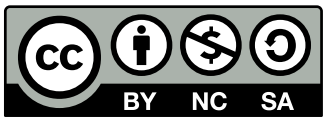
Aalto University
School of Science

Master's Programme in School of Science

Zero-shot Machine Unlearning using GANs

Ali Ghazal

Master's Thesis
2024



Aalto University
School of Science

Author Ali Ghazal

Title Zero-shot Machine Unlearning using GANs

Degree programme School of Science

Major Master's Programme in Security and Cloud Computing (SECCLO)

Supervisor Prof. Radwa El Shawi

Advisor Prof. Alex Jung

Date 18 June 2024 **Number of pages** 52+3 **Language** English

Abstract

This thesis addresses the crucial task of machine unlearning, which involves the removal of specific data from trained machine-learning models to comply with privacy regulations and enhance data quality. With the rapid advancements in AI and the extensive use of machine learning models in various applications, efficient unlearning methods are increasingly urgent. Traditional approaches, such as retraining models from scratch, are impractical due to high resource consumption and time constraints.

The thesis proposes two innovative techniques: MuGAN and zMuGAN. MuGAN, short for **M**achine **u**nlearning using **GAN**s, is designed for scenarios with limited access to the original training dataset. It uses Generative Adversarial Networks (GANs) to capture the data distribution during the model's initial training and generate synthetic data for unlearning upon receiving such a request. Similarly, zMuGAN, short for **z**ero-shot **M**achine **u**nlearning using **GAN**s, addresses situations where no access to the training data is available at all. It utilizes a GAN-based model inversion technique to approximate the original dataset and facilitate unlearning through an impair-repair unlearning process. Both techniques are evaluated on image classification tasks, particularly class forgetting, highlighting the sensitive nature of image data. The proposed methods effectively preserve model utility while ensuring effective unlearning.

The primary contribution of this thesis is proposing robust and efficient solutions for machine unlearning for scenarios with restricted data access. By leveraging GANs and innovative unlearning processes, MuGAN and zMuGAN offer significant advancements in the field, addressing the urgent need for practical and scalable unlearning techniques.

Keywords Computer Science, Artificial Intelligence, Machine Learning, Privacy

Preface

I would like to thank the SECCLO staff, my supervisor, Prof. Radwa El Shawi, My advisor, Prof Alex Jung, and my manager at work, Rommel Vasquez, for their guidance and support.

Otaniemi, 16 May 2024

Ali Mohamed Ghazal

With the support of the Erasmus+ Programme of the European Union

Contents

1 Introduction

1.1 Problem Overview

With the overwhelming advances in AI, machine learning models with their variations have become an indispensable part of modern life. They are an essential part of healthcare services, autonomous machines, and security applications. Most companies and products started using AI to streamline processes and give users smoother, refined experiences. With such rapid development, there is an insistent need to develop methods for unlearning – or "forgetting" – irrelevant, privacy-violating, or inaccurate data [5, 4, 29].

As commonly known, ML models' quality depends heavily on the data used for training. With the emergence of big data and the wide deployment of ML pipelines, ensuring the quality of those data points is becoming increasingly difficult. Having corrupted or biased data would reflect on the models' prediction. Depending on the application, many machine-learning models undergo data drifts where the original data used for training becomes irrelevant. Moreover, due to emerging privacy regulations such as GDPR [30, 21] and California Consumer Privacy Act (CCPA) [17], users now have "the right to be forgotten." Although still vague, many understand this to include the removal of private data that was used everywhere, including the weights of the machine learning models used in this service. Thus, developing machine unlearning techniques is imperative to securely and effectively deploy machine learning models.

The term machine unlearning was first coined by [5] in 2015, and it gained popularity as Google published a technical challenge under the same name [1]. Practically speaking, machine unlearning could be defined as the process of removing data from a trained model so that it would behave in a similar fashion to a model that has never seen the data before [4]. The naive solution for unlearning a subset of data would be retraining a model from scratch. However, this is very ineffective and impractical for a number of considerations. First, it unnecessarily consumes a lot of resources. For example, it was reported that training ChatGPT-3 cost OpenAI around 20 million dollars [43]. Thus, it would be unthinkable for most companies to re-train their models after each privacy complaint or whenever a biased document is detected. Second, it takes a lot of time. Thus, assuming companies would comply and retrain the model from scratch, removing the data would be delayed at best.

Thus, the research community developed various techniques for different machine-learning tasks and models' architectures. Yet, the field is still considered to be in its infancy. Many challenges still hinder the wide deployment of such techniques. First, most of the techniques suggested are not universally applicable to all ML architectures. Second, most unlearning techniques require access to the training dataset, the *forget* dataset, or both. This is often impossible because those datasets could be proprietary, deleted, or decentralized. Third, unlearning techniques vary in their efficiency. Some consume more resources and time compared to the cost of retraining.

1.2 Scope and Goal

This thesis aims to address the problem of not having access to any observable data. We are addressing two variants of this problem. The first variant is having time-limited access to a dataset. In the second variant, we do not have any access to the training data at all. In literature, those techniques are called one-glance and zero-shot, respectively.

We are proposing two innovative techniques that would enable unlearning without access to the *retain* dataset or the *forget* dataset. For the first technique, zMuGAN, GAN are used to perform model inversion to recover data points that would be used to perform unlearning [39]. For the second technique, MuGAN, assuming having limited access to the training dataset, we train GANs to capture the main distribution of the data. Later, upon receiving a deletion request, we generate samples from the trained GANs and use them for unlearning. Those two unlearning setups are selected for several reasons. First, to the best of our knowledge, the zero-shot and the one-glance unlearning setups are not addressed enough in the literature, and the few existing solutions heavily compromise the model's utility, rendering them unattractive. Second, by addressing those two setups, we are empowering ML researchers and engineers throughout the different stages of model design and deployment. That is, if a model is in the early development stages, MuGAN could be used to enable unlearning requirements; further, if a model is already in production, z-MuGAN could be used to perform unlearning without requiring any additional storage or modification of the training process.

This thesis focuses mainly on the task of removing – unlearning – information used by deep learning models. The selected use case is image classification. The rationale behind the selection was twofold. First, images represent one of the most sensitive data types; thus, this task has been widely studied in literature [15, 16, 14]. Second, due to the non-convex nature of computer vision models, there is a big gap in the literature addressing the particular setup we are working with – namely, data-restricted setups. The scope of this thesis mainly addresses the task of class forgetting, in contrast to data points forgetting or dataset forgetting. This setup was selected because it is used widely in common AI applications. For example, in face recognition models, FaceID is usually modeled as a discrete class. Thus, the task of forgetting a person would translate to the task of forgetting their associated class. Throughout our work, we will demonstrate the effectiveness of our proposed techniques in navigating those challenges.

1.3 Structure

This thesis has the following structure: in Chapter 2, we provide a literature review of the field to equip the reader with the necessary background information. In Chapter 3, we discuss our proposed technique. In Chapter 4– The Experiment Setup, we detail the experimental setup, the selected models, and the datasets used to evaluate our technique. In Chapter 5, we list the collected results and provide a comprehensive

comparison between the performance of our technique and baseline techniques. In Chapter 6, we provide commentary on the obtained results and future work. Lastly, in Chapter 7, we summarize the work.

2 Background

2.1 Machine Learning

Machine learning (ML) is a subset of artificial intelligence that involves creating algorithms capable of performing specific tasks without explicit instructions. Instead, these models rely on patterns and inference to make predictions or decisions. Typically, ML models are trained using vast amounts of data, and algorithms learn from this data. Consequently, models can make informed judgments based on learned patterns without requiring explicit programming to execute their tasks. Generally, most machine learning problems fall under two categories: regression and classification tasks. In regression tasks, the ML model tries to produce a continuous value output, such as daily temperature or property price. In contrast, classification tasks involve selecting a label or a class that best describes the input, such as plant species detection or face recognition.

Various machine learning algorithm families exist, including supervised learning, unsupervised learning, self-supervised models, and reinforcement learning [23]. In supervised learning, the algorithm is trained using a labeled dataset, where each instance in the training dataset is associated with an output label. The primary goal of supervised learning is to find a mapping between the inputs and outputs, enabling the prediction of output values for new data. Some widely used supervised learning algorithms are linear regression, logistic regression, support vector machines, and neural networks.

Unlike supervised learning, unsupervised learning algorithms are used when the information used to train is neither classified nor labeled. This type of learning studies how systems can infer a function to describe a hidden structure from unlabeled data. Some of the widely used unsupervised learning algorithms include (e.g., k-means [3] and hierarchical clustering [11])

2.1.1 Deep Learning

Deep learning, a subset of machine learning, has proven highly effective across various fields, such as computer vision, natural language processing, and time-series analysis. Its success is attributed to its deep architecture, consisting of multiple neural network layers. These architectures enable the models to learn complex data representations, particularly excelling with unstructured data like images, audio, and text. By autonomously extracting features and identifying hidden patterns, deep learning achieves superior inferential performance [18].

Convolutional Neural Networks (CNNs) are a category of deep neural networks that are particularly powerful for tasks related to recognizing and classifying images. They are also used effectively in other areas, such as video recognition, recommender systems, and natural language processing. CNNs are distinguished from other neural networks by their architecture, specifically designed to process pixel data [26].

CNNs consist of an input layer, an output layer, and multiple hidden layers. The hidden layers of a CNN typically include a series of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here's a brief description of these main layers:

Convolutional Layer: The primary building block of a CNN. This layer applies a convolution operation to the input, passing the result to the next layer. It helps CNNs learn features from the input images which are essential for recognizing objects.

Pooling Layer: Also known as a downsampling layer, this layer reduces the input volume's spatial dimensions (width, height) for the next convolutional layer. It is used to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it helps extract dominant features, which are rotational and positional invariants, thus maintaining the process of effectively training the model.

Normalization Layer: This layer normalizes the previous layers' output to improve the model's convergence rate and reduce the sensitivity to network initialization [22].

Fully Connected Layer: Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. Their activation can hence be computed with a matrix multiplication followed by a bias offset.

By utilizing these layers effectively, CNNs can successfully capture the spatial and temporal dependencies in an image by applying relevant filters, allowing them to learn features at various levels of abstraction.

Generative Adversarial Networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. This technique was introduced in [19] in 2014. A GAN consists of two parts:

Generator: The generator's role is to produce data that is indistinguishable from genuine data. It learns to create plausible data. The generated instances become positive training examples for the discriminator.

Discriminator: The discriminator's role is to identify whether a given data instance is real (coming from the training set) or fake (created by the generator). It is a classifier.

The generator and the discriminator are usually trained simultaneously: the generator aims to increase the error rate of the discriminator by producing novel synthesized instances, while the discriminator tries to decrease the error rate by learning to distinguish real from fake.

There are different GAN architectures depending on the target use case and the loss function used. In this thesis, we utilized conditional GANs, Deep convolutional GANs, and WGANs.

Deep Convolutional GANs (DCGANs): These use convolutional and convolutional-transpose layers in the discriminator and generator, respectively. They are widely used for image-generation tasks.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

Equation 1 describes the min-max optimization objective of training a GAN. In the first term, the discriminator aims to maximize its output probability on real data. In the second term, the generator is trying to produce a realistic output that would fool the discriminator into predicting that the fake output is realistic.

Conditional GANs (cGANs): These are an extension of the basic GAN architecture which includes additional conditional variables that influence the generation process, allowing the generation of images of specific classes.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z), y\sim p_{\text{data}}(y)}[\log(1 - D(G(z|y), y))] \quad (2)$$

The equation in 2 is the same as 1. The only difference is conditioning on the label y.

2.2 Machine unlearning

After the massive popularity and wide deployment of machine learning models, the need for techniques to remove data from trained models has become increasingly important. Such need stems from privacy, usability, and regulatory requirements. As mentioned in section **??**, removing – unlearning – corrupted data points leads to better-performing models. Further, after receiving a data removal request, it is not enough for the concerned parties to remove data only from their databases. That is, attacks such as Model Inversion Attacks and Membership Inference Attacks could reveal unintended information about the samples used during training. This motivated the advancements in the field of machine unlearning. That is, the field aims to study the efficient removal of data points along with their influence on trained machine-learning models. It should be clear that the purpose of machine unlearning intersects with other domains, such as differential privacy, data anonymity, lifelong learning, catastrophic forgetting, and fairness. Here are the differences between machine unlearning and those other domains.

Differential privacy aims to minimize the contribution of each data point to the final trained model. This would lead to a situation where, by only analyzing the output of the model, it couldn't be inferred whether a data point was used for training [12]. Such a field intersects with the privacy requirement posed by machine unlearning. However, machine unlearning aims at the removal of samples from the model without imposing any privacy requirements on the initial model.

Data anonymization aims to conceal sensitive data in the training dataset [37]. In contrast, machine unlearning aims to remove data points regardless of their content and the initial condition of the training dataset.

Lifelong Learning aims to continually update machine learning models to adapt to the changes happening to the data while at the same time retaining the information that was previously learned. Machine unlearning intersects with Lifelong Learning in its application to make machine learning models more adaptive [32]. However, machine unlearning could be understood as an inverse process that is mainly concerned with the removal of data. On the other hand, unintended forgetting, also named catastrophic forgetting, describes the phenomenon when models' performance degrades over time as they fail to *retain* previously learned information. Indeed, the same principle could be used to induce forgetting. However, where catastrophic forgetting is concerned with an unintentional phenomenon, unlearning is an intentional phenomenon that could be triggered by service providers.

Fairness , as related to trustworthy AI systems, aims to ensure the fair representation of classes in machine learning models. Indeed, advances in machine unlearning could heavily contribute to ensuring the fairness of models where corrupted, over-represented, or biased data points could be removed [7]. However, fairness has different objectives compared to machine unlearning. The exact objectives of machine unlearning will be introduced in next section.

2.2.1 Preliminary

Now, before providing a definition of the task machine unlearning, a few related concepts and notations should be defined first.

Based on the supervised learning context, training samples could be defined as a collection of points (X, y) where $X \in R^d$ and $y \in R$. The general purpose of machine learning is to "train" a model M with parameters θ using some learning Algorithm $\mathcal{A}$. Let's denote the original dataset used during the initial training process as D. For the context of unlearning, let's define D_f to be the subset of D that we are interested in forgetting – i.e. unlearning. On the other hand, let's define D_r to be the remaining subset of D that we are interested in retaining.

$$D_r = D \setminus D_f$$

Definition 2.1 (Machine Unlearning) *given a training dataset D, a trained model $M = \mathcal{A}(D)$, a retain dataset D_r, and a forget dataset D_f, an Unlearning algorithm $\mathcal{U}$ aims to produce $M_f = \mathcal{U}(M)$ and ensures that M_f performs as if it has never seen D_f before.*

The outcome of any unlearning algorithm would roughly fall into one of two categories: exact unlearning or approximate unlearning. Given a model M_r trained from scratch without D_f, exact unlearning aims for the unlearned model M_f to have the same output distribution as M_r. Although very difficult to implement, it ensures that no data at all could be recovered from the model. Approximate unlearning is more suitable for more complex models and has more relaxed requirements [29]. Although easier to implement, it doesn't provide any guarantees that the influence of those data points is fully removed from the model's internals. Strong approximate unlearning aims to have the distribution of the unlearned and retrained models' parameters within an acceptable threshold. On the other hand, weak approximate unlearning enforces the constraints only on the final activation layers of both models.

The objectives of Machine Unlearning could be summarized in the following points:

The first objective is to address privacy concerns in ML models. That is, machine unlearning algorithms should ensure the removal of data points and their influence from trained models. Thus, it reduces the risks of inversion and membership inference attacks.

The second objective is to improving the utility and fairness of ML models [25]. By having an efficient, effective unlearning technique, service providers would be able to easily remove corrupted or biased data points, resulting in a safer and fairer service.

2.2.2 Unlearning Techniques

Over the years, many unlearning algorithms have been proposed. Unlearning techniques, based on their mood of operation, could be divided into two categories: model manipulation and data manipulation. Model manipulation is the process of modifying the internal weights of the model that are associated with the data selected for forgetting. On the other hand, data manipulation is the process of curating a new dataset that would be used later to train the model. Such a training process with those data points will cause the deletion of the influence of the designated *forget* dataset.

Data Manipulation Multiple techniques fall under the data manipulation category. In this section, we will provide an overview of how data manipulation techniques generally work, and we will provide an in-depth description of the methods that provided state-of-the-art performance in the setups related to the one addressed in this thesis. Namely, we will discuss UNSIR [35], Just-in-Time Unlearning (JiT) [13], Zero-shot unlearning using Gated knowledge Transfer (GTK), Error Minimization-Maximation Noise (EMMN) [10], and lastly, Amnesiac Unlearning [20].

UNSIR In this work, the concept of zero-glance unlearning was introduced. It was concerned with scenarios where the *forget* dataset is not available, yet the *retain* dataset is. This is a practical use case since, maybe, due to regulations, data deletion requests could result in the timely deletion of data points from the database. Thus, at the time of performing unlearning, we only have access to the remaining dataset and the class we are concerned with forgetting. Similar to other works presented in this thesis, UNSIR, short for **UN**learning by **S**elective **I**mair and **R**epair, mainly targets deep learning models. Additionally, the analysis provided in their research proves its effectiveness for computer vision tasks, namely face recognition. To the best of our knowledge, UNSIR is one of the earliest works to address the task of multi-class unlearning, making it an influential work in the field.

UNSIR, by design, requires only access to the *retain* dataset. The forgetting process consists mainly of two steps: Impairing and repairing. During the impair phase, the model's weights are initially frozen, and it would be used to generate noise samples that would maximize the classification loss corresponding to the unlearning class [35]. The utilized loss function is listed in equation 3. In cases where multiple classes are targeted for deletion, the noise would be maximized according to the loss of the classes intended for deletion without needing to execute the algorithm more than once. Those data points would then be used along with a few samples from the *retain* dataset to train the mode. Next, in the repair phase, the model would be fine-tuned with a subset of the *retain* dataset. Through experimentation, the authors of the work suggest that only one step of impair-repair steps is enough to have the desired forgetting effect.

$$\arg\min_{\theta} \frac{1}{N}\mathbb{E}_{\theta}\left[-\mathcal{L}(f, y) + \lambda\|\mathbf{w}_{\text{noise}}\|\right] \tag{3}$$

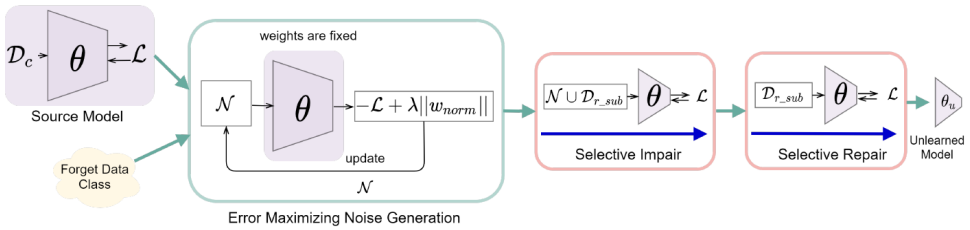


Figure 1: A conceptual diagram of UNSIR from [35]. UNSIR accepts as an input the trained model, the *retain* dataset. In the first step, UNSIR generates noise that aims to maximize the loss function evaluated on the target forget class. In the second and third steps, the model would be first trained with a dataset containing the noise input. Then, it would be trained with samples from the *retain* dataset.

Error Minimization-Maximation Noise (EMMN) The authors in [10] were the first to propose methods that target the zero-shot unlearning setup– i.e., situations where we don't have access to the *retain* data D_r or the data requested for forgetting D_f. In their work, two methods were proposed. The technique at hand, EMMN, is an extension of UNSIR [35]. In UNSIR, error-maximization was used in the impair step to generate noisy samples that would later be used to induce forgetting for the target class. EMMN used the same concept. In contrast to UNSIR, EMMN does not have access to any data points. EMMN could not perform a repair class with data from the original dataset based on the assumptions of the problem setup. Thus, the method at hand uses an error-minimization step to generate sample points that could represent the data from the retaining class. Later, the same impair-repair steps could be repeated until the desired performance is achieved. In practice, this technique performed very poorly, and that's why the authors of the same work introduced Gated Knowledge Transfer, GTK for short.

Zero-shot unlearning using Gated Knowledge Transfer (GKT) The *Gated Knowledge Transfer* (GKT) method was proposed as an enhancement to the Error Minimizing-Maximizing Noise (EMMN) approach in the cited work. This method utilizes a knowledge distillation strategy where a generator is employed to produce data points that intentionally increase the Kullback-Leibler (KL) divergence between the outputs of the student and teacher models. An essential addition is a gating mechanism, which selectively permits the flow of data points that do not pertain to the class designated to be forgotten [10].

The loss function in GKT combines the KL divergence and an attention loss, described by the following equations:

$$L_s = D_{\mathrm{KL}}(T(x_p) \| S(x_p)) + \beta L_{\mathrm{at}} \tag{4}$$

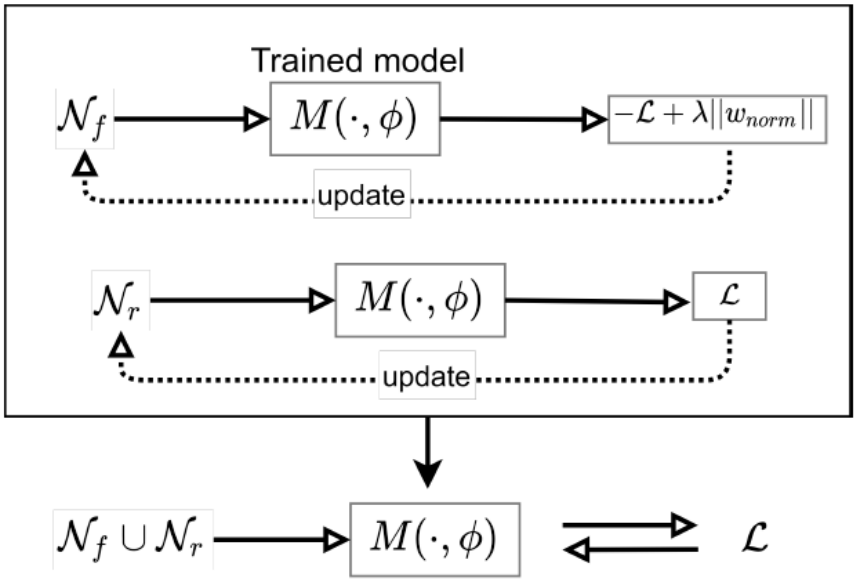


Figure 2: A conceptual diagram of EMMN from [10]. This figure describes the three main steps of EMMN. First, it generates noise to minimize the loss function on the *retain* classes. Next, it generates noise that would increase the loss on the *forget* class. Then, both datasets would be combined and used to train the model.

$$L_{\text{at}} = \sum_{l \in N_L} \frac{f(A_l^{(t)}) - f(A_l^{(s)})}{\|f(A_l^{(t)})\|_2 \|f(A_l^{(s)})\|_2} \tag{5}$$

Here, L_s represents the total loss, with D_{KL} measuring the divergence between the teacher's and the student's predictions on the processed data point x_p, and L_{at} quantifying the differences in attention mechanisms across layers, weighted by β.

The gating mechanism serves as a critical filter, utilizing the probability outputs from the teacher model to assess the relevance of each synthetic data point. Data points exhibiting a probability of association with the target forget class above a predetermined threshold are blocked, where this threshold is a hyperparameter.

Despite the theoretical promise, practical applications of the GKT approach have revealed significant shortcomings. Experiments have shown that the results initially reported were difficult to replicate using the provided source code. Additionally, the method required an impractically long execution time coupled with severely impairing the model's utility, raising concerns about its viability in real-world scenarios. This underscores the necessity for further optimization and validation to ensure the method's practicality and reliability.

Zero-shot unlearning using Lipschitz Regularization (JiT) Just-in-time Forgetting was introduced as an unlearning technique for the zero-shot setup proposed by [13]. Previously, in [10], zero-shot setup was characterized by not having access to either the forget or the retaining subset. In this work, an altered definition of the setup was assumed. Just-in-time Forgetting assumed having access to the *forget* dataset while not having to the *retain* dataset. This setup is relevant where models would receive the deletion requests along with the data that needs to be forgotten. Yet, it must be pointed out that it differs from the situation addressed in this thesis since we assume the inaccessibility of the entire training data at the moment of receiving a deletion request.

JiT is based on the concept of Lipschitz Continuity, a technique that was promoted by [41] to neural networks to reduce their sensitivity to input changes and to enhance their generalizing abilities. A model is said to Lipschitz continuous if there exists a constant k such that:

$$\|f_\theta(x) - f_\theta(y)\|_p \leq k\|x - y\|_p \tag{6}$$

JiT applies the same concept to "smooth" the model response to input perturbations. That is, it minimizes the difference between perturbed input samples and the original input. In other words, it updates the weights of the model to respond to samples from the *forget* dataset as if they were random samples. The steps of JiT are detailed in Algorithm 1. So, for a given sample from the *forget* dataset, JiT aims to optimize equation 7:

$$\ell = \mathbb{E}\left(\frac{\|f_\theta(x) - f_\theta(x+\xi)\|_2}{\|x - (x+\xi)\|_2}\right) \approx \frac{1}{N}\sum_{j=1}^{N}\frac{\|f_\theta(x) - f_\theta(x+\xi_j)\|_2}{\|\xi_j\|_2} \tag{7}$$

Algorithm 1 JIT Unlearning (based on [13])

Require: Model: $f_\theta(\cdot)$, Forget set: S, Optimizer: optim, η, σ, N

Initialise optim$(\theta, \text{lr} = \eta)$
for $x \in S$ **do**
    $\ell = 0$
    for $i \in \text{range}(N)$ **do**
        $x' = x + \xi$ for $\xi \sim \mathcal{N}(0, \sigma^2)$
        $k = \frac{\|f_\theta(x) - f_\theta(x')\|_2}{\|\ell\|_2}$
        $\ell = \ell + k$
    end for
end for
$\ell = \ell / N$
$\theta \leftarrow \text{optim}\nabla_\theta \ell$
return $f'_\theta(\cdot)$

This technique operates with varying effectiveness in handling different unlearning cases, such as class forgetting, sub-class forgetting, and data point forgetting. One notable advantage is that it does not require access to the retraining dataset. However,

it has several drawbacks, some of which stem from the inherent nature of the problem. Like other unlearning algorithms, JiT is sensitive to hyperparameter selection. Additionally, JiT faces a specific limitation in its reduced unlearning performance for models with batch normalization. This limitation arises because the optimized loss is calculated for each individual sample rather than for entire batches.

Randomized Label Unlearning: This method was introduced in [20] under the name Unlearning. The *forget* subset would be assigned a randomized label upon receiving a deletion request. Then, the model would be fine-tuned with this subset for a number of epochs until the desired forgetting effect takes place.

Amnesiac Unlearning: The following technique was suggested to address a privacy concern linked to the randomization method mentioned earlier. Specifically, sharing the *forget* dataset with service providers was considered not privacy-preserving since they could potentially store data versions. [20] proposed an unlearning technique based on modifying the training process. This involved using a separate database to track the epoch, batch, and parameter update of each data point. Upon receiving a deletion request for a data point, the parameter update for that point would be subtracted from the model. However, this technique has two main drawbacks: first, it imposes a significant storage cost on the service providers, and second, the model's performance would be severely compromised after receiving multiple deletion requests.

Model Manipulation: The model manipulation category encompasses techniques such as SCRUB and Bad Teacher. Both SCRUM and Bad teacher Algorithms are based on the teacher-student training paradigm.

SCRUM: In [25], the authors contributed an unlearning technique based on the teacher-student paradigm. Their technique requires access to the entirety of the remain and the *forget* dataset. Assuming accessibility of the training dataset, SCRUB, short for **SC**alable **R**emembering and **U**nlearning un**B**ounded, is effective for multiple unlearning scenarios, including reducing classification confusion, removing biases, and, most importantly, promoting users' privacy. SCRUB starts by instantiating two replicas of the original model; one would be designated as a student and the other as a teacher. Then, the student would be trained to obey the teacher with regard to the *retain* dataset and to disobey it with regard to the *forget* dataset. This could be naturally modeled as minimizing the $\mathcal{KL}$-divergence between the teacher and the model with regard to D_r and maximizing it with regard to D_f. Further, the authors added an additional term to the optimization function; namely, they minimized the loss of the unlearned model on the retaining data.

$$\min_{w^u}\left(\frac{\alpha}{N_r}\sum_{x_r\in D_r} d(x_r;w^u)+\frac{\gamma}{N_r}\sum_{(x_r,y_r)\in D_r} l(f(x_r;w^u),y_r)-\frac{1}{N_f}\sum_{x_f\in D_f} d(x_f;w^u)\right) \tag{8}$$

Algorithm 2 SCRUB from [25]

Require: Teacher weights w^0
Require: Total max steps MAX-STEPS
Require: Total steps STEPS
Require: Learning rate ϵ

$w \leftarrow w^0$
$i \leftarrow 0$
repeat
    if $i <$ MAX-STEPS **then**
        $w \leftarrow$ DO-MAX-EPOCH(w)
    end if
    $w \leftarrow$ DO-MIN-EPOCH(w)
    $i \leftarrow i + 1$
until $i \geq$ STEPS

Algorithm 3 DO-MAX-EPOCH

Require: Student weights w
Require: Learning rate ϵ
Require: Batch size B
Require: Forget set D_f
Require: Procedure NEXT-BATCH
  $b \leftarrow \text{NEXT-BATCH}(D_f, B)$
  repeat
    $w \leftarrow w + \epsilon \frac{\nabla_w}{|b|} \sum_{x \in b} d(x; w)$
    $b \leftarrow \text{NEXT-BATCH}(D_f, B)$
  until b

Algorithm 4 DO-MIN-EPOCH

Require: Student weights w
Require: Learning rate ϵ
Require: Batch size B
Require: Retain set D_r
Require: Procedure NEXT-BATCH
  $b \leftarrow \text{NEXT-BATCH}(D_r, B)$
  repeat
    $w \leftarrow w - \epsilon \frac{\nabla_w}{|b|} \sum_{(x_r, y_r) \in b} \alpha d(x_r; w) + \gamma l(f(x_r; w), y_r)$
    $b \leftarrow \text{NEXT-BATCH}(D_r, B)$
  until b

Acknowledging the trade-offs between different unlearning objectives and the different unlearning scenarios, the authors in [25] introduced a variant named SCRUB + R, where R stands for Rewind. Mainly, the only difference is in selecting which intermediate checkpoint demonstrates the intended unlearning characteristics. That is, SCRUB could result in an aggressive forgetting of samples, which could make them identifiable by an adversary performing a membership inference attack. Thus, the authors recommended evaluating the model at the end of the SCRUB algorithm to benchmark the model's performance on D_f and D_r. Then, the intermediate checkpoint displaying the required performance and privacy characteristics will be selected.

Bad Teacher: Similar to SCRUB, the Bad Teacher technique, introduced in [9], follows the student-teacher paradigm. It starts by instantiating two teachers: the first is a competent teacher model, which has the same weights as the original model, and the second is an incompetent teacher model, which is randomly initialized. Initially, the student has the same weights as the original model. Then, during training, the student model is optimized to reduce the KL-divergence between its output and the incompetent teacher model with regard to samples from the *forget* dataset. In contrast, the model is optimized to reduce the KL divergence between its output and the competent teacher model with regard to samples from the *retain* dataset.

$$L(x, l_u) = (1 - l_u) \times \mathcal{KL}(T_s(x) \| S(x)) + l_u \times (\mathcal{KL}(T_d(x) \| S(x))) \tag{9}$$

2.2.3 Evaluation

Evaluating the performance of unlearning proved to be an elusive task. Multiple techniques with varying requirements have been proposed to measure different quantities based on the supported use case. As detailed in [25], most of those metrics are conflicting. For example, strong forgetting performance results in compromising the privacy of those samples. [8] experimentally proved that most SOTA unlearning techniques reveal traces that samples were unlearned; consequently, this reveals that

Table 1: Unlearning Methods Data Requirments

Method	D_f Access.	D_r Access.	Meta-data Access
UNISR	-	✓	-
EMMN	-	-	-
GTK	-	-	-
JIT	✓	-	
SCRUB	✓	✓	-
Bad Teacher	✓	✓	-
Amn (rnd) Unlearning	✓	-	-
Amn Unlearning	✓	✓	✓
MuGAN	-	-	✓
zMuGAN	-	-	-

they once were members of the training datasets. In this section, we will introduce the most common metrics utilized to evaluate the performance of unlearning algorithms.

Model Weights distance from Retrained Model: Inspired by the early definition of unlearning, this metric aimed to measure how close the newly unlearned model is to the model retrained from scratch. Multiple distance measures, such as JS-divergence [27].

Output Distribution: Similar to models' weights distance from retrained models, this metric aims to measure the similarity between the output distribution of the unlearned model and the retrained one. KL-divergence is dominantly utilized in multiple papers for both assessing the distribution distance and optimizing different unlearning algorithms.

Output Entropy: Similarly, the assumption behind this metric is that for the *forget* dataset, the entropy of the output distribution should be high, and it should be very close to the entropy of the retrained model [13].

Forget and retain Accuracy: This is one of the most commonly used metrics. Mainly, it required the existence of a test dataset in cases of class-forgetting or the existence of *forget* dataset in cases of sample-forgetting. A good unlearning algorithm would result in a low forget accuracy and a high *retain* accuracy.

Retraining Time: This metric measures the time it would take the model to re-learn the *forget* dataset to reach a similar performance – i.e., accuracy – to the original model before applying the unlearning algorithm. Mainly, a higher retraining time would mean that most of the information about the *forget* dataset is forgotten. However, one drawback is that there is no clear upper bound. Such a metric could reward corrupted models because they take longer to train.

Anamnesis Index (AIN): this metric is basically the same as the retraining time but normalized over the time it would take the model re-trained from scratch to re-learn the *forget* dataset [10]. This fixes the lack of an upper bound for the retraining time. A good-performing unlearning algorithm would achieve an Anamnesis index of 1. That is, it would take the same time as a model trained from scratch to relearn the *forget* dataset.

$$\text{AIN} = \frac{r_t(M_u, M_{\text{orig}}, \alpha)}{r_t(M_s, M_{\text{orig}}, \alpha)} \tag{10}$$

Execution time: This is a simple metric that aims to measure how long it takes for the algorithm to unlearn the *forget* dataset effectively. This aims to address the efficiency objective of unlearning algorithms. As a benchmark, any unlearning algorithm should consume less resources and time compared to a model retrained from scratch.

Membership inference attack: This is one of the most common attacks applied to machine learning models. In plain English, the purpose of this attack is to infer whether a sample belonged to a training dataset or not. To understand the risk imposed by this attack, assume a scenario where an adversary would like to know if a medical diagnostic model is trained on a particular sample. If this is the case, this would leak confidential information about patients' health. Many variants of this attack exist. The main operating principle is classifying the behavior of the model on member and non-member data points. Attacks usually involve training an ensemble of classifiers on a direct output or the model or on layers' activation scores [42]. In [6], they proposed using the loss. In [31], they demonstrated the possibility of multi-dataset/multi-modal training of attack models using an embedding of the logit output of the models.

3 Zero-shot Machine Unlearning using GANs

MuGAN and zMuGAN are proposed to address the problem of having limited access to training data upon receiving an unlearning request. zMuGAN assumes having no access to any data at all. On the other hand, MuGAN assumes having access to GANs trained to produce data points representing the distribution of the data used during training. Thus, the two proposed techniques provide an effective solution for the zero-shot and the zero-glance unlearning setup.

3.1 zMuGAN

3.1.1 Preliminary: Model Inversion

Model inversion refers to the techniques used to reverse-engineer the input data on which the model was trained, mainly based on the output of the model. This process raised numerous privacy concerns, particularly to models deployed on sensitive data. Formally defined, let f denote a machine learning model trained to map pair (x, y) where $x \in X$ and $y \in Y$. The function f could be expressed as $y = f(x, \theta)$, where θ represents the parameters of the model. The goal of model inversion is finding a $\hat{x}$ such that

$$\hat{x} = \arg\max_{x'} \mathcal{L}(f(x', \theta), y)$$

where $\mathcal{L}$ is the loss between the model on input x' and the output y.

Multiple model inversion attacks have been proposed over the years. They could roughly be classified in black-box and white-box attacks [38]. In black-box attacks, the adversary has access to only the model's inputs and outputs. Attacks in this category repeatedly query the model to gather enough information to understand the behavior of the models on different samples. On the other hand, white-box inversion attacks describe the scenario where the adversary has access to the model's internal parameters, layers' activation, and the model output.

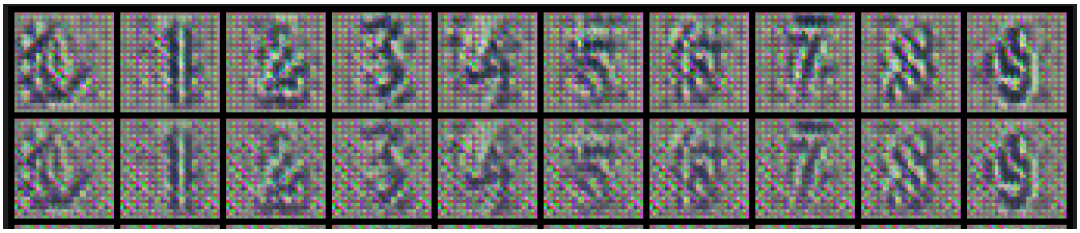


Figure 3: The output of model inversion technique utilized in zMuGAN applied on a VGG16 trained on SVHN.

3.1.2 Model Architecture

In scenarios where access to the original training data is restricted, the challenge of removing specific data points or classes from a trained machine-learning model becomes significant. Conventional approaches to machine unlearning generally necessitate direct access to the original dataset, which is often impractical due to privacy constraints, data unavailability, or proprietary limitations. To address this challenge, we introduce zMuGAN, a novel method that enables machine unlearning without requiring access to the original data.

The zMuGAN method facilitates machine unlearning through a structured sequence of operations. Figure 4 describes the steps of the technique. The algorithm of zMuGAN is listed in Alg. 5. The method starts by accepting as input the pre-trained model to be modified and the label of the target class to be forgotten. The process comprises four key steps:

1. **GAN-Based Model Inversion** Initially, a GAN-based model inversion technique is employed to approximate the dataset used for training the original model. Mainly, we utilized the architecture proposed by [39]. This step utilizes a generator and a decoder, along with the pre-trained classifier. Specifically, the generator processes a concatenated vector of a noise vector z and a label y, producing an output x. This output x is then input to the classifier to retrieve the label y and to the decoder to reconstruct the low-dimensional representation z. The generator network is trained using three losses: classification loss (from the classifier), reconstruction loss (from the decoder), and diversity loss (to enhance pixel-wise diversity in the generated outputs). The objective optimization function is listed in Eq. 11. This algorithm utilized in this step is listed in Alg. 6. A sample of this inversion technique is visualized in Fig. 3.

$$l(B) = \sum_{(f(x),\hat{z})\in B} (l_{\text{cls}}(y,\hat{z}) + \alpha l_{\text{dec}}(y,\hat{z})) + \beta l_{\text{div}}(B), \tag{11}$$

2. **Synthetic Dataset Generation**

   The trained GAN is utilized to generate a synthetic dataset that closely resembles the original training data. Based on the recommendations in [36] [39], an ensemble of GANs was used to increase the diversity of the output.

3. **Data Splitting**

   The synthetic data points are fed into the pre-trained model. Based on the model's predictions, these data points are divided into two subsets: the *retain* dataset, consisting of data points predicted as not belonging to the target forget class, and the *forget* dataset, consisting of data points predicted as belonging to the target forget class.

4. **Impair-Repair Unlearning Technique**

   Both zMuGAN and MuGAN utilize the same unlearning steps listed in Alg. 8. It consists of two steps:

   (a) **Impair Step**

   A reference model is initialized, and the Kullback-Leibler Divergence $\mathcal{KLD}$ between the outputs of the pre-trained model (when evaluated on the *forget* dataset) and the reference model (when evaluated on random inputs) is minimized. This step aims to effectively disrupt the model's retention of the *forget* dataset.

   (b) **Repair Step**

   The pre-trained model is fine-tuned using the data points from the *retain* dataset. This step is intended to restore the model's performance on the retained data while ensuring that the influence of the forgotten class is effectively eliminated.

   The final objective of executing those two steps could be expressed using equation 12.

$$L = \mathcal{KL}(M(x_f) \| M(\hat{z})) + \mathcal{L}(M(x_r), y) \tag{12}$$

By employing these steps, zMuGAN offers a robust framework for machine unlearning that obviates the need for access to the original training data. The combination of GAN-based model inversion and the impair-repair technique ensures that the model can forget specific classes while preserving its overall functionality.

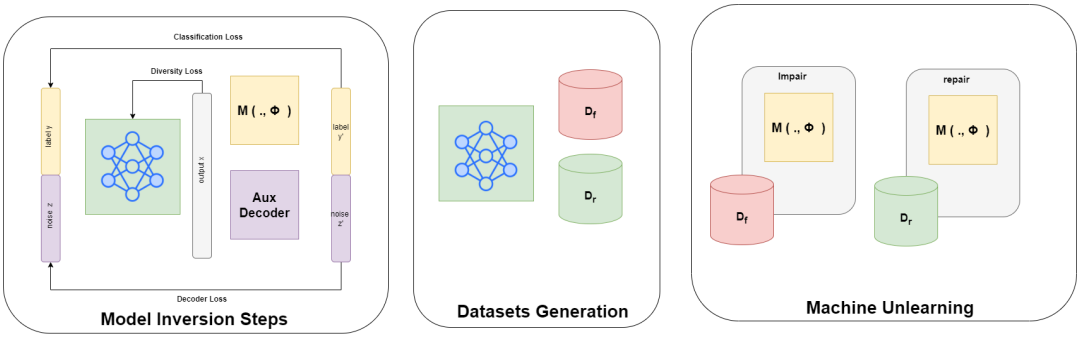


Figure 4: A conceptual diagram of zMUGAN. This figure demonstrates the main steps in zMuGAN. First, GAN-based model inversion from [39] is applied. The produced GANs would be utilized to generate the D_f and the D_r datasets. The predicted label from the model would be the main filtration criteria. Then, those two datasets would be utilized to induce forgetting and retaining.

3.2 MuGAN

The previous technique, zMuGAN, was proposed to address scenarios where access to the training data is totally restricted. However, there are cases where the original dataset was available during the initial training phase. Leveraging this access can significantly enhance the efficiency and effectiveness of the unlearning process. To address these scenarios, we propose MuGAN, a machine unlearning technique designed to utilize the original training dataset to train GANs, which would facilitate precise and effective unlearning.

MuGAN employs a structured approach for machine unlearning, assuming access to the original training dataset during the model's initial training. The method involves utilizing one Generative Adversarial Network (GAN) per split (e.g., user-specific content, class, or sub-class). The architecture of the utilized GAN is domain-dependant – in this work, we mainly utilized DCGANs. Upon receiving a deletion request, the GANs, along with the classifier model—the target of the unlearning algorithm—are used to identify the data points to be used for unlearning. Figure 5 describes the steps of the technique. The process involves generating two primary datasets:

Remain Dataset D_r This dataset includes samples from each class except the target class.

Forget Dataset D_f This dataset includes samples from the target class to be forgotten.

These two datasets are then utilized in the same two-step unlearning process described in zMuGAN.

The primary advantages of zMuGAN and MuGAN stem from their ability to facilitate machine unlearning under varying conditions of data accessibility effectively. zMuGAN is particularly adept in scenarios where access to the original training data is restricted. By utilizing GAN-based model inversion and the proposed impair-repair unlearning process, zMuGAN can approximate and execute unlearning tasks without direct access to the original dataset. This capability ensures compliance with privacy regulations and addresses issues related to data unavailability. MuGAN capitalizes on the availability of the original dataset during the initial training phase. It uses GANs to capture the original data distribution to conduct targeted unlearning. This approach guarantees high accuracy and efficiency in removing specific data points or classes while preserving the model's overall integrity and performance.

Algorithm 5 zMuGAN Algorithm

Require: Model $model$
Require: Target Forget Class $target\_forget\_class$
Require: Learning Rate $learning\_rate$
Require: Dataset Size $dataset\_size$

$GAN \leftarrow$ ApplyModelInversion($model$)
$GAN\_output\_list \leftarrow$ GenerateGanOutput($GAN, dataset\_size$)
$D_f \leftarrow$ initEmptyList()
$D_r \leftarrow$ initEmptyList()
for sample in GAN_output_list **do**
    $predicted\_class\_id \leftarrow model$.predict($sample$)
    if $predicted\_class\_id == target\_forget\_class$ **then**
        D_f.append(($sample, predicted\_class\_id$))
    else
        D_r.append(($sample, predicted\_class\_id$))
    end if
end for
shuffle(D_r)
shuffle(D_f)
$model\_f \leftarrow$ unlearn($model, D_f, D_r, learning\_rate$)
return $model\_f$

Algorithm 6 Model Inversion Algorithm Based on [39]

Require: Model $model$,
Require: Parameter β
Require: Parameter α
$ClassifierNetwork \leftarrow$ InitClassifier($model$)
$GeneratorNetwork \leftarrow$ InitGenerator()
$DecoderNetwork \leftarrow$ InitDecoder()

$classification\_loss \leftarrow$ InitLoss(method='KLD')
$decoder\_loss \leftarrow$ InitLoss(method='L2')
$diversity\_loss \leftarrow$ InitLoss(metric='L2')
for $batch$ in $num\_batches$ **do**
    $noises \leftarrow$ GenerateNoise()
    $labels \leftarrow$ GenerateLabels()

    $images \leftarrow GeneratorNetwork(labels, noises)$
    $outputs \leftarrow ClassifierNetwork(images)$

    $loss1 \leftarrow classification\_loss(outputs, labels)$
    $loss2 \leftarrow decoder\_loss(DecoderNetwork(images), noises) \times \alpha$
    $loss3 \leftarrow diversity\_loss(noises, images) \times \beta$
    $total\_loss \leftarrow loss1 + loss2 + loss3$

    minimize($total\_loss$)
end for
return $GeneratorNetwork$

Algorithm 7 MuGAN Algorithm

Require: Model $model$
Require: Pretrained GANs $pretrained\_GANs$
Require: Target Forget Class $target\_forget\_class$
Require: Learning Rate $learning\_rate$
Require: Dataset Size $dataset\_size$
$D_f \leftarrow$ GenerateDataset($pretrained\_GANs$.of($target\_forget\_class$), $dataset\_size$)
$D_r \leftarrow$ initEmptyList()
for class_id in list_all_classIds **do**
    if class_id $\neq$ target_forget_class **then**
        D_r.append(GenerateDataset($pretrained\_GANs$.of($class\_id$)))
    end if
end for
shuffle(D_r)
$model\_f \leftarrow$ unlearn($model, D_f, D_r, learning\_rate$)
return $model\_f$

Algorithm 8 Unlearn Algorithm

Require: Model $model$
Require: Forgetting Dataset D_f
Require: Retaining Dataset D_r
Require: Learning Rate $learning\_rate$

$criterion \leftarrow$ KlDivergence()
$ref\_model \leftarrow$ Copy($model$)
$ref\_model.freeze()$
$optimizer \leftarrow$ AdamOptimizer($model.parameters(), lr = learning\_rate$)
for batch in D_f **do**
    $images, \_ \leftarrow batch$
    $random\_input \leftarrow$ RandGenerator($images.shape$)
    $out\_r \leftarrow ref\_model(random\_input)$
    $out \leftarrow model(images)$
    $loss \leftarrow criterion(out, out\_r)$
    minimize($loss$)
end for
for retain_batch in D_r **do**
    $loss \leftarrow model.fit(retain\_batch)$ ▷ returns cross-entropy loss
    minimize($loss$)
end for
return $model$

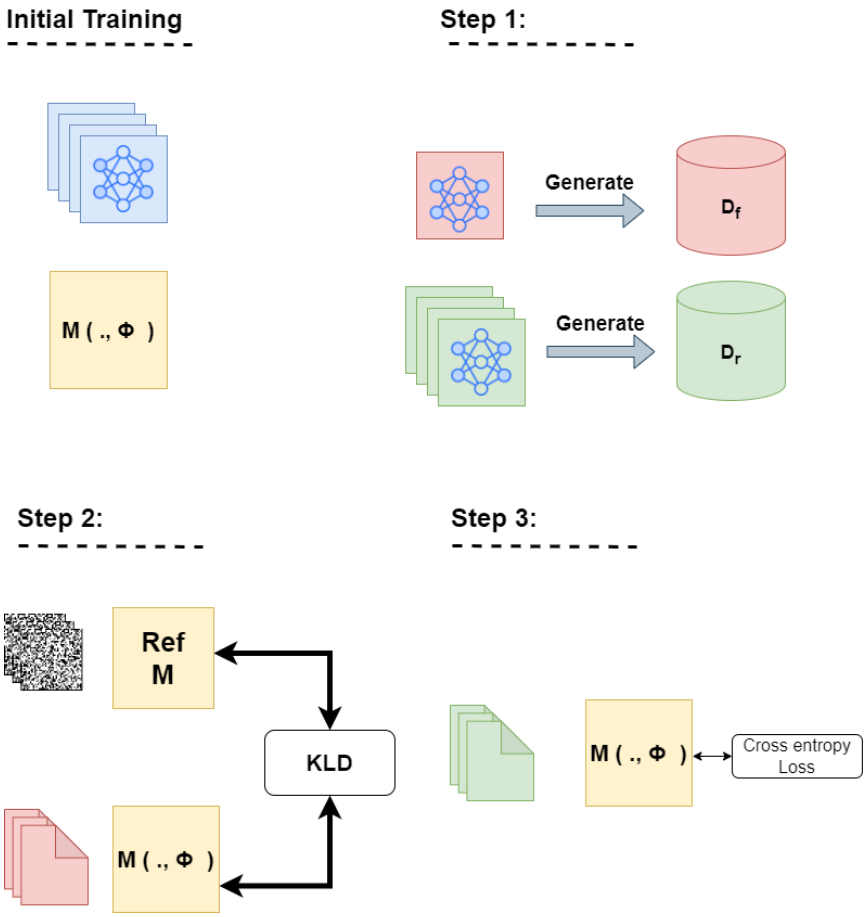


Figure 5: A conceptual diagram of MUGAN. This technique consists mainly of 4 steps. The first step takes place during the initial training of the model. One GAN would be trained per data split (For example, one GAN per class). Next, upon receiving a deletion request, those GANs would be used to generate the data points later used for *unlearning*. Then, the impair-repair steps described in Alg. 8 are applied.

4 Experiment Setup

4.1 Datasets

The CIFAR Suite and SVHN datasets were selected to evaluate the proposed unlearning algorithm. Those two datasets are heavily used in image classification research and are the main datasets utilized in the field of machine unlearning as well.

CIFAR10 consists of 60,000 of 32x32 RGB images representing 10 classes. There is no overlapping between classes, which makes this dataset a perfect candidate for evaluating class unlearning [24].

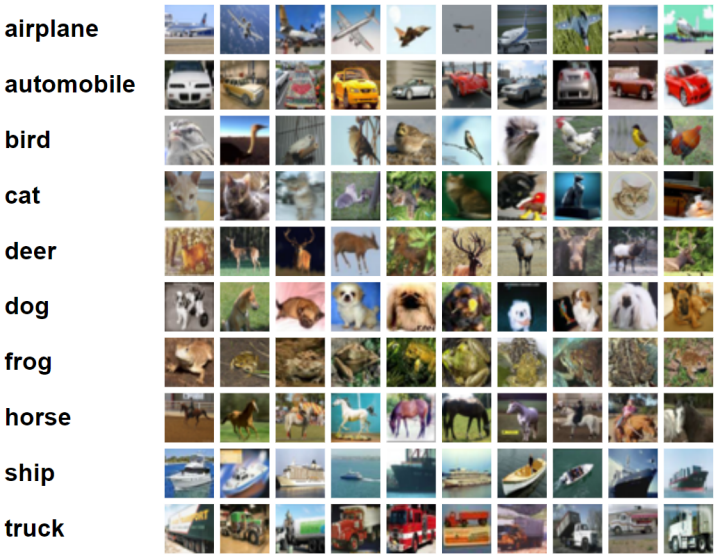


Figure 6: Samples from cifar 10 from [24].

SVHN is a dataset commonly utilized for object recognition originally published in [28]. The dataset contains 600,000 colored digit images and is extracted from the house numbers in Google Street View images. It contains ten classes representing the digits from 0 to 9. This dataset comes in two formats: Full Numbers and Cropped Digits. For our experiments, we utilized the Cropped Digits version.

For both datasets, we utilized the built-in versions that come with torchvision.datasets.

4.2 Environment

All experiments were conducted on an HP Z2 Tower G9 Workstation with an Intel Core i5-12600K, a DDR5 RAM of 32GB, and a NVIDIA RTX A2000 12 GB GDDR6.

Figure 7: Samples from svhn from [28].

4.3 Models

We mainly utilized two models of varying sizes during our experimentation. Namely, We used AllCNN and VGG16. Those models were selected because they are the most commonly used ones for the unlearning task. Further, with their varying sizes, they could represent how the model's architecture could influence the unlearning algorithm's performance, along with the selection of the unlearning hyperparameter.

AllCNN , short for All Convolutional Network, is introduced in [34] with the aim to simplify the common Convolutional neural networks architecture through utilizing Convolutional layers to replace fully connected and using striding to replace pooling. Such changes are believed to preserve spatial hierarchical features. Further, utilizing global average pooling at the end of the network, instead of the typically used fully connected layers, significantly reduces the number of learnable parameters, thus increasing the model efficiency. At the time of its introduction, AllCNN demonstrated very competitive performance on benchmark datasets such as CIFAR-10 and CIFAR-100.

VGG: The VGG architecture is highly influential, as it demonstrated that the depth of the network significantly contributes to its performance [33]. It comes in several variations, with VGG16 and VGG19 being the most notable, denoting the number of

Model	# Trainable params.
AllCNN	1,620,010
VGG16	14,728,266

Table 2: Number of paramters per model.

layers used. This architecture is characterized by its simplicity, mainly utilizing 3x3 convolutional filters, max-pooling layers with a stride of 2, and three fully connected layers at the end.

4.3.1 Models Training

Those models were trained on the CIFAR10 and the SHVN dataset for 30 epochs with a learning rate of 0.001 and an Adam optimizer. The accuracy and the loss of the models are shown in the figures [1(a), 1(b), 2(a), 2(b), 3(a), 3(b), 4(a), 4(b),] in the Appendix.

4.4 Baseline Methods

A number of unlearning algorithms were implemented for a comparative analysis. Although only EMMN and GKT address the same problem constraint as our proposed method, other methods were implemented as well to give a clear overview of the trade-offs between different unlearning algorithms. We utilized the original model and the retrained model as the baseline models. We also evaluate the performance of SCRUB, Bad Teacher, Amnesiac Unlearning, UNSIR, Catastrophic forgetting (finetuning), and Lipschitz (JiT) Unlearning. The implementation of our method and the other baseline methods can be found in our repo at
https://github.com/DataSystemsGroupUT/zero-shot-machine-unlearning

The tables included in the evaluation section list the results after executing each experiment for five trials. The hyper-parameters for most of the methods were selected after conducting a hyper-parameter search using Optuna [2] for 100 trials. However, GKT, based on the implementation of the authors, takes very long to execute, so we used the same parameters that were mentioned in the paper [10]. For MuGAN, we used a learning rate of 1e-4 and a dataset side of 2500 samples per class. For zMuGAN, we used trained five generators through the model inversion process. Each one of them produced 1000 samples, totaling 5000 samples that were later divided into the *forget* and the *retain* datasets. For JiT, we utilized a learning rate of 1e-6, a noise standard deviation of 0.3, and a 100 perpetuated variant per each sample in the forget dataset. For SCRUB, UNSIR, and EMMN, we utilized a learning rate of 5e-5.

4.5 Evaluation Metrics

We utilized the *forget-retain* Accuracy to evaluate the effectiveness of the forgetting algorithm. Further, we also calculated the Anamnesis index to evaluate how long it would take for the model to relearn the *forget* dataset. Also, we implemented the membership inference attack based on utilizing an ensemble of logistic regressions trained on the entropy of the output. Indeed, there are more sophisticated membership inference attack methods; however, this variant is heavily utilized in the unlearning literature. Lastly, the execution time for each method was recorded to compare those methods' efficiency.

5 Evaluation

5.1 Comparative analysis

In tables [6 5, 3, 4], the performance of our proposed techniques is reported in comparison to other state-of-the-art-techniques. Mainly, we selected one class for full-class forgetting. For cifar10, we selected the cat class, and for SVHN, we selected the digit-3 class. Such classes were selected in compliance with other work in the field. Further, Both classes proved challenging for most unlearning algorithms because some features of their content intersect with other classes. For example, cats and dogs are very similar in the CIFAR10 dataset. Further, digits 3,8,0 are very close in the SVHN dataset. Unlearning scores over all classes are also listed in the subsequent subsections. zMuGAN had the best performance in all experiments compared to all other zero-shot algorithms. Further, MuGAN had a very competitive performance compared to SCRUB which requires access to the entire retrain and forget datasets. In our experiments, most techniques resulted in a comparable AIN score, mainly because most of those techniques are under the approximate unlearning category. Our techniques have a number of hyperparameters, such as the number of generated data points used for forgetting, the number of generators used, and the learning rate. In the following section, we will discuss the impact of the size of the generated dataset and the learning rate. However, most notably, using only 2500 data points per class, MuGAN had the shortest execution time of all algorithms. On the other hand, zMuGAN took around 20 minutes to apply model inversion. Subsequently, unlearning using those points took less than 5 seconds. GKT, the direct competing system in the same setup, took more than one hour to run, and it produced sub-optimal results across all datasets and all models.

Table 3: VGG16 Full-class unlearning performance for the cat-class in CIFAR-10:

Method	$D_f ACC.$	$D_r ACC.$	MIA	Time	AIN
BSLN	70.7	87.3	45.15	0.068	0.018
RTRN	0	88.3	54	864.6	-
FNTN	23.2	90.4	41.3	80.2	0.054
SCRUB	8.6	85.5	40.5	21.6	0.017
UNSIR	0	24.6	94.5	45.6	0.133
JiT	35.1 ±0.0	67.4 ±0.0	45.6	55.62	0.017
EMMN	33.8 ±2.4	19.4 ±0.03	35.4	45.8	0.0175
GKT	0.0 ±0.0	8.4 ±0.0	46.58	5000+	0.0175
MUGAN	**0** ±0.0	**82.8** ±1.6	**35.4**	10.458	0.085
Z-MUGAN	0.64 ±.02	53.55 ±1.59	44.1	1260	0.066

Table 4: VGG16 Full-class unlearning performance for digit-3 across SVHN:

Method	$D_f ACC.$	$D_r ACC.$	MIA	Time	AIN
BSLN	89.7	95	27.9	0.082	0.012
RTRN	0	95.3	54.5	1184	-
FNTN	0.5	95.3	32	117.3	0.047
SCRUB	6.6 ±0.1	93.8 ±0.0	21.7	31.784	0.011
UNSIR	3.1 ±0.6	57.8 ±0.5	10.2	72.79	0.073
JiT	37.6 ±0.0	93.8 ±0.0	28	94.25	0.012
GKT	0.0 ±0.0	22.3 ±0.00	32.26	5000+	0.0311
EMMN	58.3 ±1.39	74.2 ±0.30	28.26	41.6	0.0121
MUGAN	**00.0** ±0.0	**94.8** ±0.17	20.1	**10.03**	0.081
Z-MUGAN	00.0 ±1.2	83.8 ±2.24	37.1	1171	0.087

Table 5: ALLCNN Full-class unlearning unlearning performance for the cat-class CIFAR-10

Method	$D_f ACC.$	$D_r ACC.$	MIA	Time	AIN
BSLN	83.6	82.3	12.5	0.012	0.795
RTRN	0	87.4	32.07	730.2	-
FNTN	41.3	90.7	7.6	71.75	0.341
SCRUB	**3.4** ±0.04	**85.1** ±2.801	12.6	20.91	0.340
UNSIR	0.1 ±0.02	37.7 ±0.03	32.8	42.79	0.067
JiT	36 ±0.0	80.4 ±0.0	12.3	52.28	0.09
GKT	0.00 ±0.0	12.11 ±0.0	30.2	5000+	0.2681
EMMN	0.61 ±0.2	11.9±0.6	**12.1**	5.88	0.1881
MUGAN	**5.2** ±0.2	**81.8** ±1.7	13.9	11.582	0.523
Z-MUGAN	3.0 ±1.34	55.3 ±3.2	33.1	1319	0.093

Table 6: ALLCNN Full-class unlearning performance for digit-3 for SVHN

Method	$D_f ACC.$	$D_r ACC.$	MIA	Time	AIN
BSLN	93.2	94.4	15.5	0.012	0.0076
RTRN	0	95.3	22.2	496.2	-
FNTN	45.7	95.3	15.56	103.9	0.085
SCRUB	5.9 ±0.2	93.1 ±0.07	**9.4**	30.97	0.023
UNSIR	42.3 ±0.5	77.3 ±.8	17.6	69.34	0.043
JiT	48.8 ±0.0	93.5 ±0.0	15.6	88.92	0.007
GKT	89.1±0.0	88.2 ±0.0	14.2	5000+	0.003
EMMN	47.2 ±0.33	69 ±1.23	15.09	5.99	0.00746
MUGAN	**0** ±0.0	**93.6** ±0.6	15.6	10.56	0.142
Z-MUGAN	14 ±2.35	84.6 ±3.20	18.19	1244	0.0174

5.2 MuGAN: Effect of dataset size

IN MuGAN, GANs are mainly utilized to generate the retain and the forget datasets used during the unlearning algorithm. In tables [7 ,8], we compare the performance of MuGAN using a different number of data points per class. Notably, the proposed technique is effective with as few as 300 points. In most experiments, the forgetting effectiveness increases as we increase the data points. However, after a certain threshold, depending on the model and the dataset, the model experiences over-forgetting, which is reflected in the decreasing D_r accuracy and the increasing MIA score. In tables [9, 10], we showed the impact of the generated dataset size on the performance of zMuGAN. Here, zMuGAN doesn't offer the same level of control over the number of samples generated per class. This is mainly due to the stochastic nature of model inversion. In our work, an ensemble of generators was utilized to increase the diversity of the generated output. Following the default recommendations in [39], our experiments utilized five generators. The hyper-parameter listed here is the number of points generated by each GAN.

# of ins.	$D_f ACC.$	$D_r ACC$	MIA
100	20.2	93.1	9.9
300	00.00	95.2	**0.79**
500	00.00	95.2	4.57
1000	**00.00**	**95.4**	9.5
2500	00.00	94.8	14.1
5000	00.00	94.2	23.1
10000	00.00	93.5	35.5

Table 7: Effect of dataset size: MuGAN VGG16 SVHN

# of ins.	$D_f ACC.$	$D_r ACC$	MIA
100	01.8	87.7	10.53
300	11.1	92.6	03.61
500	20.2	93.2	**02.59**
1000	**00.00**	**93.4**	02.70
2500	00.00	92.7	07.79
5000	00.10	89.1	13.30
10000	00.00	87.9	16.59

Table 8: Effect of dataset size: MuGAN ALLCNN SVHN

# of ins.	$D_f ACC.$	$D_r ACC$	MIA
100	95.41	90.84	96.50
200	95.24	86.60	94.56
300	92.88	81.115	84.11
500	13.41	84.941	**0.3**
1000	**00.00**	**89.67**	1.03
2000	52.30	91.23	19.44

Table 9: Effect of dataset size: zMuGAN VGG16 SVHN

# of ins.	$D_f ACC.$	$D_r ACC$	MIA
100	83.38	88.81	78.94
200	73.27	81.7	71.60
300	56.82	75.19	57.57
500	20.43	61.55	39.34
1000	07.97	49.20	28.92
2000	**2.23**	**48.62**	**25.28**

Table 10: Effect of dataset size: zMuGAN ALLCNN SVHN

5.3 Effect of Learning Rate

Most machine unlearning algorithms are highly sensitive to the value of the learning rate. Having a higher learning rate would result in over-forgetting and possibly compromise the model's utility. One the other hand, having a lower learning rate would render the unlearning algorithm ineffective. In tables [11, 12], We list the performance of MuGAN depending on the learning rate. It should be noted that the selection of the appropriate learning rate (LR) depends on the selected model and the original training dataset. Further, the trade-off between privacy, as demonstrated in the MIA score, and utility, as demonstrated in the D_f and D_r accuracy, is evident. A higher learning rate results in over-forgetting and, consequently, a high MIA score. That is, the model is susceptible to membership inference attacks. In tables [13 , 14], we listed the impact of LR on the performance of zMuGAN. The results show a similar pattern. Having a lower learning rate would reduce the effect of the unlearning algorithm; on the other hand, having a learning rate higher than needed would have either a diminishing or negative impact.

LR	$D_f Acc.$	$D_r Acc.$	MIA
1.00E-08	95.60	93.30	15.32
1.00E-07	95.40	93.20	15.33
1.00E-06	95.50	93.10	15.16
1.00E-05	95.50	93.20	13.83
1.00E-04	93.00	93.70	7.39
1.00E-03	76.10	95.10	**1.32**
1.00E-02	**0.00**	**85.10**	14.10
1.00E-01	0.00	11.10	0.00

Table 11: Effect of Learning Rate: MuGAN ALLCNN SVHN

LR	$D_f Acc.$	$D_r Acc.$	MIA
1.00E-08	94.00	94.50	28.07
1.00E-07	93.70	94.50	28.03
1.00E-06	93.90	94.40	27.47
1.00E-05	93.40	94.70	21.39
1.00E-04	85.30	95.50	**3.64**
1.00E-03	**0.00**	**94.40**	15.03
1.00E-02	0.00	72.70	52.74
1.00E-01	0.00	11.10	0.00

Table 12: Effect of Learning Rate: MuGAN VGG16 SVHN

LR	$D_f Acc.$	$D_r Acc.$	MIA
1.00E-08	73.45	80.48	71.38
1.00E-07	72.87	80.4	70.97
1.00E-06	65.08	77.78	65.66
1.00E-05	**00.76**	**48.92**	29.09
1.00E-04	49.71	50.23	36.98
1.00E-03	00.00	12.09	**11.62**
1.00E-02	00.00	15.20	33.48
1.00E-01	00.00	10.91	77.28

Table 13: Effect of Learning Rate: zMuGAN ALLCNN SVHN

LR	$D_f Acc.$	$D_r Acc.$	MIA
1.00E-08	92.60	89.80	93.23
1.00E-07	92.00	89.20	93.41
1.00E-06	92.08	89.20	92.66
1.00E-05	86.29	86.94	80.07
1.00E-04	**00.00**	**89.80**	**15.56**
1.00E-03	32.97	62.81	46.72
1.00E-02	00.00	12.06	36.54
1.00E-01	00.00	11.11	45.54

Table 14: Effect of Learning Rate: zMuGAN VGG16 SVHN

5.4 Unlearning Performance for different classes

In tables [6 5, 3, 4], adhering to the common practice in most machine unlearning literature, we listed the performance of all selected methods on forgetting on selected class — for CIFAR-10, it was the ***Cat*** class, and for SVHN, it was the digit ***3***. However, during our experiments, we realized that unlearning algorithms don't perform with the same effectiveness across all classes. This is mainly due to the models' inherent confusion. In figures [8, 10, 9], the performance of SCRUB, Z-MuGAN, and MuGAN across different classes was reported. On the y-axis, we have the target forget label, and on the x-axis, the per-class accuracy is plotted. In the case of retraining, we will have completely dark cells across the left diagonal. Compared to SCRUB, MuGAN has the best average performance across all classes. This stems from its design philosophy. Two filtration techniques are applied to make sure that only the information related to the target class is forgotten: first, GANs are trained per class split. Second, the output of the GANs is further filtered using the model to make sure that the output is highly associated with the target forget class. In the case of zMuGAN, only the second filtration step is applied. Although not as surgical as MuGAN, zMuGAN demonstrates high effectiveness in forgetting 6 out of 10 classes. The limitations of the technique will be further analyzed in the discussion section.

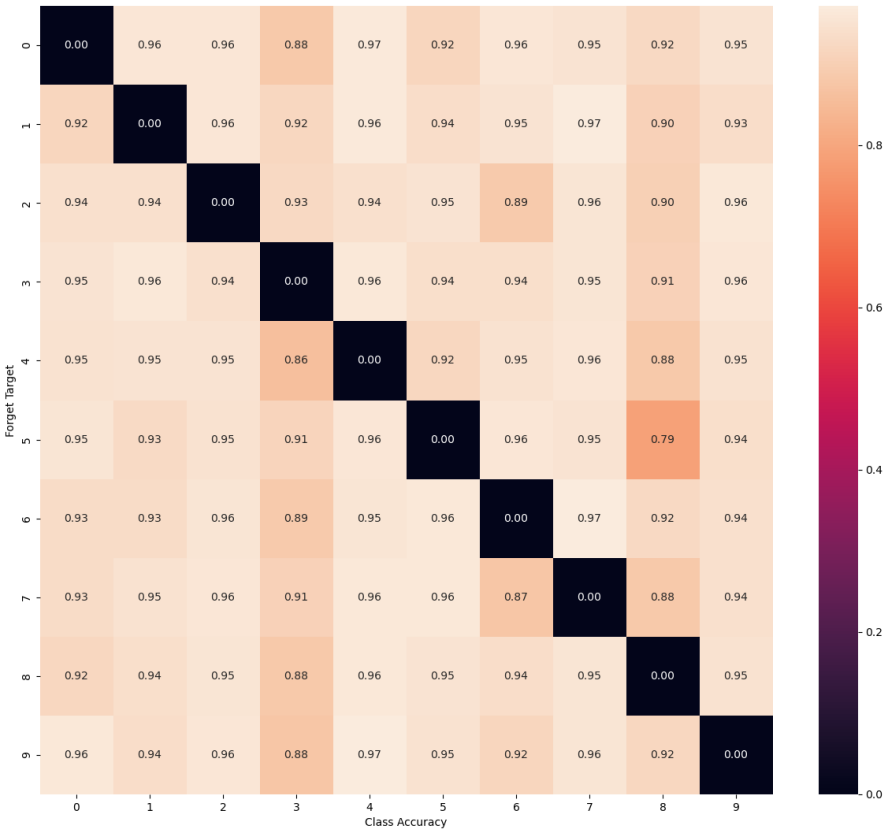


Figure 8: Performance of MuGAN for different forget targets

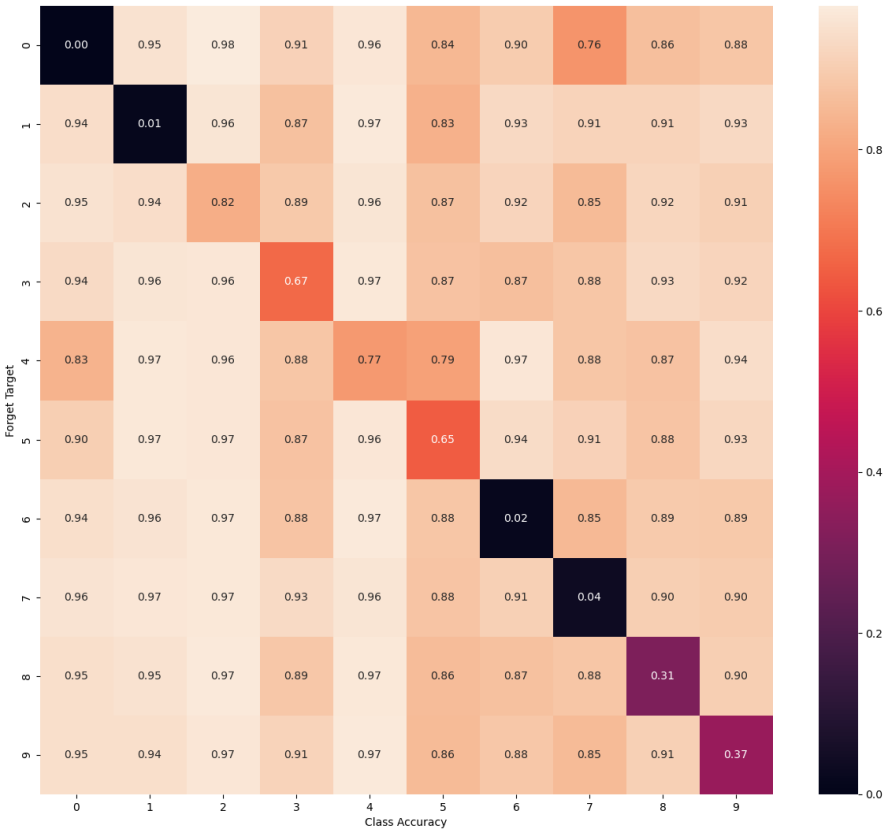


Figure 9: Performance of zMuGAN for different forget targets

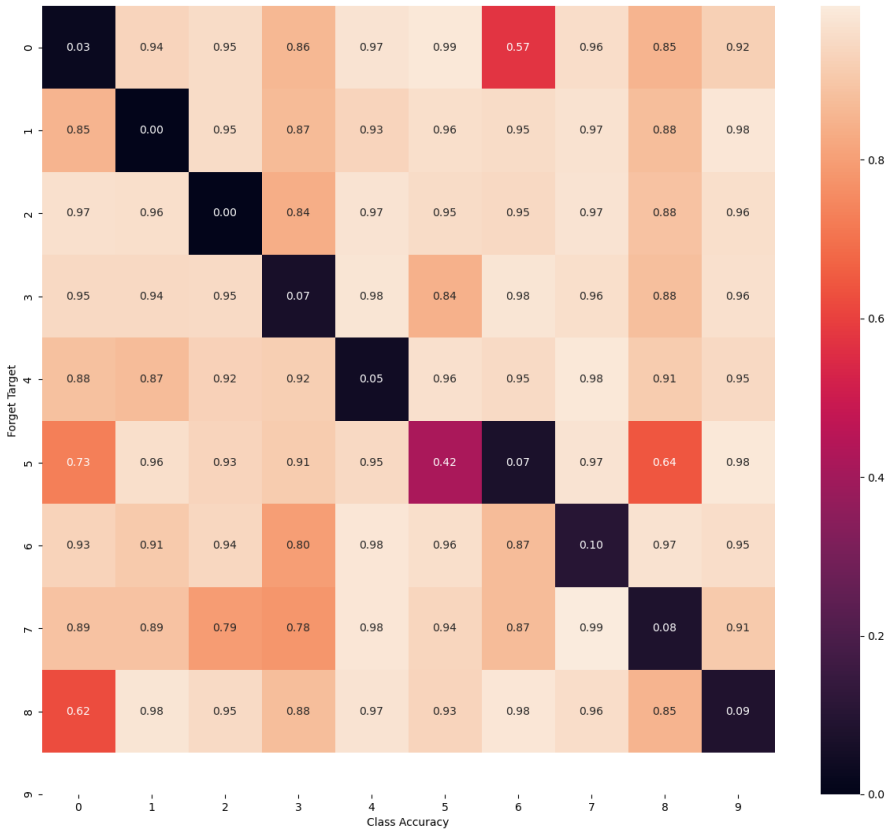


Figure 10: Performance of SCRUB for different forget targets

6 Discussion and Limitation

In this work, we address the challenge of machine unlearning under limited access to training data through the introduction of two techniques: zMuGAN and MuGAN. zMuGAN employs GAN-based model inversion to extract data points for unlearning from the trained model, enabling it to operate out-of-the-box and be readily applied to already deployed models. Conversely, MuGAN leverages access to the training dataset during initial model training to enhance unlearning performance, integrating unlearning capabilities into the early stages of model development.

The design of machine unlearning algorithms necessitates careful consideration of the trade-offs between utility, effectiveness, and efficiency. Our proposed techniques offer varying balances between these objectives, providing versatility and adaptability to different scenarios. zMuGAN excels in effectiveness and utility but is less time-efficient. Compared to other zero-shot learning algorithms, such as GTK or EMMN, which often significantly compromise model utility, zMuGAN maintains high performance across diverse models and datasets. However, the model inversion process is a potential bottleneck due to the extreme condition of having no access to observable training data.

MuGAN, on the other hand, demonstrates high effectiveness across utility, effectiveness, and efficiency. Nevertheless, modifications to the model development process necessitate additional storage costs for the weights of the utilized GANs. These characteristics highlight the strengths and limitations of each technique, making them suitable for different unlearning scenarios based on the specific requirements of utility, effectiveness, and efficiency.

6.1 Future Work

The field of machine unlearning remains in its infancy. While the techniques proposed in this work demonstrate efficacy in zero-shot and one-glance scenarios, there is considerable scope for improvement. This research has shown the feasibility of performing effective unlearning without access to data at the time of the deletion request. However, the following points could be addressed to improve the performance of the proposed methods further.

Firstly, during experimentation, we employed DCGAN with the vanilla GAN loss function, which presented several drawbacks. Notably, this approach required multiple training iterations due to the propensity to collapse. Furthermore, the output of GANs often lacked diversity, resulting in the model learning only a single representation of the class. While zMuGAN addressed this issue by incorporating the diversity loss proposed by KegNet (as cited in [39]), there remains room for performance enhancement.

Additionally, the output quality of the model inversion process could be improved

by utilizing a few samples from the same distribution and domain. As demonstrated in [40], incorporating such samples allows for augmenting the loss calculation to include the distance between the GAN output and samples from the original distribution, thereby enhancing the model's output.

Lastly, the zero-shot setup assumes no access to training data. However, most evaluation metrics currently rely on training data to assess the quality of forgetting and to select the optimal checkpoint demonstrating the desired performance. To address this challenge, we propose generating a synthetic dataset using GANs trained initially in MuGAN or derived from the model inversion technique. These synthetic data points could be evaluated on the original model, providing a benchmark to assess the output of the unlearning algorithm based on selected hyperparameters.

Future research should focus on these areas to enhance the robustness and applicability of machine unlearning techniques, ensuring they meet the evolving requirements of data privacy and model performance.

7 Conclusion

In conclusion, this thesis addresses the vital challenge of machine unlearning, an essential process in the context of modern AI and machine learning applications. With the rapid expansion of AI technologies and the increasing importance of data privacy regulations, such as GDPR, the need for efficient and practical unlearning methods has become paramount. Traditional methods, like retraining models from scratch, are resource-intensive and often impractical, underscoring the necessity for innovative approaches.

This research introduces two novel techniques, MuGAN and zMuGAN, designed to facilitate machine unlearning under different data access conditions. MuGAN is tailored for scenarios where limited access to the training dataset is available. By utilizing Generative Adversarial Networks (GANs), MuGAN effectively generates synthetic data that mirrors the original data distribution, enabling precise unlearning while preserving the model's integrity. zMuGAN, on the other hand, is crafted for situations where no access to the training data exists. It leverages a GAN-based model inversion technique to reconstruct the training data and implements an impair-repair process to remove specific data points or classes from the model. This method ensures compliance with privacy mandates and addresses issues of data unavailability.

The efficacy of these techniques is demonstrated through their application to image classification tasks, focusing on the critical task of class forgetting. By addressing both class and sub-class forgetting, MuGAN and zMuGAN show their robustness and versatility, making them suitable for a wide range of AI applications.

In summary, the contributions of this thesis provide significant advancements in the field of machine unlearning. The proposed techniques offer scalable and efficient solutions that adapt to various stages of model development and deployment. By enabling effective unlearning, this research supports the responsible use of AI technologies, ensuring that models can comply with evolving data privacy requirements while maintaining high performance. This work sets the stage for further innovation in unlearning methods, promoting the ethical and secure deployment of machine learning models.

References

[1] Neurips 2023 machine unlearning challenge, 2023. Accessed: 2024-05-15.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

[5] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE, 2015.

[6] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[7] Simon Caton and Christian Haas. Fairness in machine learning: A survey, 2020.

[8] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 896–911. ACM, 2021.

[9] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher, 2023.

[10] Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan Kankanhalli. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 18:2345–2354, 2023.

[11] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms, 2017.

[12] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[13] Jack Foster, Kyle Fogarty, Stefan Schoepf, Cengiz Öztireli, and Alexandra Brintrup. Zero-shot machine unlearning at scale via lipschitz regularization, 2024.

[14] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 792–801, 2021.

[15] Aditya Golatkar, Alessandro Achille, and Stefano 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*, pages 9304–9312, 2020.

[16] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 383–398. Springer, 2020.

[17] Eric Goldman. An introduction to the california consumer privacy act (ccpa). *Santa Clara Univ. Legal Studies Research Paper*, 2020.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[20] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning, 2020.

[21] Chris Jay Hoofnagle, Bart Van Der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[23] Myeongsu Kang and Noel Jordan Jameson. Machine learning fundamentals. In *Prognostics and Health Management in Electronics: Fundamentals, Machine Learning, and Internet of Things*. Wiley Online Library, 2018.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

[25] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning, 2023.

[26] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 253–256, 2010.

[27] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[28] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[29] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

[30] Jeffrey Rosen. The right to be forgotten. *Stan. L. Rev. Online*, 64:88, 2011.

[31] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models, 2018.

[32] Sebastian Schelter. "amnesia" - towards machine learning models that can forget user data very fast. In *10th Conference on Innovative Data Systems Research (CIDR)*, Amsterdam, The Netherlands, January 12–15 2020. Online Proceedings, www.cidrdb.org.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015.

[34] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2015.

[35] Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–10, 2024.

[36] Yaxing Wang, Lichao Zhang, and Joost van de Weijer. Ensembles of generative adversarial networks, 2016.

[37] Zhuo Wang, Kai Wei, Chunyu Jiang, Jiafeng Tian, Minjing Zhong, Yuan Liu, and Yanmei Liu. Research on productization and development trend of data desensitization technology. In *20th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1564–1569, Shenyang, China, October 20–22 2021. IEEE, IEEE.

[38] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, 2016.

[39] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[40] Youngsik Yoon, Jinhwan Nam, Hyojeong Yun, Jaeho Lee, Dongwoo Kim, and Jungseul Ok. Few-shot unlearning by model inversion, 2023.

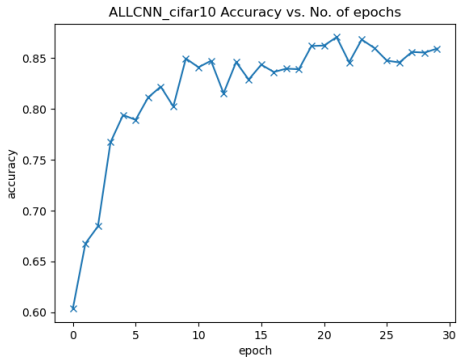
[41] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning, 2017.

[42] Xiaoyong Yuan and Lan Zhang. Membership inference attacks and defenses in neural network pruning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4561–4578, Boston, MA, August 2022. USENIX Association.

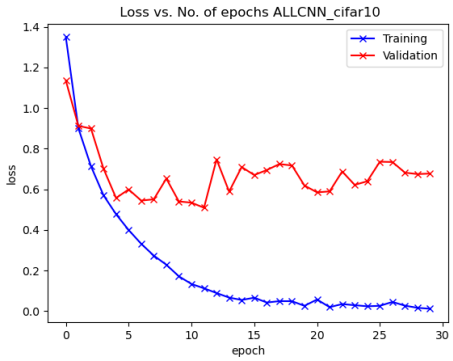
[43] Wayne Xin Zhao, Ke Zhou, Jing Li, Tianyi Tang, Xiaolei Wang, Yao Hou, Yichen Min, Bo Zhang, Jing Zhang, Zongheng Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Appendix

A Accuracy and Loss for Selected Models

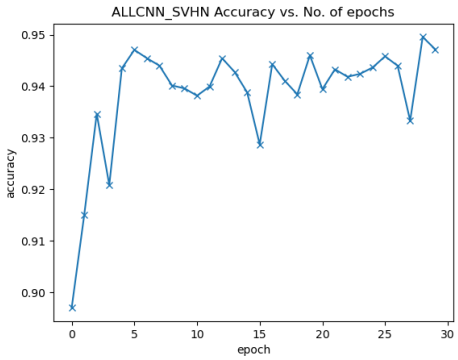


((a)) Training and Validation Acc.

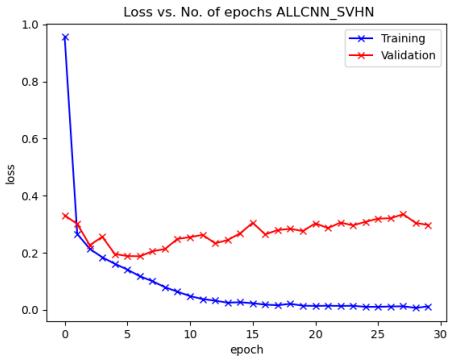


((b)) Loss

Figure A1: Training Stats. AllCNN-CIFAR10

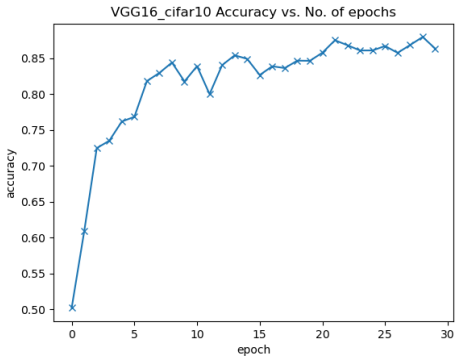


((a)) Training and Validation Acc.

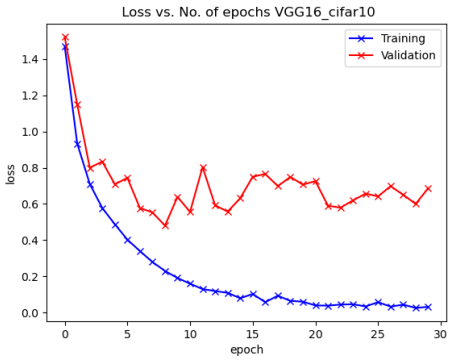


((b)) Loss

Figure A2: Training Stats. AllCNN-SVHN

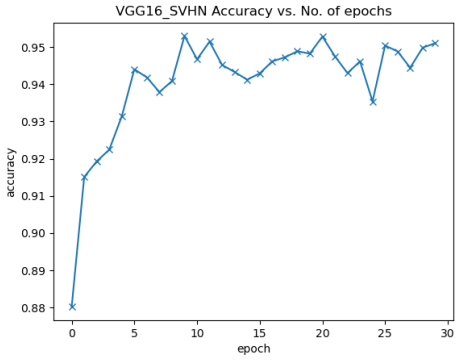


((a)) Training and Validation Acc.

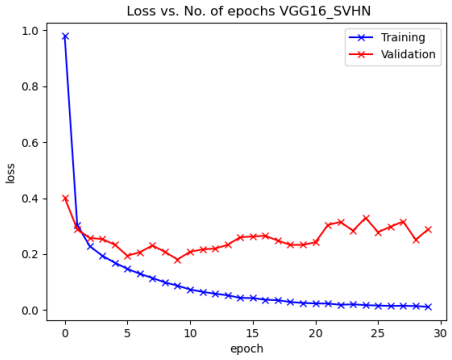


((b)) Loss

Figure A3: Training Stats. VGG16-CIFAR10



((a)) Training and Validation Acc.



((b)) Loss

Figure A4: Training Stats. VGG16-SVHN