Speedup obtained reinforces the core motivation behind MU itself.

5 Conclusions

ERASURE introduces a unified, extensible platform for MU, addressing a critical need for reproducible research. Using results from two representative methods across four domains as a proof of concept, we demonstrate the framework's ability to support diverse experimental setups at scale. ERASURE's modular design enables the integration of custom datasets, methods, and evaluation measures with minimal overhead. We believe this platform will accelerate progress in the field by providing researchers with the tools needed to build, compare, and refine unlearning methods efficiently.

Acknowledgements

The work is partially funded by the "GenXAI - Generative Artificial Intelligence for eXplainability" project of Univaq and by the European Union Next-Generation EU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) Projects: "Enhanced Network of intelligent Agents for Building Livable Environments - ENABLE", CUP E13C24000430006, and FAIR - Future Artificial Intelligence Research (CUP PE00000013).

GenAI Usage Disclosure

During the preparation of this work, the authors used ChatGPT-4o to correct typos and grammatical mistakes. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

[1] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[2] Xavier F Cadet, Anastasia Borovykh, Mohammad Malekzadeh, Sara Ahmadi-Abhari, and Hamed Haddadi. 2024. Deep Unlearn: Benchmarking Machine Unlearning.

[3] Jiali Cheng and Hadi Amiri. 2024. Mu-bench: A multitask multimodal benchmark for machine unlearning.

[4] Dasol Choi and Dongbin Na. 2023. Towards machine unlearning benchmarks: Forgetting the personal identities in facial recognition systems.

[5] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2023. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security* 18 (2023), 2345–2354. doi:10.1109/TIFS.2023.3265506

[6] Marco Cotogni, Jacopo Bonato, Luigi Sabetta, Francesco Pelosin, and Alessandro Nicolosi. 2023. Duck: Distance-based unlearning via centroid kinematics. doi:10.48550/arXiv.2312.02052

[7] Kate Crawford. 2022. Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence.

[8] Andrea D'Angelo, Claudio Savelli, Gabriele Tagliente, Flavio Giobergia, Elena Baralis, and Giovanni Stilo. 2025. How to Make Reproducible Research in Machine Unlearning with ERASURE. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, James Kwok (Ed.). International Joint Conferences on Artificial Intelligence Organization, 11025–11029. doi:10.24963/ijcai.2025/1255 Demo Track.

[9] Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. 2023. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation.

[10] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. arXiv:1903.02428 [cs.LG] https://arxiv.org/abs/1903.02428

[11] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast machine unlearning without retraining through selective synaptic dampening. 12043–12051 pages.

[12] Keltin Grimes, Collin Abidi, Cole Frank, and Shannon Gallagher. 2024. Gone but Not Forgotten: Improved Benchmarks for Machine Unlearning.

[13] Antonio Gulli. 2005. AG's Corpus of News Articles. http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html. Accessed: 2025-05-31.

[14] Jamie Hayes, Ilia Shumailov, Eleni Triantafillou, Amr Khalifa, and Nicolas Papernot. 2024. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy.

[15] HuggingFace. 2025. Hugging Face: Natural Language Processing and Machine Learning Tools. https://huggingface.co.

[16] Zhuoran Jin, Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, and Jun Zhao. 2024. RWKU: Benchmarking Real-World Knowledge Unlearning for Large Language Models.

[17] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. 2025. The UCI Machine Learning Repository. https://archive.ics.uci.edu. Accessed: January 22, 2025.

[18] Alkis Koudounas, Claudio Savelli, Flavio Giobergia, and Elena Baralis. 2025. " Alexa, can you forget me?" Machine Unlearning Benchmark in Spoken Language Understanding.

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild.

[20] Jan Machacek, Aleksa Vukotic, Anirvan Chakraborty, and Jessica Ditt. 2008. Introducing Inversion of Control. 31–72 pages.

[21] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms.

[22] Alessandro Mantelero. 2013. The EU Proposal for a General Data Protection Regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review* 29, 3 (2013), 229–235.

[23] Steven John Metsker. 2002. Design patterns Java workbook.

[24] Hiroshi Sakiyama, Masaki Fukuda, and Yasushi Okuno. 2021. Prediction of Blood-Brain Barrier Penetration (BBBP) Based on Molecular Descriptors of the Free-Form and In-Blood-Form Datasets. *Molecules* 26, 24 (Dec. 2021), 7428. doi:10.3390/molecules26247428

[25] Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. 2024. Muse: Machine unlearning six-way evaluation for language models.

[26] TorchVision. 2025. TorchVision: PyTorch's Computer Vision Library. https://pytorch.org/vision.

[27] Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. 2024. Machine Unlearning: A Survey. *Comput. Surveys* 56, 1 (2024), 9:1–9:36.