Technische Hochschule Ingolstadt

University of Applied Science Ingolstadt

Faculty Computer Science

Degree Artificial Intelligence

Bachelor Thesis

Topic: Selective Data Removal in Machine Learning Models: A Study on Machine Un-Learning

First Name, Surname: Moritz, Kastler

issued: 2024/11/20

turned in: 2025/01/31

First examiner: Prof. Dr.-Ing Munir Georges

Second examiner: Prof. Dr. Sören Gröttrup

Technische Hochschule Ingolstadt

Declaration

I hereby declare that this thesis is my own work, that I have not presented it elsewhere for examination purposes and that I have not used any sources or aids other than those stated. I have marked verbatim and indirect quotations as such.

Place, Date

First name, Surname
(Signature)

Technische Hochschule Ingolstadt

Abstract

In an era where data privacy and ethical considerations are increasingly crucial, the ability to selectively remove information from machine learning models—known as machine unlearning—has become a significant research challenge. This thesis explores methods for selective data removal, focusing on gradient-based approaches. It provides an overview of existing unlearning techniques and metrics, classifies them based on interference time and , and introduces a new algorithm: Generator Feature Machine Unlearning (GeFeU). Building on the Fast Yet Efficient Machine Unlearning (FEMU) approach, GeFeU improves upon existing methods by incorporating a noise generator and refining the target unlearning process to subsets within a class. The study evaluates GeFeU's effectiveness using multiple datasets (MNIST, Colored MNIST, FashionMNIST) and compares its performance against baseline models and prior unlearning algorithms. Results indicate that GeFeU successfully removes targeted feature representations while maintaining overall model integrity if prior fine-tuning takes place. Otherwise the unlearning skews to the model's parameters. All of the code can be found here:

- Main Repository of Bachelor Study for Moritz Kastler:
  https://github.com/MoraiT01/study_on_unlearning
- Noise Generator Testing:
  https://github.com/MoraiT01/Fast-Machine-Unleaning-Prop

Glossary

CNN Convolutional Neural Network (CNN) is a regularized type of feed-forward neural network that learns features by itself via filter (or kernel) optimization [1]. 1, 3, 25

Colored MNIST Colored Version of MNIST. 9, 24, 25, 28–30, 32, 35, 36, 38

FashionMNIST Well-known dataset consisting of different clothing products. 9, 22, 24, 25, 28, 30, 34–36, 38

FEMU The algorithm proposed in the *Fast Yet Efficient Machine Unlearning* paper [2]. The authors of this paper did not give this name, we introduced it to make it easier to refer to. Stems from *F*ast Yet *E*fficient *M*achine *U*nlearning. 1, 2, 7, 8, 10, 11, 15–17, 19–23, 31, 38

FEMU+Gen The algorithm proposed in the *Fast Yet Efficient Machine Unlearning* paper [2], main difference to FEMU being the substitution of noise batch with noise generator. 10, 18–22, 24, 26–29, 32, 33, 35–39

GDPR The General Data Protection Regulation (Regulation (EU) 2016/679). European Union regulation on information privacy in the European Union and the European Economic Area [3]. 1, 6, 8, 17, 31

GeFeU The algorithm proposed in this bachelor thesis. Stems from *Ge*nerator *Fe*ature Machine *U*nlearning. 8, 10, 18, 22–24, 26–40

LLM Large Language Models, a type of machine learning model designed for natural language processing tasks such as language generation. LLMs are language models with many parameters, and are trained with self-supervised learning on a vast amount of data [4]. 12–14, 40, 41

MLP Multilayer Perceptron, a feedforward neural network consisting of fully connected neurons with nonlinear activation functions, organized in layers [5]. 1, 3, 25, 40

MNIST Well-known dataset consisting of handwritten numbers. 7, 9, 22, 24–28, 30–32, 35, 36, 38

MU The process of removing specific data's influence from a trained model, ensuring it behaves as if the data was never used. 2, 5, 10, 11, 13, 14, 16, 22, 24, 26, 28–30, 32, 35–40

Technische Hochschule Ingolstadt

Contents

List of Figures

List of Tables

Part I

Introduction

1 Motivation

Data when handled accordingly can create algorithms, for classifying street sights (as in [6]) or analysing natural language (current trends for Natural Language Processing [7]), and interesting insights, about the behaviour of animals (as in [8]), humans or machine learning models themselves.

With the emergence of *Big Data*, the ethical and practical implications of incomprehensible amounts of data come into sharp focus, as discussed in [9]. As Cathy O'Neil discusses in her book "Weapons of Math Destruction" [10] data, when handled poorly, can cause self-confirming echo chambers, which can end in a human-harming downward spiral. Especially sensitive data (described by [11]) has the potential to ignite cases of discrimination and unfair behaviour, as shown by the reported cases in the book [10]. Such Data shall be handled carefully, and everyone should have full control over their data, as stated by the "GDPR".

The ability to selectively remove information from a trained model poses both a profound technical challenge and an essential societal need, discussed in [12].

Structure and Objectives To establish a comprehensive understanding, I will begin by talking about the fundamentals of machine learning. This includes a little overview of core principles, moving the focus towards supervised learning and classification tasks. When it comes to neural networks and their architecture, special attention will be given to multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs), as these architectures are particularly suited for image data, the type of input I will work with throughout this thesis. By grounding the study in these foundations, I aim to provide a clear basis for the more advanced concepts discussed later.

Building on this groundwork, I will transition to recent developments in machine unlearning. The rising interest in this field underscores the need for efficient and scalable approaches. Again, I will give an overview of various techniques, while putting the focus on gradient-based methods, which align well with neural networks. This exploration will lead to the specific challenge addressed in this thesis: unlearning a subset of a particular class within a classification task. While previous work, such as the "Fast Yet Efficient Machine Unlearning" (FEMU) algorithm of the paper [2], has demonstrated promising results for class-level unlearning, my goal is to adapt and extend these ideas to enable feature unlearning, a finer-grained approach tailored to subsets within a class, united by a joint feature.

Methodological Approach To achieve this, I will first analyze the FEMU algorithm, highlighting its strengths and limitations. This analysis will inform the development of my proposed algorithm, which modifies FEMU to handle subset-level unlearning. The goal I want to achieve lies in the selective removal of a feature associated with specific subsets, enabling models to forget patterns without compromising overall performance. This approach represents a midway point between the unlearning of entire classes versus the unlearning of single samples. The second type is where I want to head in future works, for a different use-case like language generation and/or models like transformers.

This study is empirical. I will evaluate the proposed algorithm across three datasets, utilizing 30 models per type of model. This evaluation aims to provide robust evidence of the algorithm's effectiveness and practical utility.

Contribution and Impact Through this work, I aspire to advance the understanding of machine unlearning by introducing the concept of feature unlearning and demonstrating its feasibility in practice. Ultimately, it reflects my commitment to leveraging machine learning not only to build intelligent systems but also to ensure their adaptability and alignment with human values.

2 Fundamentals of Machine Learning

To have a better understanding of MU, we want to create a general overview of the fundamentals of Machine Learning, the steps, which need to be taken, its goals and the challenges.

2.1 Machine Learning

Tasks, Models and Data Machine learning is a subset of artificial intelligence focused on building systems that can learn from data and improve their performance over time without explicit programming. At its core, Machine Learning involves three primary tasks: regression, classification, and clustering. Regression tasks aim to predict continuous outcomes, while classification tasks focus on assigning inputs to discrete categories. Clustering, on the other hand, involves grouping similar data points without predefined labels.

A wide range of model types is used to address these tasks, including decision trees, support vector machines, and neural networks. These models must work with diverse data types, ranging from numerical and categorical data to text, images, and time-series data. Among these, image data presents unique challenges and opportunities, particularly in tasks involving computer vision, where convolutional neural networks (CNNs) excel due to their ability to capture spatial hierarchies.

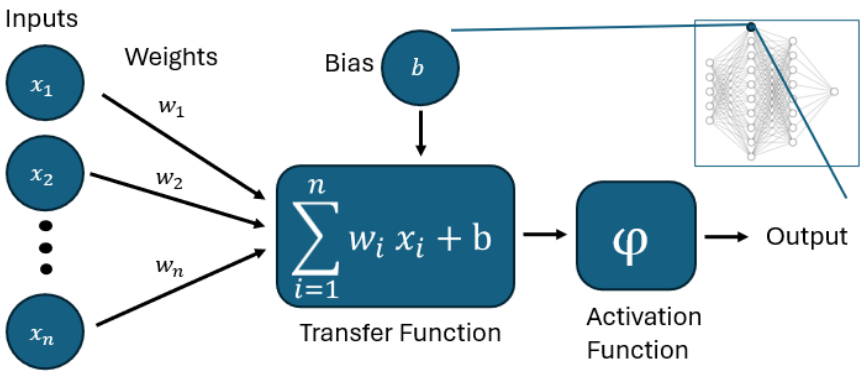

Figure 1: Each neuron includes as the input all of the previous layer's neurons weighted by importance, summed along with the neuron's bias, and then passed to the activation function. The activation function "decides" whether the neuron's output should be sent to the next layer. [13]

Network Architecture Neural networks, particularly multi-layer perceptrons (MLP) and CNNs, form the backbone of many Machine Learning applications. These networks consist of interconnected neurons, each representing a mathematical function, as seen in Figure 1. The structure of a neural network is defined by layers of neurons, where each neuron applies an activation function to its weighted inputs and adds a bias term. These weights and biases are the parameters learned during training and therefore key to how a neural network "remembers". The activation function is added after the transfer function to enable the neural network to learn non-linear relationships.

Model Training With the architecture, the playground for learning is set, now the rules must be set. At its core, when we train a neural network we simply want it to minimize or maximize a *loss*, which we define passed on the task at hand, e.g. for a Regression we might consider a *Mean Square Error* and for classification a *Cross Entropy Loss*, further explained in chapter 3.2.2 and 8.6 of [14].

Loss functions can be customized however we see fit. This includes e.g. a term that punishes big weights, which is used as a *regularization* measure, further explained in chapter 5.5 of [14].

How the models know how to min or max the loss is based on the optimization algorithm employed during training. Most commonly, variants of the *gradient descent* algorithm are used. These methods rely on computing the gradient of the loss function with respect to the model's parameters and adjusting the parameters in the direction that minimizes the loss, as visualized in Figure 2. For example, *Stochastic Gradient Descent (SGD)* is a widely used approach that updates the parameters based on individual or small batches of training

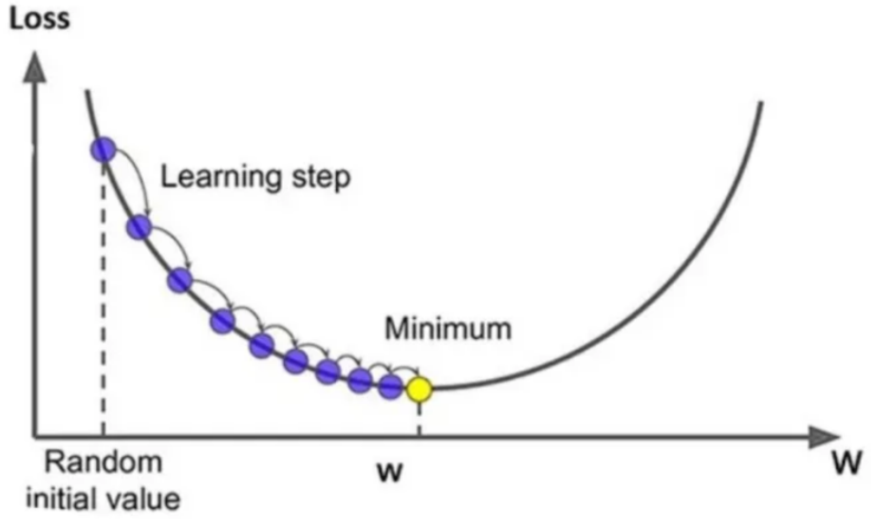

Figure 2: With every Iteration the Weights and Biases are adjusted towards an Optimum. [15]

samples, balancing efficiency and convergence stability.

Additionally, advanced optimizers like *Adam* or *RMSProp* integrate techniques such as adaptive learning rates and momentum to enhance convergence rates and handle non-stationary objectives more effectively, mentioned in chapter 4.2 of [14].

Regularization plays a critical role in controlling overfitting and improving generalization. This can be incorporated not only in the loss function but also through techniques like *dropout*, *batch normalization*, or weight decay. These measures ensure that the model remains robust even when exposed to noisy or unseen data.

Epochs and Updates: An epoch refers to a complete pass through the entire training dataset during the training process. However, since most datasets are large, the data is often divided into smaller subsets called batches. Each batch is used to update the model parameters, and this update is known as an iteration. The number of updates per epoch is determined by the batch size, calculated as $\textit{Number of Updates per Epoch} = \frac{\textit{Training Samples}}{\textit{Batch Size}}$. Larger batch sizes reduce the number of updates per epoch but may require more memory, while smaller batches lead to more frequent updates and can improve generalization and convergence.

The learning rate α is one of the most critical hyperparameters, controlling the step size during parameter updates. A higher learning rate allows faster convergence but risks overshooting the optimal solution, while a lower learning rate may lead to slower convergence or getting stuck in local minima. Strategies like learning rate schedules or adaptive learning rates (e.g., in optimizers like Adam) dynamically adjust α during training, often starting with a higher value and reducing it as training progresses to refine the model's performance,

as explained by chapter 4.2 of [14].

And then there also is the random initialization, basically your starting point of the optimization process. The initial values of the weights and biases can impact the convergence behaviour and final performance. This means that every neural network can be different from one another, even if they have the same architecture and were trained the same way. Poor initialization may lead to slow convergence or getting stuck in suboptimal solutions. Techniques like *Xavier* or *He initialization* are commonly employed to ensure that the starting weights are scaled appropriately, promoting stable gradient flow during training, as discussed [16].

Hyperparameter Optimization Hyperparameter optimization is an essential aspect of training neural networks, as it also determines the performance of the final model. Hyperparameter tuning can significantly impact the final performance of the model, and methods like grid search, random search, or even automated approaches like Bayesian optimization are commonly employed for this purpose. Tools like Optuna facilitate this process by automating the search for optimal hyperparameters, such as learning rates, batch sizes, and the number of neurons in each layer.

By understanding how neural networks learn, we gain insights into how they can be made to unlearn. This foundation sets the stage for exploring advanced techniques in MU, which aim to selectively remove learned information while preserving the integrity of the remaining model.

Evaluation Evaluating machine learning models is a critical step to assess their performance and ensure their suitability for the intended task. This process typically involves dividing the available dataset into training, validation, and test sets to avoid overfitting and ensure unbiased evaluation. Key metrics vary depending on the task: for regression, common metrics include Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), while classification tasks rely on metrics like accuracy, precision, recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (AUROC). Cross-validation, particularly k-fold cross-validation, is frequently employed to obtain more reliable performance estimates by averaging results across multiple data splits. Additionally, confusion matrices and error analysis help identify specific areas where the model struggles, guiding further improvements. Beyond numerical metrics, robustness to adversarial examples, fairness across subgroups, and computational efficiency are increasingly important evaluation criteria, especially for real-world deployment. More information to these metrics can be found in [14].

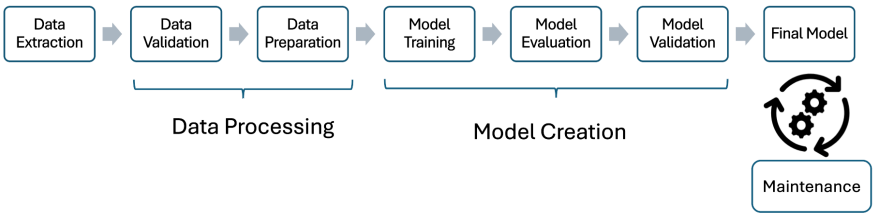

Figure 3: Basic Overview of the Standard Machine Learning pipeline

2.2 Challenges in Machine Learning

The Machine Learning Pipeline, shown in Figure 3, comprises several key stages, each of which is critical to building robust and reliable models while bringing their own challenges with them. These challenges are as follows:

Data Quality and Availability: [17] High-quality data is crucial for building reliable Machine Learning models. However, real-world data is often incomplete, noisy, or imbalanced, which can adversely affect model performance. Access to sufficient labelled data can be a bottleneck, especially in supervised learning tasks.

Overfitting and Generalization: [18] A model may perform well on training data but fail to generalize to unseen data, leading to overfitting. Striking a balance between model complexity and performance remains a persistent challenge.

Scalability: [19] As datasets grow larger and more complex, training and deploying Machine Learning models become computationally expensive and time-intensive. Efficient algorithms and hardware acceleration are critical to address these issues.

Explainability and Interpretability: Many Machine Learning models, particularly neural networks, are often described as "black boxes," making it difficult to interpret their decisions. This lack of transparency is a barrier in domains requiring accountability, such as healthcare and finance, discussed in [20].

Bias and Fairness: Models can inadvertently learn and propagate biases present in the training data, leading to unfair or discriminatory outcomes [10]. Ensuring fairness and equity in Machine Learning systems is a significant ethical and technical challenge.

Ethical and Privacy Concerns: [9] The use of sensitive data raises concerns about user privacy and compliance with regulations like GDPR. We produce a lot of data daily online

and following the saying "when it is free, you are the product" many of these online platforms optimize their use to gain more screen time or make it more likely for you to buy a product.

Hyperparameter Tuning: [21] Optimizing hyperparameters like learning rates, batch sizes, and model architecture can be time-consuming and require expertise. Automated tools like Optuna can help but are not universally applicable.

Adversarial Attacks: [22] Machine learning models are vulnerable to adversarial examples—inputs designed to deceive the model into making incorrect predictions. Building robust models that can withstand such attacks remains a challenge.

Deployment and Maintenance: [23] Transitioning models from research to production involves addressing practical concerns like latency, scalability, and integration with existing systems. Moreover, models must be monitored and updated over time to maintain their performance.

Domain Adaptation and Transfer Learning:[24] Models trained on one domain often struggle to perform well in another due to domain-specific variations. Transfer learning and domain adaptation techniques aim to mitigate this issue but are not always straightforward to implement.

Energy Efficiency: [25] Training large-scale models, especially deep neural networks, consume significant computational resources, contributing to environmental concerns. Efficient algorithms and hardware are critical to reducing the carbon footprint of Machine Learning.

Unlearning and Model Updates: [26] As we will try to emphasize, selectively updating or removing knowledge from a model without retraining from scratch is a complex and emerging challenge in Machine Learning.

Out of these challenges, some are particularly relevant for the emerging field of Machine Unlearning. Particularly:

1. Data Quality and Privacy Concerns
2. Overfitting and Generalization
3. Bias and Fairness
4. Deployment and Maintenance

5. Unlearning and Model Updates

The next section gives a general overview of recent studies that explore solutions to these problems. We will highlight advancements in selectively removing learned information while maintaining overall model performance, to move us into our conducted empirical research.

Part II

Main Section

3 Recent Research on Machine Unlearning

Machine Unlearning has emerged as a critical area of study, especially in light of increasing concerns over data privacy and ethical considerations in Machine Learning. This section provides an overview of the field, highlighting key research, methodologies, and challenges.

When it comes to getting started in the field of Machine Unlearning, we found that [27], [28] and [26] offer a great overview of the field. Despite existing work in the field, we aim to highlight various approaches, ideas, metrics, and challenges that arise in the context of unlearning. Furthermore, we attempt to categorize them based on our understanding of the field.

3.1 Importance of Machine Unlearning

Machine Unlearning addresses crucial concerns such as compliance with data privacy regulations like GDPR, enhancing trust in ML systems, and improving the adaptability of models. The ability to selectively remove information enables ML systems to meet ethical and legal standards while maintaining efficiency and flexibility.

Unlearning can also mitigate the influence of corrupted or biased samples on a model. For instance, removing erroneous data points can enhance a model's fairness and robustness. This is essential in applications where data evolves over time, such as recommendation systems or fraud detection.

Why Machine Unlearning Matters

- **Ethical and Legal Compliance:** Adherence to laws like GDPR and ensuring users' rights to data deletion.
- **Enhancing Adaptability and Efficiency:** Streamlining model updates without full retraining.

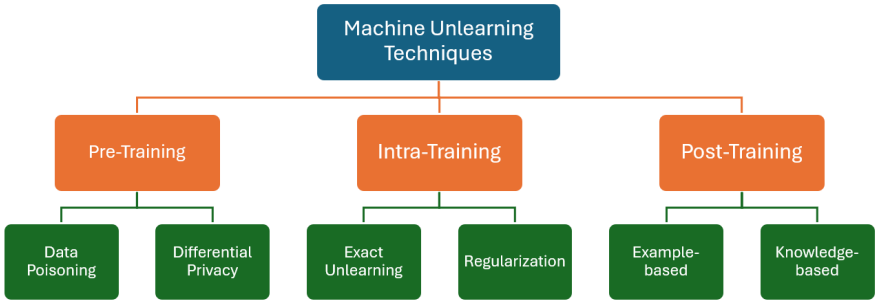

Figure 4: Categorization of Machine Unlearning Approaches based on the time of interference.

How does a Model forget? To understand how a model forgets, it is essential to first explore how it "remembers." Let's show it according to an extreme case, Overfitting. An overfitted model memorizes specific details of the training data, effectively "learning by heart." It encodes intricate patterns, noises, and outliers in its weights and biases, rather than generalizing from the data. Its ability to memorize makes it possible to understand the influence of individual data points on the model's structure and predictions. This would mean, that, in the extreme case, all the details of the training samples are implicitly stored in the model. Data Extraction Attacks could have it easier to learn about the used data, discussed in [29].

Such insights are critical in Machine Unlearning, as they inform methods to selectively remove or negate specific data influences while preserving overall model performance.

On the other side, underfitting a model makes it more difficult for a data extraction attack. Other papers such as [30] even mention regularization as an unlearning measure or proposing an underfitted model by design.

Since we are already talking about potential ideas and algorithms, we want to present a few approaches from other papers.

3.2 Unlearning Algorithms

Unlearning algorithms are designed to selectively remove the impact of specific data points. They can be categorized based on their approach and use cases. In this bachelor thesis, we propose a different categorisation based on the interference time. The hierarchy is shown in figure 4.

Our goal was not to show a complete list of known algorithms, but rather offer a new view for sorting the ideas for unlearning (in green) into different groups (orange). If we found any noteworthy algorithms which tried to implement these ideas, we listed them below in their respective paragraph.

Data Poisoning: [27] It involves intentionally altering the training data to embed or remove specific patterns. For unlearning, this technique manipulates the dataset to either nullify the influence of a specific instance or mislead the model to disregard certain information. This approach is mostly known as a malicious way to modify the data before it reaches the training stage. In that case, it does not seem useful in MU, but due to how *Data Poisoning* works and how it influences the training of the model, we might be able to transfer this weight manipulation to unlearning.

Differential Privacy: [31] Differential privacy focuses on protecting individual data points from being inferred by limiting the sensitivity of the model to any single record. Techniques in this category introduce noise to the learning process or data itself, ensuring that the contribution of a specific data point cannot be pinpointed.

While often discussed in MU, the Pre-Training approaches do not seem like unlearning for us. They seem more like trying to remove the need for MU for data privacy reasons. When it comes to other challenges like model maintenance, corrections or fairness, they seem less applicable.

Exact Unlearning: [32] Exact unlearning refers to approaches that guarantee the removal of specific data points' influence from the model. Methods like **SISA** (Sharded, Isolated, Sliced, and Aggregated proposed in [32]), leverage modular training pipelines to localize and remove the effects of individual data points. These methods involve re-training parts of the model or maintaining modularized datasets to isolate the impact of removed data.

Regularization: [33] Regularization techniques aim to adjust the loss function or model optimization process to inherently limit the influence of specific data. By adding penalties like L1, L2, or coded regularization as in [33], these methods encourage sparsity or other constraints that encourage Underfitting, working against the fact that overfitting makes it easier to extract data samples, shown in [29].

We also like to refer to Intra-Training approaches as "Unlearning-by-Design", since the guarantee for unlearning lies in the design of the models themselves. *Exact Unlearning*, since they are basically efficiently retrained instances of the original model, serve a great role as baselines for approximative approaches.

Example-based: Example-based unlearning focuses on analyzing the influence of individual data points or sets on the model. Algorithms like **Bad Teacher** from [34] and from

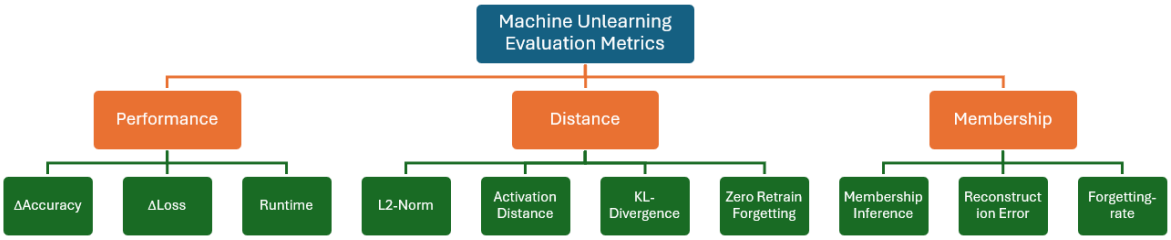

Figure 5: Classification of metrics used in MU

FEMU [2] adjust the contributions of specific examples on already trained models. Gradient Ascent is a prominent example of gradient-based machine learning algorithms, where adjustments are made to counteract the gradients caused by particular data points.

Knowledge-based: Each algorithm has its constraints and trade-offs, such as computational cost, accuracy, or compatibility with different model architectures. Knowledge-based unlearning revolves around removing knowledge or learned patterns from the model. These approaches try to identify and isolate the parts of the model that store undesirable information. The difference to *Example-based* is that the removal is concept-driven rather than tied to specific data points.

In our opinion, only Post-Training should be considered MU, since in this category the starting point is a trained model, which implicitly contains undesirable information. When Machine Learning is the path from an untrained model to a trained model, MU should be the reverse, from a trained model $f_{trained}$ to an untrained model $f_{untrained}$.

$$f_{untrained} \underset{MU}{\overset{ML}{\rightleftharpoons}} f_{trained}$$

In our empirical study below, we will dive a bit deeper into the field of Example-based MU, using an adaption of FEMU from [2] as an example.

3.3 Unlearning Metrics

Metrics are critical for evaluating the effectiveness of unlearning methods. They assess whether the target information has been successfully removed and quantify the impact on the model.

In figure 5 we show a categorisation for the types of metrics we describe below. Like above, we did not reach for completeness when it came to listing all the metrics in the field of MU but focused on the ones, which seemed most noteworthy for us.

ΔAccuracy: [35] [32] Since we want to make sure that an unlearned model still offers the same performance, the difference in accuracy between the baseline and the unlearned offers a

simple to interpret option. On the other hand, instead of examining the level of performance, we may look at the accuracy as a success indicator for removal.

ΔLoss: Has a similar usage like ΔAccuracy, but it less popular. This is most likely due to it being less interpretable and comparable since it scale of the loss changes from task to task.

Runtime: [36] [2] Runtime measures the computational efficiency of an unlearning algorithm, specifically how long it takes to remove the impact of specific data points. Faster runtime is generally desirable, especially when frequent unlearning requests are expected. However, improving runtime can involve trade-offs with accuracy or the degree of unlearning precision. We might say, that the unlearning always needs to be faster than retraining runtime, but that is met with certain conditions, most notably that the training data is still available. This is especially true for large-scale tasks, like LLM.

L2-Norm: [37] The L2-Norm evaluates the distance between the parameters of the original model and the unlearned model. A large L2-Norm indicates that the model has undergone big changes, suggesting overshooting the goal of unlearning. This metric is particularly useful when assessing algorithms that aim to retain most of the model's functionality while erasing specific data influences. However, it may not fully capture the nuanced impact of the removed data.

Activation Distance: [35] Activation Distance measures the difference in activations between the original and unlearned models for specific inputs. It is a bit more of a sophisticated approach than the *L2-Norm*. This metric provides insights into how the internal representation of the data changes due to unlearning. It is particularly relevant for understanding the effects of unlearning on intermediate model layers, making it valuable for debugging, fine-tuning and measuring if the unlearning algorithms have affected all layers.

Kullback-Leibler-Divergence: [38] Kullback-Leibler (KL) Divergence assesses the divergence between the predicted probability distributions of the original and unlearned models, or the baseline model and the unlearned model. This metric is often used in probabilistic models to quantify the extent to which the unlearned model deviates from its original counterpart, or how close it is to match the behaviour of the baseline. A lower KL Divergence suggests that the model has retained its general behaviour while removing the influence of specific data points, as well as a very low KL Divergence to the baseline suggests that the unlearned model has almost perfectly approximated its behaviour.

Zero Retrain Forgetting (ZRF): [34] The metric compares the prediction of the unlearned model for the forgotten samples to the prediction of an *Incompetent Teacher* (or an untrained/random initialized model), which, for the architecture described in [34], represents the ideal behaviour for "not-knowing" a sample. The irrelevance of a retrained model for evaluation makes this metric desirable, but the interpretability is challenging.

Membership Inference: [39] [40] Membership Inference measures the likelihood that a specific data point was part of the unlearned model's training dataset. This metric evaluates how well the unlearning process reduces the risk of privacy leakage. A successful unlearning process should result in a model where the membership of a data point cannot be confidently inferred. This metric is highly relevant in privacy-sensitive applications.

Reconstruction Error: [41] Reconstruction Error quantifies the difference between the original input and its reconstruction based on the unlearned model. It is particularly relevant in generative models, where unlearning aims to ensure that the model no longer retains identifiable features of the removed data. A higher reconstruction error indicates successful unlearning, as the model struggles to reconstruct the removed data accurately.

We like to compare the relationship between *Membership Inference* and *Reconstruction Error* to the one between *Accuracy* and *Loss*. One measure, **whether** the sample is there (classifiable) or not, while the other measures **how** good it is on classifying it.

Forgetting-Rate: [42] The Forgetting Rate (FR) evaluates how effectively an unlearning method transforms samples from "member" to "non-member" status, as determined by a *Membership Inference* oracle. It is defined as: $FR = \frac{AF-BF}{BT}$ Where AF and BF are the counts of non-member predictions after and before unlearning, respectively, and BT is the initial count of member predictions. An FR of 100% indicates complete unlearning, while 0% implies no unlearning occurred.

This metric provides a practical and comparative measure of unlearning similar to *Reconstruction Error* and is a more sophisticated metric than *Membership Inference*.

Notable Benchmarks A very relevant task in the Knowledge-based MU field is to unlearn/remove misinformation from LLMs. For this task, there are two major benchmarks.

- **TOFU [43]:** *A Task of Fictitious Unlearning for LLMs* The Task of Fictitious Unlearning (TOFU) is a new benchmark designed to help us understand how well machine learning models can "forget" specific pieces of training data. Instead of focusing on for-

getting labels, TOFU focuses on erasing information about specific individuals in the data. Key points are:

Dataset TOFU includes 200 made-up author profiles, each with 20 question-answer pairs. A smaller subset of these profiles called the "forget set," is used as the target for unlearning.

Goal Models fine-tuned on this dataset need to forget 1%, 5%, or 10% of the training data (2, 10, or 20 profiles) while still remembering the rest.

Rules The task has a strict rule, which is no retraining from scratch. Instead, methods must be efficient and only use as much computation as the number of profiles being unlearned (O(n)).

Importance Tests with current unlearning methods show that they struggle to meet these goals, highlighting the need for better approaches to unlearning.

- **WMDP [44]:** *Weapons of Math Destruction Proxy*; inspired by the book of the similar name [10], it is a dataset created to test how well large language models (LLMs) handle and forget hazardous knowledge. It focuses on three critical areas: biosecurity, cybersecurity, and chemical security. Key points are:

Dataset WMDP contains 3,668 multiple-choice questions designed by experts. These questions don't include directly harmful information but instead focus on related or precursor knowledge that could aid in malicious activities.

Purpose It serves two roles: Firstly, a tool to evaluate how much hazardous knowledge LLMs have, and secondly a benchmark to test unlearning methods designed to remove this knowledge.

Importance The benchmark fills a gap in public evaluations of LLM risks, as current evaluations are limited and private. WMDP allows open, transparent progress in assessing and mitigating risks from LLM misuse.

These benchmarks provide standardized frameworks for evaluating unlearning algorithms for knowledge-based MU algorithms and show how much space for improvement is still left in the field of MU

3.4 Unlearning Challenges and Constraints

The Unlearning Hardness Problem One of the central challenges in MU is the *"Unlearning Hardness Problem"*, which comes from the varying influence of data points. Influential data points significantly affect the model's decision boundary, while non-influential ones may have a negligible impact. This is shown in figure 6 originating from [28].

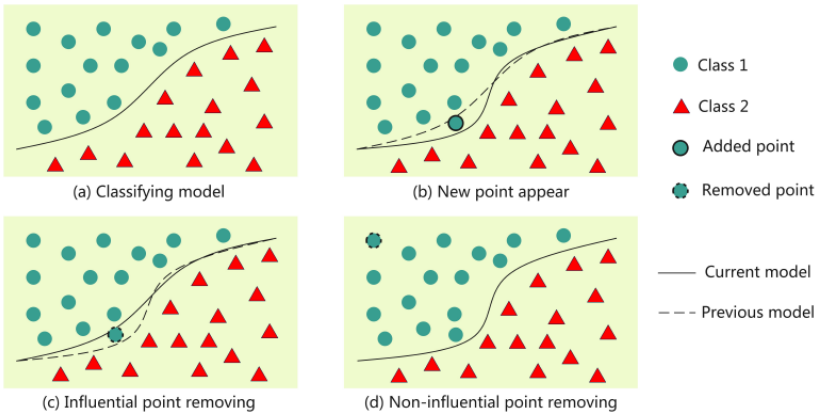

Figure 6: The model changes when adding a new point or removing a point. (a) A normally trained classifying model classifies classes 1 and 2. (b) When a new point appears, the model is trained based on it, and the classifying line is pushed to classify it. (c) When we need to remove an influential point, we should recover the contribution of this data point on the model. (d) When we remove a Non-influential point, the model may not need to change a lot.[28]

Data Usage Different scenarios arise regarding the availability of data when implementing machine unlearning. Often, the data to be removed is no longer available after the initial training phase. This introduces challenges in implementing unlearning methods that rely on accessing the original dataset. Is it possible to unlearn models just by using the samples, which we want to forget?

Without the data, it becomes difficult to reverse its contributions to the model. Methods such as FEMU from [2] aim to address this by ensuring that unlearning can be performed without retaining or revisiting the original data, naming their constraint *Zero Glance Privacy*. While the samples aren't needed in the unlearning, FEMU still needs alternative information, which would be the label and a subset of the training data.

Verifiability Verifying that a data point has been effectively unlearned from a machine learning model is a challenge by itself. This verification must provide guarantees that the model no longer retains any influence or information derived from the removed data.

Verifying the removal also goes hand in hand with the *Unlearning Hardness Problem*. What if the removed data point is in the midst of many other similar data points, it might be more difficult to check if it has been successfully removed than an outlier data point.

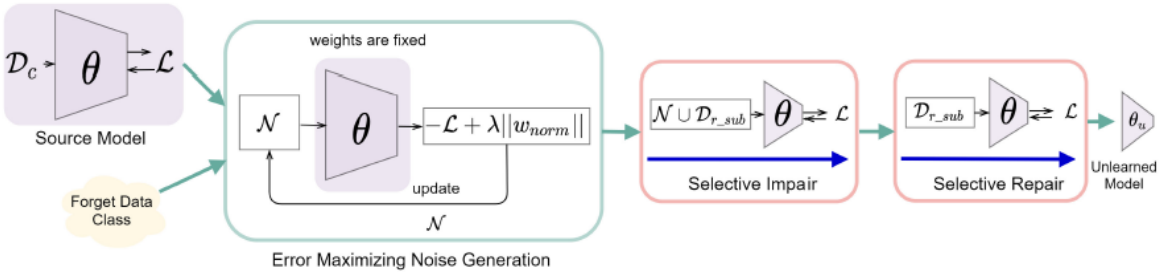

Figure 7: 3 Phase Structure of FEMU

4 Introduction to FEMU

Fast Yet Effective Machine Unlearning (FEMU) provides a scalable and efficient algorithm for selectively removing specific classes from a trained machine learning model. This section discusses the foundational principles of FEMU, its pipeline, and proposed improvements.

Motivation FEMU aligns closely with the objectives of this thesis, offering a gradient-based unlearning in a classification setting. Notably, its adaptability and the availability of source code make it my foundation for further exploration. Its authors discussed how the algorithms should be easily adjusted to different tasks, networks and datasets, which makes it great as a starting point for what we have planned for the experiments below.

Overview The goal of FEMU is to unlearn single or multiple classes without revisiting the complete training dataset. The method leverages an error-maximizing noise generation strategy combined with impair and repair steps to ensure MU while preserving overall model performance and generalizing to different neural networks[2].

4.1 FEMU Unlearning Pipeline

The FEMU algorithm consists of three primary steps: Error Maximizing Noise Generation, Selective Impairment, and Selective Repair, seen in Figure 7. The following subsections detail these steps.

The FEMU algorithm consists of three primary steps: Error Maximizing Noise Generation, Selective Impairment, and Selective Repair. Noise, in this context, refers to artificially generated data patterns designed to "untrain" the model into forgetting specific classes by feeding them into the model. These patterns are crafted to maximize the model's error on the targeted class, effectively creating a sort of anti-pattern to the classes the model was previously trained on classifying.

Error Maximizing Noise Generation The algorithm begins by generating an error-maximizing noise matrix, $\mathcal{N}$. This matrix creates patterns that counteract the target class(es) by maximizing the model loss on the forget set, D_c. Regularization ensures that $\mathcal{N}$ does not dominate other features, enabling scalable and robust noise generation.

$$\arg\min_{\mathcal{N}} \mathbb{E}_\theta[-\mathcal{L}(f(\mathcal{N}), y) + \lambda ||w_{\text{norm}}||] \tag{1}$$

Selective Impairment In this step, $\mathcal{N}$ is combined with a subset of retained data, D_r, and used to impair the model. A high learning rate facilitates sharp unlearning of the target class by modifying relevant weights.

Selective Repair Finally, the repair step restores the model's generalization ability by training on D_r alone. This stabilizes weights and ensures the model retains knowledge of the non-forgotten classes, effectively lifting the overall lost performance.

Just using a standard random noise, which is not shaped into an anti-pattern, would not be enough for unlearning the data. In the worst case, the model would just learn to identify the noise as one of the unlearned classes, still being able to correctly classify most of the class samples which we want to forget.

Data Requirements FEMU does not require all the original data used for training the model. The following is necessary:

- **Retain Subset (D_r):** A subset of the original dataset that excludes any data from the target class(es) to be unlearned. This subset should represent the remaining data distribution.
- **Forget Class Data (D_c):** Data from the class(es) to be unlearned shouldn't be available during the unlearning process. Since the noise is trained on the class label, we only need to know which label to unlearn.

Zero-Glance Privacy Policy The zero-glance privacy setting states that once a data deletion request is made, the model cannot access the corresponding data (D_c) even for unlearning purposes due to data protection regulations (GDPR) or the data is simply not available anymore. This strict policy prioritizes privacy by preventing any dependency on the target data during weight updates.

4.2 Discussion

As mentioned in the paper [2], they talk about a few ideas, which they weren't able to try out, two of which we'd like to continue, which will take us closer to GeFeU

Firstly, **noise generation process**: Think that the training of a noise batch of fixed size seems static, we'd like to replace it with a noise generator, which is trained on the labels of the forgotten class to create an according anti-pattern.

Secondly, **unlearning random subsets**: While the unlearning of random subsets would hurt the *Zero-Glance Privacy*, it points in the same direction as trying to unlearn a class subset. Instead of random subsets, we want to just subsets joined by a certain visual feature.

Additionally, we'd like to include the usage of the *KL-Divergence* as a metric, comparing the unlearned models to the baseline model, which was never trained on the forgotten data to begin with.

Assumptions By using a generator we expect to introduce more variance to the noise generation process, making it less prone to creating artefacts/being stuck in the same patterns.

On top of that, using a generator has a bigger potential than a fixed-size batch. It could theoretically produce unlimited and flexible amounts of noisy samples, which could make the tuning for the Impair and Repair phase more flexible.

Implementation Changes for FEMU+Gen [2] trained 2 batches in their example, one for each class to be unlearned. We will do the same with the noise generators, creating one per class.

$$\arg\min_{g} \mathbb{E}_{\theta}[-\mathcal{L}(f(g(\mathcal{N})), y) + \lambda ||w_{\text{norm}}||] \tag{2}$$

The Size of the batches was *BatchSize* × *Samples Dimesions* which in the case of the used dataset (CIFAR-10) was $256 \times 3 \times 32 \times 32$ with a batch size of 256. The noise generator on the other hand does not produce a batch, but only one sample, which means it has an output size of $3 \times 32 \times 32$.

When it comes to the structure of the noise batch, it is straightforward, since it is just a trainable 4-dimensional vector. The generator starts with a random vector, the starting noise. Then it consists of multiple hidden layers, consisting of 1024 neurons, and each layer with a *ReLU* activation function, until resulting in the output layer of the above dimensionalities.

4.3 Comparison

The subset unlearning will be tackled further down in the next section. Firstly we'd like to compare the paper approach of using a noise batch to training a noise generator which can

craft us noisy samples suited for the task at hand.

Methodology To facilitate the comparison, we will use three different model types:

1. **Baseline model** (f_r): This model is trained only on D_r.
2. **FEMU model** (f_{femu}): This model follows the unlearning approach proposed by [2] in their experimental setup, with *torch.manual_seed* set to 100.
3. **FEMU+Gen model** (f_{femu+}): This model employs our approach.

The unlearning task is restricted to removing classes "0" and "2". For the baseline model, samples from classes "0" and "2" were excluded during training, making it the ideal reference for evaluating the behaviour of the unlearned models.

To optimize performance, the following parameters were fine-tuned on the training data using *Optuna* with 200 trials. The objective was to minimize the KL divergence between the unlearned model (FEMU+Gen) and the baseline model. The best trial achieved a KL divergence of 2.6192. For deployment, we will use its rounded values, shown in Table 1, for the hyperparameters.

Hyperparameters	**Values**
Number of Epochs for the Noise Maximization Phase	10
Number of Batches Trained per Epoch	15
Learning Rate for the Noise Maximization Phase	0.18
Batch Size for the Noise Maximization Phase and the Impair Phase	370
Number of Noise Batches used in the Impair Phase (Number samples from D_r for that phase is fixed at 1000 per class not in D_c)	1
Regularization Term for the Noise Maximization Phase	0.15
Number of Hidden Layers for the Noise Generator	6
Dimensions for the starting Noise in the Generator	70

Table 1: Hyperparameters Optimize for FEMU+Gen

After fine-tuning the listed hyperparameters in 1, we unlearned the CIFAR-10 trained ResNet-18 model f_{all}, which was trained the same way as in [2], 30 times, resulting in 30

Unlearning Algorithm	D^{train}	D^{val}	D_r^{train}	D_r^{val}	D_f^{train}	D_f^{val}
FEMU	3.00	2.40	2.96	2.25	3.19	3.20
FEMU+Gen	3.00	2.41	2.99	2.32	3.06	3.07

Table 2: We measured the KL-Divergence between baseline and either the model unlearned via FEMU or FEMU+Gen

different f_{femu+}^i models. The purpose of training multiple instances is to analyze the variance of FEMU+Gen. Like f_{femu}, f_{femu+}^i have the same origin model, meaning they all unlearned/stem f_{all}.

Once all the models are prepared, we will compare FEMU and FEMU+Gen to the baseline model using the *KL-Divergence* metric across the following subsets of D:

- All training data (D^{train})
- All validation data (D^{val})
- Retained samples of the training data (D_r^{train})
- Retained samples of the validation data (D_r^{val})
- Forgotten samples of the training data (D_f^{train})
- Forgotten samples of the validation data (D_f^{val})

In addition to *KL-Divergence*, we will evaluate accuracy per class, plotting all three model types (f_r, f_{femu}, $f_{FEMU+Gen}$) side by side for comparison.

Results The results for the first comparison can be seen in Table 2, where FEMU has better scores on most of the datasets. However, the difference to FEMU+Gen is not significant. It is worth pointing out that FEMU+Gen offers, in comparison to the other scores, a better score on D_f, training and validation.

Referring to the accuracies shown in Figure 8, all models achieve 0% accuracy on D_c. Moving to the retained classes, the scores vary across different classes. In some cases, FEMU is closer to the desired behaviour, while in others, FEMU+Gen performs better. Overall, the two approaches show a balanced relationship. The total $\Delta Accuracy$ is 7.5% for FEMU and 7.6% for FEMU+Gen.

As noted earlier, we trained 30 instances of f_{femu+}^i to analyze variance, which was found to be minimal. The scores for KL-Divergence in Table 2 and the accuracies in Figure 8 represent the averages across all f_{femu+}^i. Variations in metric scores were typically only observed in the third decimal place.

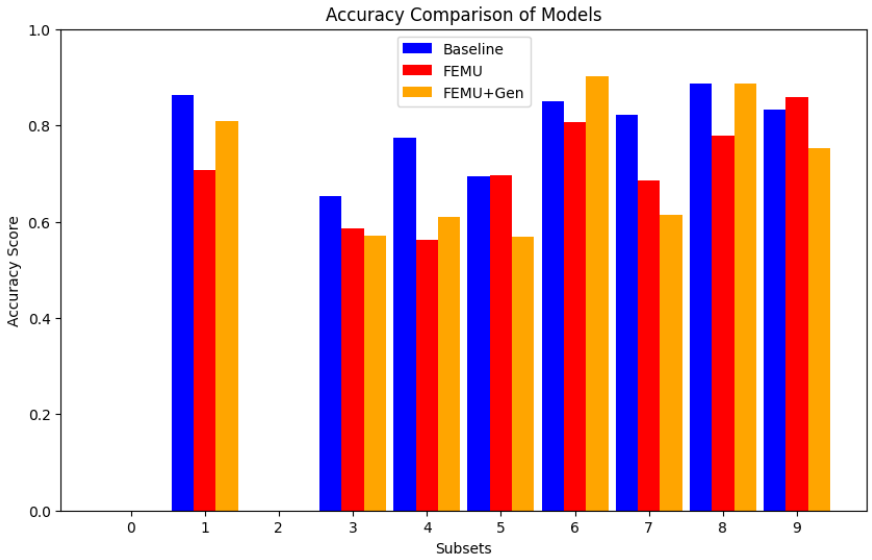

Figure 8: Accuracy Performance on Validation Data per Model

In terms of time and complexity, it is worth mentioning that FEMU offers a less resource-intensive approach, as it avoids the need for a computationally expensive generator.

4.4 Conclusion

The comparison between FEMU and FEMU+Gen highlights key trade-offs in their approaches. While FEMU offers a simpler, resource-efficient method with stable performance, FEMU+Gen introduces a noise generator that adds flexibility and potential for improvement.

The generator-based approach (FEMU+Gen) showed promise. Maybe with a more sophisticated architecture, the performance of the algorithm may be further improved. However, the increased computational cost and time required for training the generator must be considered. Despite these challenges, the generator introduces additional variance and adaptability, potentially enabling more effective tuning for the impair and repair phases. Especially its better performance on forgotten samples (D_f), seems to indicate that, while not major, the added variance of the generator made good anti-pattern.

The evaluation also showed that both methods achieve comparable accuracy and *KL-Divergence* on most metrics, with FEMU+Gen demonstrating minimal variance across 30 instances. However, the simplicity of the batch-based approach in FEMU makes it a more practical choice in scenarios where resources are constrained or training time is critical. On the other hand, we are fixed to the batch of say 265 samples, while a noise generator potentially offers infinite different samples to unlearn with.

Looking ahead, whether to continue with a generator-based approach or revert to the batch-based strategy may depend on the specific application requirements. If flexibility and

adaptability are key priorities, FEMU+Gen provides a solid foundation with room for future improvements. On the other hand, for faster deployment or cases with limited computational resources, FEMU remains a reliable option.

Nevertheless, this experiment is meaningful in the grand scheme of things. Further investigation is needed to fully establish these approaches, particularly for large-scale models. Additionally, the balance between the three phases remains a point of discussion. Maybe the need for a Repair phase will be obsolete.

Still, the overhead for the MU is non-existent since we didn't need any information about the training of the to-be-unlearned model and expected that certain samples were used (which is the basis for the need to unlearn). The necessary things needed are the model samples, while still not ideal, making the approach flexible. Also, while calculation costs are worth noting, going forward they will play less of a role for MU, since the biggest constraint will be the absence of data, not the time it takes to train the models.

Because of its flexibility and the potential of the generator, we'll keep building on FEMU+Gen further down in this thesis.

5 Feature Unlearning

In section 4.2 we talk about two ideas for FEMU, noise generation process and unlearning random subsets. This section will focus on the second one, the unlearning of subsets.

5.1 Introduction to GeFeU

GeFeU is our proposed algorithm for feature unlearning. It stems from FEMU in its core, with the key difference of its goal. While FEMU designated use is to unlearn single or multiple classes (in one run), we narrow it down to one subset of one class, continuing on the ideas noted in the conclusion of [2]. Of course, you could apply GeFeU multiple times sequentially, but we did not investigate this usage and its possible effects on the unlearned model.

We want to mention what we understand as a subset. When looking at the samples of a certain class in a dataset, we are usually able to form subclasses based on certain observations. For example, the sevens class of MNIST could be divided into the ones with and the ones without a middle line. Similarly, we could do the same with another dataset, e.g. FashionMNIST for which the samples of clothing category may be divided into sandals or shoes with or without heels. As these descriptions are linked to particular data features, we call it **Feature Unlearning**.

5.2 Assumptions

For the first phase of FEMU, a noise trained on maximizing the loss for the selected class is used. During the training, the loss is calculated by comparing the model's prediction to the class's label, which is represented as a single-dimensional one-hot-encoded vector. This makes sense for the task of unlearning entire classes, but regarding the goal of unlearning a subset of the class, it might cause the loss of the retained class data to increase as well.

Hypothesis: For the task of selective data removal in machine learning models, utilizing the model's prior predictions as labels to train noise for unlearning is more effective than using one-hot encoded class vectors. This approach leverages the model's prediction as an alternative representation of the specific sample to be unlearned. Transitioning from unlearning entire classes to individual samples, the prediction-based method provides a more precise and contextually relevant target for noise generation, facilitating more accurate and efficient unlearning.

5.3 Realization

To approach this differently, we decided not to use a one-hot encoded vector. Instead, GeFeU leverages the predictions of the model that we aim to unlearn. These predictions, obtained using the Softmax function, are probability vectors that match the shape of the original labels. For each data point x_f in the subset D_f (the portion of the training dataset D we want to forget), a new label y_f is generated. This introduces the following modification to the algorithm:

$$\arg\min_g \mathbb{E}_\theta[-\mathcal{L}(f(g(\mathcal{N})), f(x)) + \lambda||w_{\text{norm}}||] \quad \forall \quad x \in D_f \tag{3}$$

Here, the parameters θ of the generator g are optimized based on the expected negative loss $-\mathcal{L}$. The model's prior predictions $f(x)$ for each sample x in the subset D_f, which we want to forget, serve as labels. We keep the Regularization term $\lambda||w_{\text{norm}}||]$ for θ to make sure no exploding gradients occur.

We hypothesize that the model's predicted class distribution for a data point serves as an abstract representation of that point. This suggests that to effectively remove the influence of a data point, it is essential to understand how the model interprets and classifies it. By focusing on these predictions, we can better guide the unlearning process.

However, this approach goes against the *Zero-Glancy Privacy Assumption* proposed by [2]. This is because we still require access to the samples to obtain their output distributions, which are used to train the noise generator. That said, it is generally more feasible to acquire

the specific samples we need to unlearn, compared to retaining access to the entire original training dataset.

6 Methodology

Having set the groundwork for the experiments, it's time to show empirical proof to our assumptions. Focusing on GeFeU against FEMU+Gen, we aim to highlight the improved precision when using model prediction as the label for the *Error Maximizing Noise Generation*.

6.1 Research Design

In these experiments, we aim to highlight the improved precision of GeFeU. As a comparison, we look at the behaviour of the following models: an untrained f_u, a trained, a retrained (serving as the baseline), and models unlearned using GeFeU and FEMU+Gen. Additionally, we include a *Simple Gradient Ascent* approach to the MU algorithms to demonstrate the most basic idea of reverting the influence of specific samples.

The *Trained* and *Retrained* models represent the starting point and the desired endpoint, respectively. The *Unlearned* model is included to examine whether the models are "over-unlearned," regressing too far in their behaviour. If the models are too similar to this type, it may indicate that the MU process has overstepped and removed too much influence during unlearning.

For each of the above-listed model types, 30 instances will be created. In particular, for the MU algorithms, we aim to analyze their variance in performance.

To evaluate their adaptability to different use cases, we will include three different datasets, testing the robustness and transferability of the MU algorithms. Also, the MU algorithms will only be fine-tuned on one of the chosen datasets.

6.2 Used Data and Models

For our experiments, we selected three datasets: MNIST, Colored MNIST, and FashionMNIST. For each dataset, we chose a specific class and divided it into subclasses to create a subset whose influence we aimed to remove after training.

1. MNIST: The class "7" was divided into two subclasses: "7 with a middle line" and "7 without a middle line". Here "7"s without a middle line are unlearned!

2. Colored MNIST: Similar to MNIST, the class "7" was also divided into the same two subclasses. Here "7"s with a middle line are unlearned!

3. FashionMNIST: The class "5" (sandals) was split into "sandals with heels" and "sandals without heels." Here sandals with a heel are unlearned!

The splitting process was done manually, which may have introduced some inconsistencies. However, we accepted this risk for the purpose of our study. The splits used in our experiments are openly available in the reference repository on GitHub.

When selecting models, we sought inspiration and a reliable baseline, as our goal was not to develop the best model for each classification task but rather to focus on the unlearning process. For MNIST, we referred to the comprehensive study by [45], which explored architectures achieving the best results. However, we opted against using a CNN and instead chose a simpler MLP—specifically, a 2-layer perceptron with a *ReLU* activation function. This decision was made to start with a basic and straightforward model.

For Colored MNIST, we employed a more sophisticated CNN to evaluate the performance of the algorithms on this dataset. For FashionMNIST the same model as for MNIST was used, as the data share its dimensionalities/size with MNIST. The idea was to test the algorithms on different datasets while keeping the model architecture consistent.

The implementation details of all models can be found in the reference GitHub repository.

6.3 Used Metrics

We will use the following metrics to evaluate the different model types:

- **Accuracy**: A standard metric for classification. Since the goal is always to preserve performance, significant increases or decreases are unwanted. Additionally, accuracy can serve as a distance measure to the *Retrained* model, as it should closely match its performance.
- **KL-Divergence**: A standard and widely used metric for quantifying the difference between probability distributions.
- **Loss**: While accuracy is a key metric, we also monitor the loss. If the algorithms are stable, the loss should remain within acceptable bounds.
- **L2-Norm**: This metric examines how much the unlearning models deviate from the *Trained* model from which they originated.
- **Layer-wise Distance**: Similar to the L2-Norm, but this metric provides a deeper analysis by identifying whether the changes are distributed across layers or localized to specific layers.

Our primary focus will be on *Accuracy* and *KL-Divergence*. Accuracy is emphasized for its interpretability in assessing the model's performance, while KL-Divergence is highlighted for its ability to quantify the similarity in prediction behaviour.

6.4 Procedures

Training Parameters Firstly, we take a look at how the *Trained* and *Retrained* model types are trained using the Adam Optimizer. The learning rate varies depending on the task: 0.00001 for MNIST, 0.0001 for CMNIST, and 0.00002 for FashionMNIST.

We apply the StepLR learning rate scheduler to reduce the learning rate every five steps by a factor of 0.1.

We use a batch size of 8 for our model training. This smaller size was chosen to balance the trade-off between computational efficiency and training stability. The number of updates are dataset-specific, with MNIST and FashionMNIST requiring 60,000 updates and CMNIST requiring 20,000. This number makes sure all samples in the training data are at least used once, laying the bases for MU.

About FEMU+Gen Compared to its version in Section 5, the only change is in D_f. In this case, D_f is not an entire class but a subset of it. The noise remains trained on the class label, making this the only difference in the MU process design compared to GeFeU.

About *Simple Gradient Ascent* We included this approach for MU due to its simplicity, and because it is often the first intuitive idea when considering how to reverse the effects of learned samples. It serves as a baseline for unlearning, providing insight into how far the field has progressed. In this method, we perform a single training epoch on the subset D_f, but invert the direction of the optimization steps, in practice *Ascent instead of Descent*.

Unlearning Parameters For the MU algorithms, we used **Optuna** to search for the optimal hyperparameters. Around 100 trials were conducted for each GeFeU (4) and FEMU+Gen (3) to identify the best values. The specific hyperparameters that were tuned can be found in the reference tables. During the optimization process, we focused on minimizing the KL-Divergence between the *Unlearned* model and the baseline *Retrained* model using the training data. For more details, an Optuna Dashboard can be found in the repository.

We'd like to mention again, that the search for the right hyperparameters was only done for MNIST!

Hyperparameters	**Values**
Number of Epochs for the Noise Maximization Phase	7
Learning Rate for the Noise Maximization Phase	0.24
Batch Size for the Noise Maximization Phase and the Impair Phase	100
Ration between Noise Batches and Retain Data Batches	4.75
Regularization Term for the Noise Maximization Phase	0.07
Number of Hidden Layers for the Noise Generator	5
Dimensions for the starting Noise in the Generator	128
Learning Rate for Impair Phase	0.04
Learning Rate for Repair Phase	0.02

Table 3: Hyperparameters Optimize for GeFeU

Hyperparameters	**Values**
Number of Epochs for the Noise Maximization Phase	6
Learning Rate for the Noise Maximization Phase	0.3
Batch Size for the Noise Maximization Phase and the Impair Phase	32
Ration between Noise Batches and Retain Data Batches	11.5
Regularization Term for the Noise Maximization Phase	0.26
Number of Hidden Layers for the Noise Generator	1
Dimensions for the starting Noise in the Generator	35
Learning Rate for Impair Phase	0.05
Learning Rate for Repair Phase	0.01

Table 4: Hyperparameters Optimize for FEMU+Gen

While we tried to use the fine-tuning of Optuna for the Gradient Ascent approach, too, we decided against it. Reason being, that we wanted it to be a representative of a naive reverse training loop. We employed the same hyperparameters as were used in the training of each model. To be more precise, we carried over the Learning Rate and Unlearning Rate from the original training session. For example, in the case of MNIST, we utilized a Learning

Rate of 0.0001, a batch size of 8, and unlearning for 1 Epoch. This makes it the only one of the evaluated MU algorithm that changes (in comparison rather small) amount of hyperparameters.

6.5 Limitations

Some evaluations, like switching the D_f and D_r of the affected class, fine-tuning on Colored MNIST and FashionMNIST and assessing their performance, and random subset (as proposed by [2]), could not be conducted as they exceeded the scope of this Bachelor's thesis. Due to time constraints, the average losses for FashionMNIST were not evaluated.

7 Results

Regarding the results of our experiment, we will first take a look at the performance of our MU algorithms of MNIST since we fine-tuned for it. Then check how Colored MNIST and FashionMNIST worked out.

7.1 MNIST

Simple Gradient Ascent Among the models, *Simple Gradient Ascent* stands out as a significant outlier. Its accuracy remains below 40% for all classes except class "9" (Figure 10). Overall, it achieves an accuracy of just 23.91% on the retained data, D_r, and 0% on the forgotten data, D_f. While a 0% accuracy on D_f might seem desirable, this result falls far short of the baseline target performance.

In this case, *Gradient Ascent* has overshot the goal of forgetting D_f. Analyzing the Layer-wise Distance in Table 12 reveals minimal changes in the model's weights and biases compared to other algorithms. Despite these limited adjustments, the model's performance degraded significantly.

FEMU+Gen From Figure 10, it is evident that FEMU+Gen demonstrates strong performance, coming close to the baseline, although both it and GeFeU show some struggles with specific classes. Notably, for class "7", which the model aims to unlearn the subset from, FEMU+Gen achieves performance closest to the baseline (Figure 10). This trend is also reflected in Figure 9, where FEMU+Gen outperforms other MU algorithms in approaching the baseline on the D_f subset. However, this comes with a trade-off, as FEMU+Gen produces a higher than desirable loss on D_f, which can be observed in Figure 15.

When analyzing the Layer-wise Distance (Figure 12), FEMU+Gen exhibits the most significant changes, yet it manages to preserve much of the model's overall behaviour. Its

KL-Divergence further supports this from the *Trained* model, which averages 2.39, compared to 9.77 for *Simple Gradient Ascent*, shown in Figure 11.

GeFeU It achieves the best overall results among the MU algorithms. While FEMU+Gen outperforms in specific aspects—such as the desired accuracy on D_f (Figure 10)—GeFeU is more consistent and precise, particularly when considering the variance displayed in Figure 9 and 14. This also holds true for the Loss, observable in Figure 15. Although GeFeU does not perform as well on average as FEMU+Gen at forgetting the D_f subset, it provides better overall performance across other classes.

The KL-Divergence further highlights GeFeU's preciseness as a MU algorithm. With a KL-Divergence score of 2.29, GeFeU stays closer to the *Trained* model compared to FEMU+Gen. A similar trend is observed in the KL-Divergence relative to the baseline, with scores of 3.34 for GeFeU and 3.39 for FEMU+Gen. Though the difference is minor, it represents a consistent improvement. Additionally, these improvements were achieved with fewer changes to the model (seen in Figure 12) compared to FEMU+Gen.

7.2 Colored MNIST

A change in network architecture, transitioning from a two-layer perceptron to a convolutional neural network, alongside a shift in data from grayscale images to RGB images, significantly impacted the performance of all MU algorithms. None of the algorithms achieved an accuracy above 25% overall test data, as shown in Figure 9. When going into more details, FEMU+Gen has pushed the accuracy of the class affected by the unlearning, according to Figure 9, indicating that instead of decreasing it has increased the influence of all samples in D_f.

Interestingly, FEMU+Gen dramatically increased the accuracy of D_f, raising it from approximately 60% to 97.85%. In contrast, GeFeU overcompensated during the unlearning process, causing the overall performance of the model to deteriorate significantly.

Layer-wise Distance This aspect is important because the model used for this test case is larger than the previous one, whose results are shown in Figure 12. Apart from the later layers, the further we move from the output towards the input, the smaller the changes seem to become.

Average Loss The behaviour of average loss is also noteworthy. Although the algorithms are not explicitly designed to unlearn more than the specified subset, the losses for FEMU+Gen and GeFeU closely resemble those of untrained models. This can be seen in Figure 15 and is underlined be the KL-Divergence results 11.

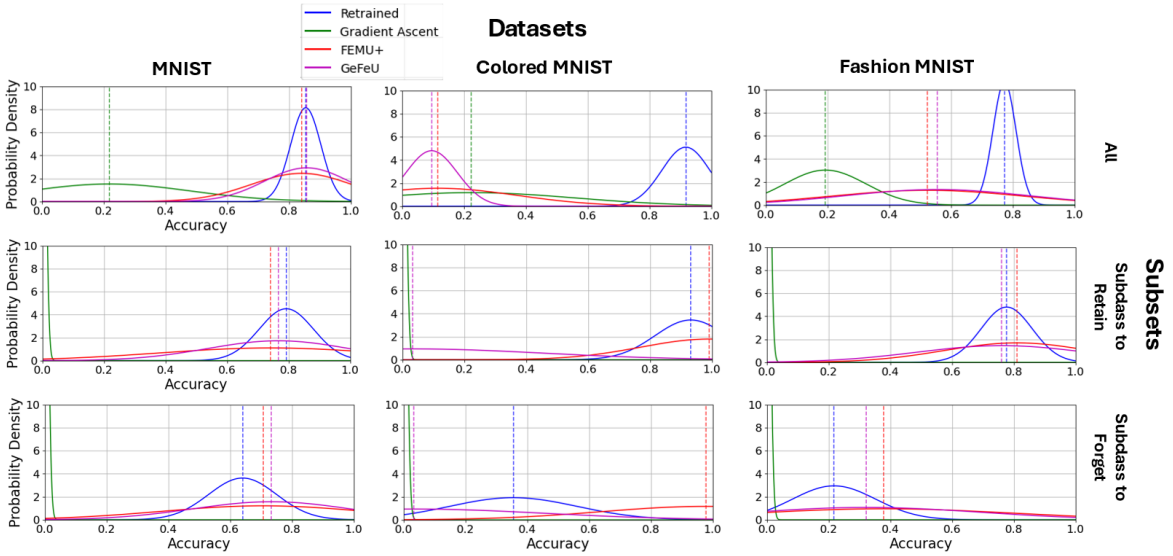

Figure 9: An overview of all test cases, where the evaluation sets are divided into three categories: the full test dataset, only the retained data of the affected class (for MNIST, this corresponds to class 7), and the forgotten data of the affected class. The more the curve of the unlearned models (green, red, pink) approximates the baseline (in blue), the better the algorithm. The variance in the distributions reflects the reliability of the (re)training or unlearning process for each model type. This shows that GeFeU is not only better at approximating the baseline but also more reliable in doing so.

7.3 FashionMNIST

Although the results are not as dramatic as those observed for Colored MNIST, the performance of the MU algorithms on the FashionMNIST task still leaves much to be desired. The accuracy values in Figure 9 suggest that GeFeU achieves the best results, achieving results closest to the baseline. However, the resulting models show significant variance and inconsistency. This issue is not limited to GeFeU.

Part III

Discussion

8 Summary

In the rapidly evolving landscape of artificial intelligence, the ability to remove specific learned information from machine learning models has become a new and interesting area of research. This process, known as **Machine Unlearning**, is particularly relevant in the context of data

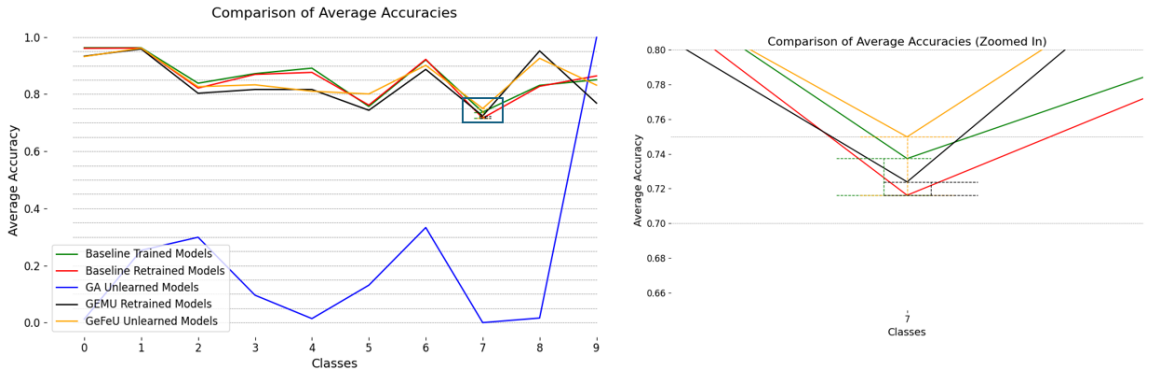

Figure 10: Comparison of the average accuracy scores across all 30 instances per model type on the MNIST test data. This visualization provides a clearer comparison of model performance across different classes, while Figure 9 emphasize the overall accuracy performance distribution and 14 offers more detail. Note that the forgotten data is still included in this evaluation.

privacy regulations such as the General Data Protection Regulation (GDPR), which grants individuals the *Right to Be Forgotten*. As Machine Learning models are increasingly used in sensitive domains, ensuring that they can "forget" certain data without requiring full retraining is both a technical challenge and is societal necessity.

This thesis investigates various approaches to machine unlearning, categorizing existing techniques based on their implementation stage: pre-training, intra-training, and post-training unlearning. Among these, post-training methods remove learned information after model deployment. The study specifically focuses on gradient-based unlearning approaches, which leverage optimization techniques to selectively erase specific data influences from a model while maintaining its overall predictive performance, by using a similar methodology for the training of a ML model.

A major contribution of this thesis is the development of a new unlearning algorithm, Generator Feature Machine Unlearning (GeFeU), which builds upon the Fast Yet Efficient Machine Unlearning (FEMU) framework. While FEMU enables class-level unlearning by introducing error-maximizing noise, GeFeU refines this approach by incorporating a noise generator and targeting unlearning at the feature level. Instead of removing entire classes from a model's memory, GeFeU enables the selective removal of subsets of data within a class based on shared visual or structural characteristics. This represents a step towards the goal of erasing single data points.

The study evaluates GeFeU's effectiveness through empirical experiments on three well-known datasets: MNIST, Colored MNIST, and FashionMNIST. The experimental design includes two model architectures, multilayer perceptrons (MLPs) and convolutional neural

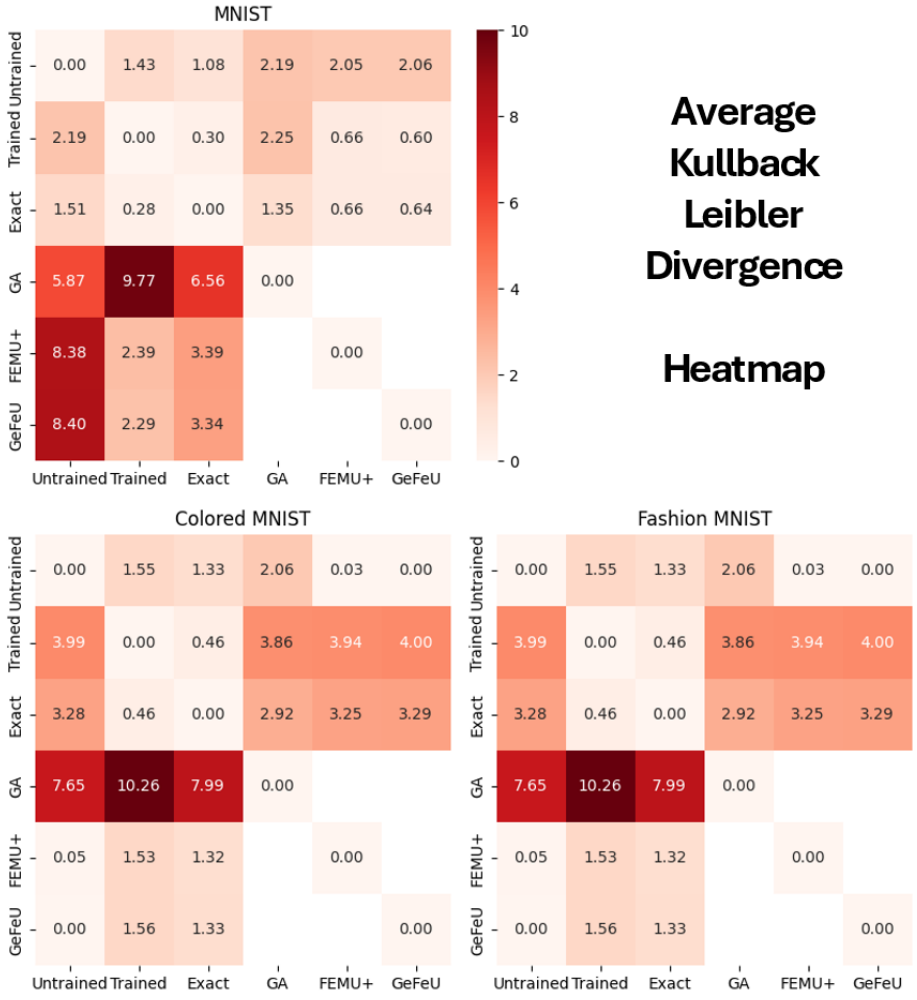

Figure 11: This figure compares the KL-Divergence of the forget data. Since KL-Divergence is not symmetrical, the matrix contains different values for each comparison pair, depending on which distribution serves as the base.

Test case: MNIST Among the MU algorithms, GeFeU exhibits the smallest distance to the baseline model, followed closely by FEMU+Gen, with a significant gap to *Simple Gradient Ascent*.

Test case: Colored MNIST The results appear unpredictable, possibly due to the lack of fine-tuning. However, it is noteworthy that *Simple Gradient Ascent*'s influence has remained relatively consistent.

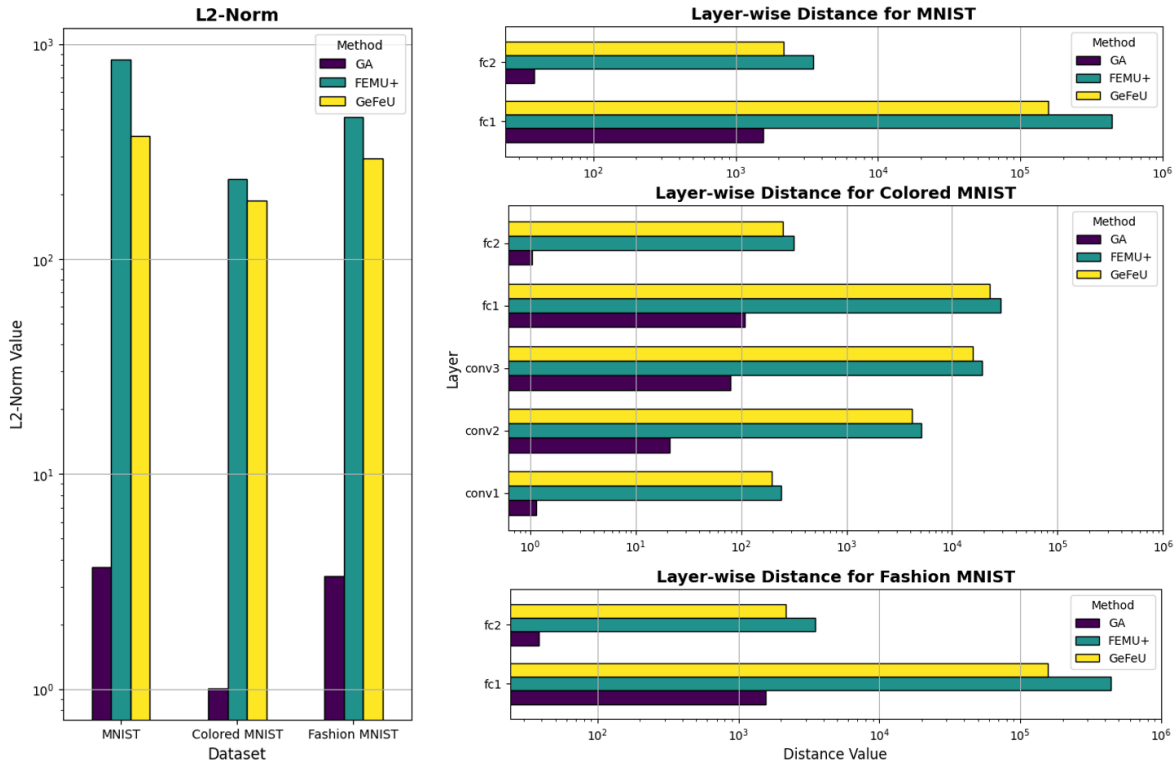

Figure 12: To assess the extent of parameter changes, we used the L2-Norm and Layer-wise Distance metrics. Interestingly, across all approaches and test cases, the graphs exhibit relatively similar behaviour.

Left: The plot shows the overall average L2-Norm between the original model (before un-learning) and the unlearned model. Notably, *Simple Gradient Ascent* causes only minor changes, whereas FEMU+Gen introduces significantly larger modifications. In comparison, GeFeU results in fewer changes, highlighting its improved precision.

Right: This plot provides a more detailed view by showing how changes are distributed across different layers of the neural network. This helps reveal how deeply the modifications propagate.

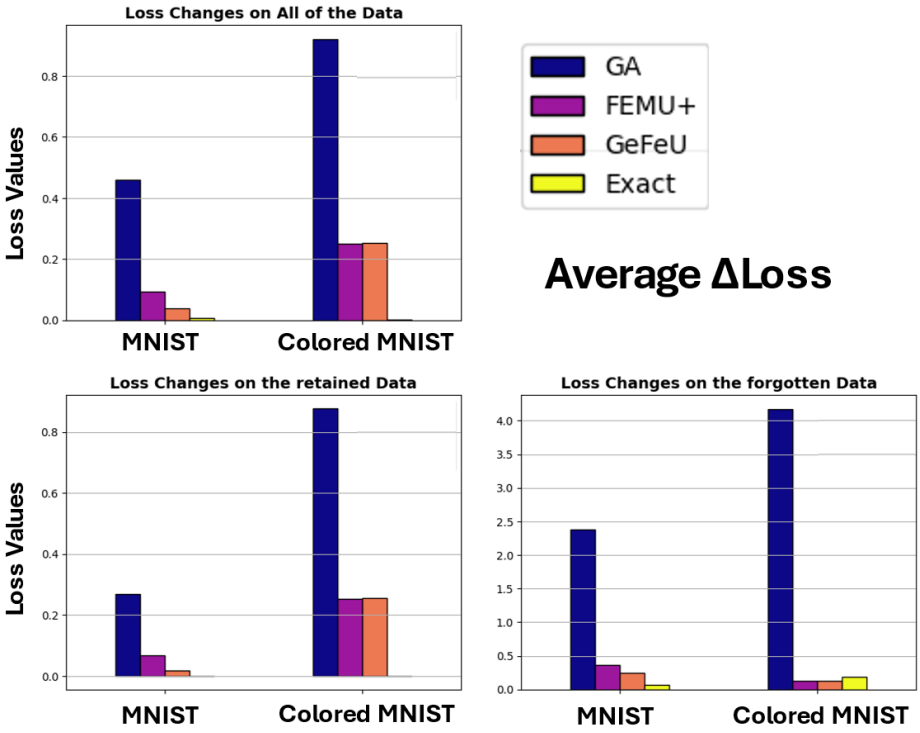

Figure 13: This figure illustrates the average loss change by comparing the loss of the original model to that of the unlearned model. These graphs should be interpreted in conjunction with the parameter changes shown in Figure 12.
Simple Gradient Ascent produces the most significant loss alterations despite making the smallest changes to the model itself.
In total, while not in every case, GeFeU comes closest to the desired amount.
(Unfortunately, FashionMNIST could not be included.)

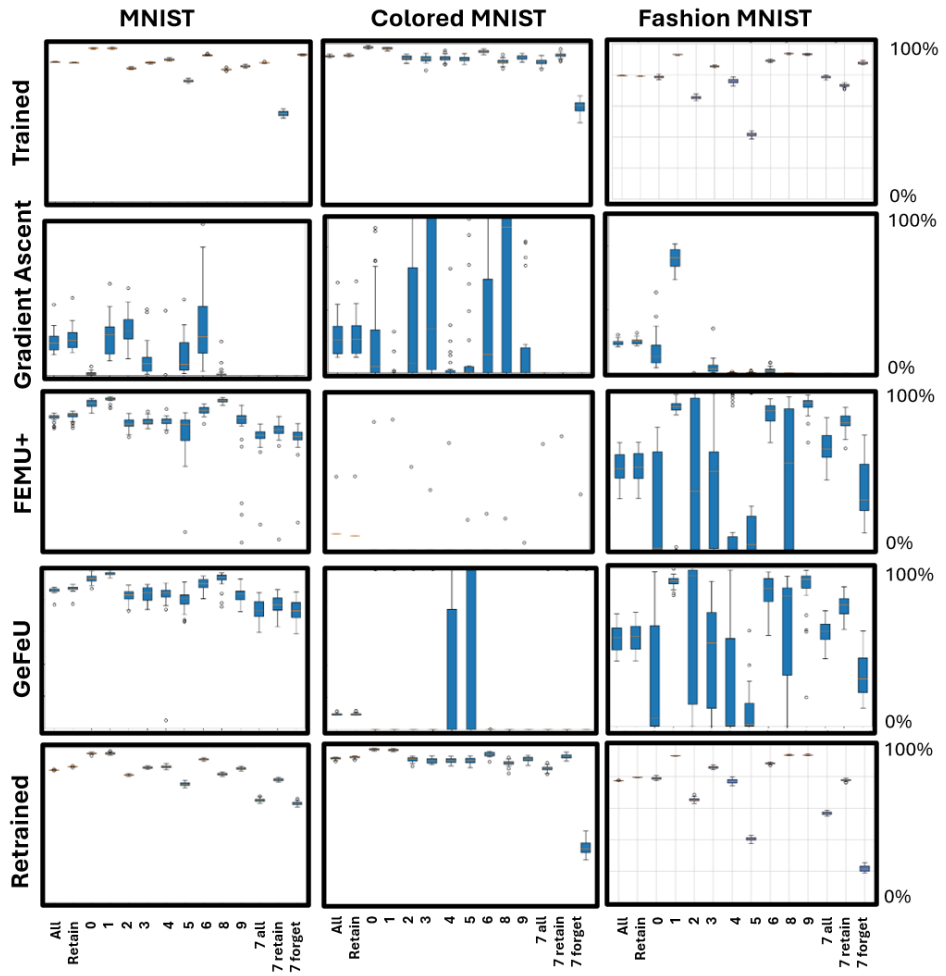

Figure 14: This figure presents the test accuracies, categorized by test case, algorithm, and data subset. Our goal was to provide a more detailed perspective on model performance, particularly with respect to different data subsets.

Test case: MNIST As expected, *Simple Gradient Ascent* negatively impacts overall model performance. In contrast, FEMU+Gen and GeFeU exhibit less impact on the overall performance, approximating the desired behaviour of the baseline retrained model. Notably, GeFeU achieves even better alignment due to its lower variance.

Test case: Colored MNIST & FashionMNIST For both test cases, unlearning with each MU algorithm was unstable.

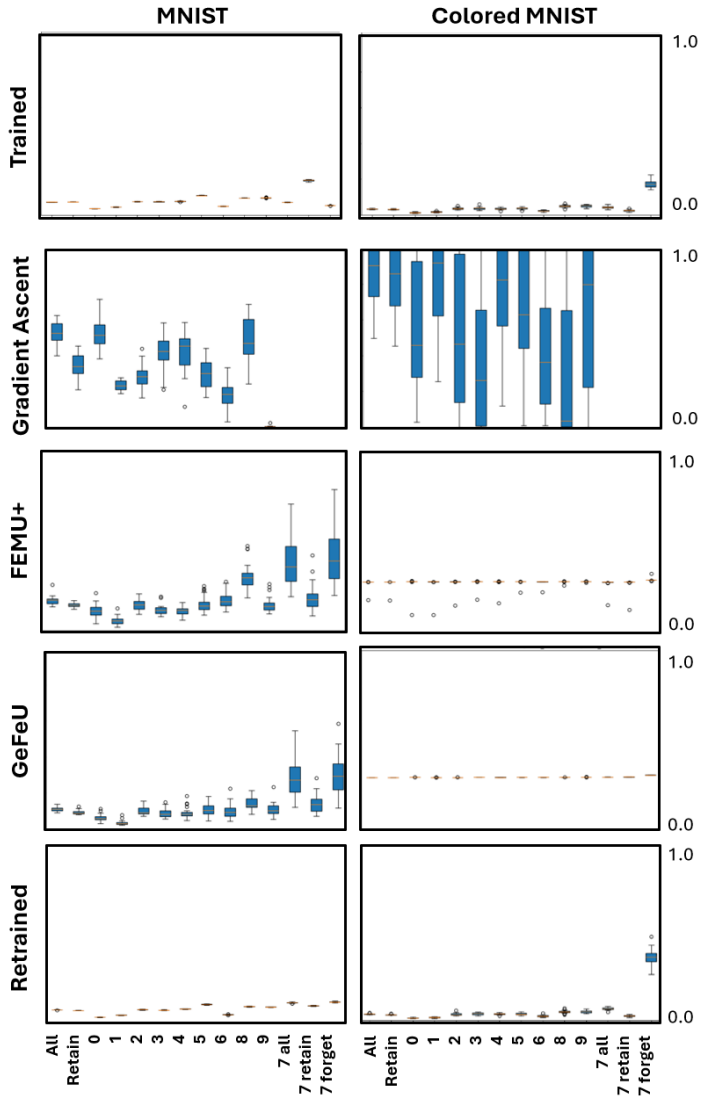

Figure 15: This figure presents an ensemble of all test losses recorded during our experiments. Unfortunately, FashionMNIST was not completed in time.

Test case: MNIST Examining the different graphs, we observe that causes a loss increase across all classes, with some values even exceeding the scale. In contrast, FEMU+Gen and GeFeU primarily affect the targeted class, which is most evident for MNIST. Additionally, GeFeU exhibits lower variance compared to FEMU+Gen, indicating improved precision.

Test case: Colored MNIST Due to the lack of fine-tuning, the results for the MU algorithms deviate significantly. Even though *Simple Gradient Ascent* introduces the fewest changes — as shown in 12 — it still produces high variance across runs. Interestingly, FEMU+Gen and GeFeU display loss behavior similar to that of an untrained model. While this outcome may be desirable, it could also indicate excessive unlearning.

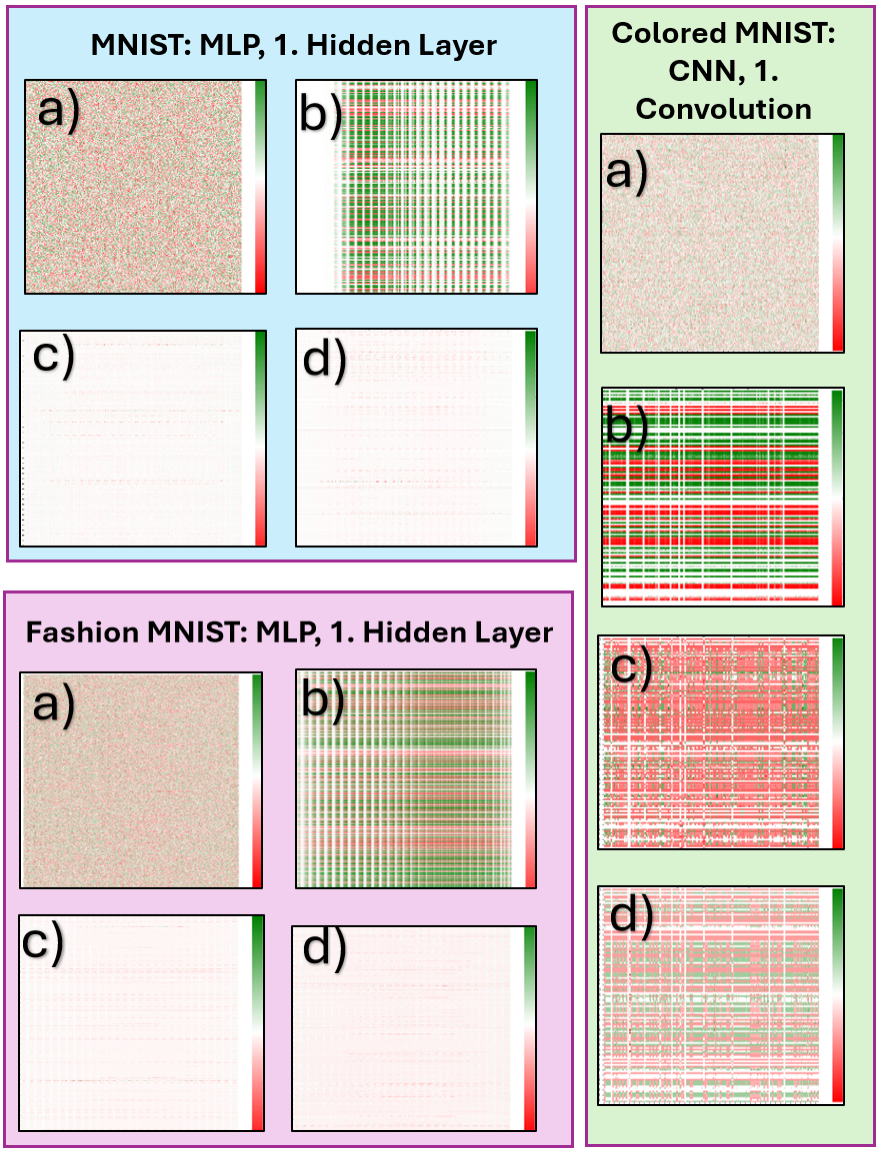

Figure 16: This figure illustrates changes in neuron activations after MU. **(a)** serves as a reference point, showing the differences between two random, unrelated models. **(b)** represents a change in model parameters using *Simple Gradient Ascent*. **(c)** corresponds to FEMU+Gen, and **(d)** to GeFeU. Our goal was to dive deeper into analysing unlearned models to determine whether targeted changes or patterns can be observed visually, which makes the scale obsolete.

networks (CNNs). The unlearning process is assessed using key metrics, including accuracy, KL-divergence, and layer-wise distance, to determine both the efficacy of unlearning, quantive the change and the extent to which the model retains generalization capabilities.

Findings indicate that GeFeU effectively removes targeted feature representations from the model while maintaining performance on retained classes. The introduction of a noise generator adds variance to the unlearning process, preventing the model from relying on static patterns and improving flexibility in data removal. While FEMU+Gen, a prior extension of FEMU, demonstrated efficiency in unlearning entire classes, GeFeU advances this approach by addressing the challenge of feature-level forgetting.

Regarding the challenges, this work depicts a dependency on fine-tuning the algorithms to their respective use cases, indicating a still missing robustness and transferability between model and data types. While offering good results during the experiments of MNIST, referring to Colored MNIST and/or FashionMNIST undesirable changes to the models.

9 Interpretation

This section is here to discuss the results. Our shared thoughts are meant to provoke new ideas to try out and discuss certain behaviour patterns, which stuck in our minds. Claims are open for discussion.

Upon reviewing our results, we find them unsatisfactory. We had hoped that at least one of the tested algorithms would produce results somewhat comparable to the baseline. However, the strong dependence on hyperparameters suggests that the evaluated MU approaches lack robustness. For example, in the case of Colored MNIST, the loss plots for FEMU+Gen and GeFeU may suggest that there was "too much unlearning" taking place. A mistake which could be fixed by reducing the amount of noise used during the *Impair Phase* or reducing the *Learning Rate* accordingly

While appears too naive in its approach to serve as an effective MU algorithm, it has the advantage of a straightforward process with minimal constraints, requiring only D_f. The minimal changes it introduces to the model, as shown in Figure 12, could be considered beneficial. Additionally, its execution remains consistent, as highlighted by the KL-Divergence values in the heatmap of Figure 11, whereas FEMU+Gen and GeFeU might be more difficult to interpret, because of their randomness induced by the noise.

Nevertheless, the results on the MNIST dataset serve as a promising starting point. We plan to continue our research in this area, primarily focusing on engineering a reverse version of the gradient-based training process in machine learning.

One way to mitigate undesirable outputs is to introduce a filter, augment the existing network with additional layers, or erase/modify individual neurons. E.g. ignoring the output of the output neuron responsible for its respective class, for the task of Class Unlearning. While these approaches provide a straightforward solution by masking the influence of specific samples, they do not address the root cause of the problem. Instead, they merely "fight fire with fire", in the case of masking it with another model. Moreover, applying a filter could inadvertently make models more vulnerable to Data Extraction Attacks, as sensitive information might be embedded not only in the model but also in the filtering mechanism itself.

Another point to highlight is, based on Layer-wise Distance analysis, GeFeU appears to be more conservative than FEMU+Gen, making fewer overall modifications while achieving similar—if not superior—performance. This brings up the question of how much is necessary to change? Should there be a limit to the amount of changes done to the model, maybe based on the size of the model?

Verifiability and Evaluation of MU This leads to an important consideration: how should we evaluate modification techniques in terms of both their impact on the model and their ability to preserve performance? A potential new metric could combine these factors, balancing the extent of model modifications against the level of performance retention.

From this perspective, *Simple Gradient Ascent* stands out as a strong candidate at first. It induces minimal changes to the model but significantly degrades its performance. Conversely, while FEMU+Gen preserves performance more effectively, it requires more modifications, which we consider less desirable, than GeFeU.

Additionally, we should explore how to scale MU techniques based on various factors, such as task complexity, training data size, model architecture, and the volume of data to be unlearned. This principle also applies to the suggested evaluation metrics.

For instance, within the GeFeU framework, the scope of the *Error Maximization Noise Generation* process should vary depending on whether we aim to unlearn a single sample or a thousand. Similarly, in the *Impair* and *Repair* phases, the learning rate and the volume of *Retained* data must be considered. While it might seem straightforward to scale based on the amount of data to forget, model size and data dimensionality introduce additional complexities. A key question remains: How does data dimensionality influence the unlearning process? Is it merely a matter of scaling, or does it introduce deeper structural challenges?

Returning to Equation 3, we previously emphasized the necessity of specific samples for the process to function effectively. Theoretically and practically, we could extend this concept

by replacing `f(x)` with an arbitrary `y`, allowing us to eliminate undesirable outputs selectively. This could pave the way for a feedback-driven approach akin to a generator-discriminator architecture, maybe similar to how LLM are trained using *Reinforcement Learning*.

Sequentially applying the discussed MU algorithms should also be considered worth investigating. Is it the same as performing unlearning in a single step? Does the overall performance degrade with each iteration?

from a MLP to LLMs While our experiments focus on the relatively simple task of classifying MNIST images, the implications of model unlearning extend far beyond this small-scale use case. Large language models (LLMs) present a significantly greater challenge due to their high-dimensional feature space, massive training datasets, and intricate dependencies across tokens.

A promising direction for future work is to extend the GeFeU framework to LLMs by adapting its *Error Maximization Noise Generation* mechanism to textual representations. Similarly, the *Impair* and *Repair* phases could be adjusted to account for token-based learning.

While unlearning techniques have shown promise in simple classification tasks, scaling them to large models will require a deeper understanding of knowledge representation (moving from *Example-base* to *Knowledge-based* 4), computational efficiency, and ethical constraints. Addressing these challenges is crucial for developing robust and verifiable machine unlearning systems for real-world applications.

10 Conclusion

Ultimately, this thesis bridges the gap between theoretical unlearning methods and practical implementations, demonstrating that fine-grained unlearning is achievable through gradient-based approaches. The results highlight the potential for future research while also revealing the challenges and complexities that remain to be addressed.

By conducting experiments across multiple datasets and models, we gained a deeper understanding of MU algorithms. On one hand, these experiments demonstrate the feasibility of the approach; on the other, they reveal the inherent issues and challenges that accompany it.

Looking ahead, AI and machine learning systems must be designed with human-centric principles in mind. This includes ensuring that these systems can adapt to ethical and societal expectations, particularly regarding privacy and data usage. Compliance with these principles should be a fundamental requirement for AI systems. However, many advanced AI

models, such as LLMs and text-to-image generators, remain largely unregulated from a legal standpoint.

Machine unlearning offers a way to gain greater control over the learning and behaviour of AI models. It represents a step forward in addressing critical issues such as bias mitigation, security, and system accountability. By enabling the removal of specific data influences, unlearning contributes to preventing data misuse, ensuring algorithmic fairness, improving model transparency, and making AI more understandable.

References

[1] "Convolutional neural network," 2025, accessed: 2025-01-31. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[2] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[3] "General data protection regulation," 2025, accessed: 2025-01-31. [Online]. Available: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

[4] "Large language model," 2025, accessed: 2025-01-31. [Online]. Available: https://de.wikipedia.org/wiki/Large_Language_Model

[5] "Mutlilayer perceptron," 2025, accessed: 2025-01-31. [Online]. Available: https://en.wikipedia.org/wiki/Multilayer_perceptron

[6] D. Yasmina, R. Karima, and A. Ouahiba, "Traffic signs recognition with deep learning," in *2018 International Conference on Applied Smart Systems (ICASS)*, 2018, pp. 1–5.

[7] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, 2023.

[8] J. J. Valletta, C. Torney, M. Kings, A. Thornton, and J. Madden, "Applications of machine learning in animal behaviour studies," *Animal Behaviour*, vol. 124, pp. 203–220, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003347216303360

[9] E. G. Howe III and F. Elenberg, "Ethical challenges posed by big data," *Innovations in Clinical Neuroscience*, vol. 17, no. 10-12, pp. 24–30, October 2020.

[10] C. O'Neil, *Weapons of Math Destruction*. Crown Books, 2016.

[11] P. Quinn and G. Malgieri, "The difficulty of defining sensitive data—the concept of sensitive data in the eu data protection framework," *German Law Journal*, vol. 22, no. 8, p. 1583–1612, 2021.

[12] J. Xu, Z. Wu, C. Wang, and X. Jia, "Machine unlearning: Solutions and challenges," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 3, pp. 2150–2168, 2024.

[13] Laubheimer, "How do generative ai systems work?" 2024, accessed: 2025-01-31. [Online]. Available: https://www.nngroup.com/articles/how-ai-works/

[14] A. Burkov, *The hundred-page machine learning book*. Andriy Burkov Quebec City, QC, Canada, 2019, vol. 1.

[15] Pushkar, "Understanding optimizers in deep learning: Exploring different types," 2023, accessed: 2025-01-18. [Online]. Available: https://medium.com/codersarts/understanding-optimizers-in-deep-learning-exploring-different-types-88bcc44ff67e

[16] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial intelligence review*, vol. 55, no. 1, pp. 291–322, 2022.

[17] V. Gudivada, A. Apon, and J. Ding, "Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations," *International Journal on Advances in Software*, vol. 10, no. 1, pp. 1–20, 2017.

[18] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022022, feb 2019. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1168/2/022022

[19] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Information and Software Technology*, vol. 127, p. 106368, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920301373

[20] E. Azzali, "Accountability in ai as global issue," *Management*, vol. 20, p. 22, 2020.

[21] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[22] O. Ibitoye, R. Abou-Khamis, M. el Shehaby, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security – a survey," 2023. [Online]. Available: https://arxiv.org/abs/1911.02621

[23] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: a survey of case studies," *ACM computing surveys*, vol. 55, no. 6, pp. 1–29, 2022.

[24] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," *arXiv preprint arXiv:1812.11806*, 2018.

[25] M. Kumar, X. Zhang, L. Liu, Y. Wang, and W. Shi, "Energy-efficient machine learning on the edges," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, pp. 912–921.

[26] K. Z. Liu, "Machine unlearning in 2024," May 2024. [Online]. Available: https://ai.stanford.edu/~kzliu/blog/unlearning

[27] T. Shaik, X. Tao, H. Xie, L. Li, X. Zhu, and Q. Li, "Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[28] W. Wang, Z. Tian, C. Zhang, and S. Yu, "Machine unlearning: A comprehensive survey," *arXiv preprint arXiv:2405.07406*, 2024.

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[30] N. Aldaghri, H. Mahdavifar, and A. Beirami, "Coded machine unlearning," *IEEE Access*, vol. 9, pp. 88 137–88 150, 2021.

[31] M. Pawelczyk, J. Z. Di, Y. Lu, G. Kamath, A. Sekhari, and S. Neel, "Machine unlearning fails to remove data poisoning attacks," *arXiv preprint arXiv:2406.17216*, 2024.

[32] 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.

[33] N. Aldaghri, H. Mahdavifar, and A. Beirami, "Coded machine unlearning," *IEEE Access*, vol. 9, pp. 88 137–88 150, 2021.

[34] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7210–7217.

[35] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.

[36] 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.

[37] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[38] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational bayesian unlearning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 025–16 036, 2020.

[39] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, "Mixed-privacy forgetting in deep networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 792–801.

[40] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 516–11 524.

[41] K. Y. Tan, Y. Lyu, Y. S. Ong, and I. W. Tsang, "Unfolded self-reconstruction lsh: Towards machine unlearning in approximate nearest neighbour search," *arXiv preprint arXiv:2304.02350*, 2023.

[42] Z. Ma, Y. Liu, X. Liu, J. Liu, J. Ma, and K. Ren, "Learn to forget: Machine unlearning via neuron masking," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3194–3207, 2022.

[43] P. Maini, Z. Feng, A. Schwarzschild, Z. C. Lipton, and J. Z. Kolter, "Tofu: A task of fictitious unlearning for llms," *arXiv preprint arXiv:2401.06121*, 2024.

[44] N. Li, A. Pan, A. Gopal, S. Yue, D. Berrios, A. Gatti, J. D. Li, A.-K. Dombrowski, S. Goel, L. Phan *et al.*, "The wmdp benchmark: Measuring and reducing malicious use with unlearning," *arXiv preprint arXiv:2403.03218*, 2024.

[45] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *Icdar*, vol. 3, no. 2003. Edinburgh, 2003.