AUGMENTED MIXUP PROCEDURE FOR PRIVACY-PRESERVING COLLABORATIVE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixup, introduced by Zhang et al., is a regularization technique for training neural networks that generates convex combinations of input samples and their corresponding labels. Motivated by this approach, Huang et al. proposed InstaHide, an image encryption method designed to preserve the discriminative properties of data while protecting original information during collaborative training across multiple parties. However, recent studies by Carlini et al., Luo et al., and Chen et al. have demonstrated that attacks exploiting the linear system generated by the mixup procedure can compromise the security guarantees of InstaHide. To address this vulnerability, we propose a modified mixing procedure that introduces perturbations into samples before forming convex combinations, making the associated linear inverse problem ill-conditioned for adversaries. We present a theoretical worst-case security analysis and empirically evaluate the performance of our method in mitigating such attacks. Our results indicate that robust attack mitigation can be achieved by increasing the perturbation level, without causing a significant reduction in classification accuracy. Furthermore, we compare the performance of our approach with that of InstaHide on standard benchmark datasets, including MNIST, CIFAR-10, and CIFAR-100.

1 INTRODUCTION

Data mixing was initially introduced as a dataset augmentation technique, generating new samples by computing weighted averages of subsets from the original dataset Zhang et al. (2017). Originally designed as a regularization method for training neural networks, this approach has also been adapted for privacy-preserving protocols, as the mixing process obscures the original data during model training Liu et al. (2019); Fu et al. (2019).

Although the mixup strategy appears to preserve privacy without significantly degrading model performance, directly applying the method proposed by Zhang et al. (2017) can introduce vulnerabilities that allow attackers to recover private data under certain conditions Huang et al. (2020):

1. **Mixup samples from a private dataset only:** If mixup samples are generated exclusively from a private dataset, an attacker can identify which samples share a common private component by analyzing the expected value of the dot product between mixup samples. Once a set of related mixup samples is identified, the common private sample can be reconstructed by averaging these samples.
2. **Mixup samples from both private and public datasets:** When mixup samples are generated using both private and public datasets, repetitions of private samples in the mixup process can be avoided by leveraging the public dataset. However, since the public dataset is accessible, an attacker can perform a similar statistical analysis to identify which public samples were used in the mixup. Once the public components are determined, the private sample can be trivially reconstructed.

Building on their security analysis, Huang et al. (2020) proposed a mixup-based algorithm called InstaHide. The key innovation of their method is the application of a sign-flipping mask to images generated by computing a weighted sum of both public and private samples. The authors analyzed the security of InstaHide and formally proved that its security depends on the computational hardness of the subset-sum problem. However, the assumptions underlying their security model do not

accurately reflect the properties of real-world data. In particular, Huang et al. (2020) assumed that each sample consists of an arbitrary sequence of values. For example, in the context of images, this assumption implies that pixels are independently and randomly distributed, which does not hold in practice. Consequently, although the security proof is mathematically valid, it does not offer practical security guarantees. This limitation was highlighted by Carlini et al. (2021a), who developed efficient attacks on samples generated by the InstaHide algorithm, enabling near-complete recovery of the original data.

2 CONTRIBUTIONS

The primary contribution of this paper, presented in Section 3, is a mixup algorithm designed to address a key vulnerability exploited by all major attacks on InstaHide, the repeated use of the same private sample across multiple mixup operations. In Section 4, we analyze three prominent attacks targeting InstaHide and evaluate their effectiveness against our proposed method. Furthermore, we introduce a theorem that formalizes the security guarantees of our algorithm under the assumption that the data follows an isotropic Gaussian distribution.

Experimental results, reported in Section 5, show that our approach maintains classification accuracy within 5% of InstaHide's performance on three benchmark datasets: MNIST, CIFAR-10, and CIFAR-100. We also provide empirical evidence quantifying the extent to which an adversary can reconstruct original samples from the mixed data and mixing weights, across various parameter settings of our algorithm. These results demonstrate that our mixup procedure effectively protects the original data while incurring only a minimal reduction in accuracy.

3 SINGULARIZED MIXUP

In this section, we introduce our algorithm, which is based on the singularization framework. This approach enhances security by ensuring that each execution is unique, thereby making it broadly applicable to a variety of systems Gaber et al. (2023). Singularization has previously been used to strengthen encryption algorithms without modifying their underlying structure Macario-Rat & Plesa (2024), which motivates our adoption of this framework in the design of our mixup algorithm.

We begin with a brief overview of InstaHide, followed by a detailed description of our proposed algorithm.

3.1 INSTAHIDE ALGORITHM

Consider a private dataset $(x_i, y_i)_{i=1}^n$ consisting of n samples, where $x_i \in \mathbb{R}^d$ denotes the input example and $y_i \in \mathbb{R}^c$ is the corresponding one-hot encoded label.

The fundamental idea behind Mixup, as introduced by Zhang et al. (2017), is to replace each data point with a convex combination of the current sample and $k-1$ other samples selected uniformly at random from the dataset. Specifically, each new data point is generated by taking a weighted average of k instances and their associated labels:

$$\tilde{x_i} \leftarrow w_{i1} x_i + \sum_{j=2}^{k-1} w_{ij} x_{\pi_i(j)} \tag{1}$$

$$\tilde{y_i} \leftarrow w_{i1} y_i + \sum_{j=2}^{k-1} w_{ij} y_{\pi_i(j)} \tag{2}$$

where $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^n$ represents the encoded dataset and π_i is a random permutation over $\{1, 2, \ldots, n\}$.

The InstaHide approach builds upon Mixup but introduces two key modifications:

1. **Public images**: InstaHide augments the private dataset with samples from public datasets, expanding the pool of mixing samples to $(x_i, y_i)_{i=1}^{n} \cup (x_i, y_i)_{i=n+1}^{n+m}$, where m denotes the size of the public dataset.
2. **Sign mask**: The sign of each pixel in a mixup image is randomly flipped using a random sign mask $\sigma_i \sim \Lambda_{\pm}^{d}$.

As a result equations (1) and (2) are modified as follows:

$$\tilde{x_i} \leftarrow \sigma_i \circ \left(w_{i1} x_i + \sum_{j=2}^{k_s - 1} w_{ij} x_{\pi_i(j)} + \sum_{j=k_s+1}^{k-k_t} w_{ij} x_{\pi_{i_p}(j)} \right) \tag{3}$$

$$\tilde{y_i} \leftarrow w_{i1} y_i + \sum_{j=2}^{k_s - 1} w_{ij} y_{\pi_i(j)} \tag{4}$$

Here, k_s denotes the number of private images, k_t the number of public images, and π_{i_p} is a random permutation over the set $\{n+1, \ldots, n+m\}$. Note that public images are used solely as a source of structured noise, and their labels are not included in the mix.

The InstaHide mixing procedure can also be expressed using linear operators. Let X be the $n \times d$ matrix of original samples, where the i-th row of X is x_i; let $\tilde{X}$ be the $n \times d$ matrix of mixup samples, where the i-th row of $\tilde{X}$ is $\tilde{x}_i$; let W be the $n \times n$ matrix of weights, where the i-th row of W is w_i; and let E be the $n \times d$ noise matrix, where the i-th row of E is given by $\sum_{j=k_s+1}^{k-k_t} w_{ij} x_{\pi_{i_p}(j)}$.

The InstaHide process can then be written as:

$$\tilde{X} = \sigma \left(WX + E \right) \tag{5}$$

This formulation has important implications for security analysis. Specifically, recovering the private images, up to the sign of each pixel, given the encoded samples and the mixing weights, reduces to solving a linear system of equations with added noise.

3.2 SINGULARIZATION ALGORITHM

The principal aim of the singularization algorithm is to transform the original dataset $\{(x_i, y_i)\}_{i=1}^{n}$ into a new set $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n}$ such that the resulting dataset preserves the discriminative characteristics of the original data, while ensuring that the original data cannot be recovered.

A key vulnerability of the InstaHide algorithm arises from the possibility that two encoded samples may share the same original input during the mixup process. This issue is inherent to the mixup strategy underlying InstaHide: the algorithm generates an encoded dataset of the same size as the original, with each encoded sample constructed as a convex combination of k_s samples selected from the original dataset. Even when public data is included, InstaHide often incorporates multiple private samples in each combination. This enables an attacker to cluster encoded samples based on shared private components and subsequently reconstruct the original private image from each cluster. In principle, this recovery is feasible because, from the attacker's perspective, the shared sample appears as a consistent signal, while the other components act as noise that can be filtered out.

In our method, each encoded input is constructed to include exactly one private data point, while the remaining $k - 1$ components consist of noise added prior to the mixup operation. Introducing noise before the weighted sum makes it significantly more challenging for an attacker to filter out, and if the noise level is sufficiently high, inverting the process becomes an ill-conditioned problem. Unlike InstaHide, our approach does not require public data or the use of sign-flipping masks. Previous attacks Carlini et al. (2021a); Chen et al. (2020); Luo et al. (2022) have demonstrated that sign-flipping masks can be circumvented by analyzing the absolute values of the mixed images, and that public images act as noise, which tends to average out when multiple mixup samples are examined.

Our singularization algorithm is presented in Algorithm 1.

Algorithm 1 Singularized Mixup

Require: Dataset $\{(x_i, y_i)\}_{i=1}^n$; mixing parameter k; error norm r
Ensure: Mixed dataset $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^n$

```
π_j ~ Uniform(S_n), j = {2, 3, ..., k}          //Sample k − 1 random permutations
for each i = 1 to n do
    (w_i1, ..., w_ik) ~ Dirichlet(1, 1, ..., 1)          //Sample mixing coefficients
    x̃_i ← w_i1 x_i
    ỹ_i ← w_i1 y_i
    for each j = 2 to k do
        e_ij ~ Uniform(B(0, r))          //Sample a noise vector with norm at most r
        x̃_i ← x̃_i + w_ij (x_{π_j(i)} + e_ij)
        ỹ_i ← ỹ_i + w_ij y_{π_j(i)}
    end for
end for
return {(x̃_i, ỹ_i)}_{i=1}^n
```

Similarly to (5), Algorithm 1 can also be expressed in matrix form. The matrices X, $\tilde{X}$, and W retain the same definitions as in (5); however, in our case, the noise matrix E is not constructed from public images but is instead defined by noise vectors. Specifically, the i-th row of E is given by $\sum_{j=2}^{k} w_{ij} e_{ij}$. We have:

$$\tilde{X} = WX + E \tag{6}$$

It is important to note that, in our algorithm, noise is added to each private sample x_i prior to computing the mixup result $\tilde{x}$. Specifically, the mixing coefficient applied to $x_{\pi_j(i)}$ is identical to the coefficient applied to the corresponding noise term e_{ij}. This differs from the approach described in (3), where the weights assigned to public images are distinct from those associated with private images. In our configuration, this coupling between the noise and the private data makes inverting the algorithm an ill-conditioned problem.

3.3 Practical Instantiation

Similar to InstaHide Huang et al. (2020), the primary application of our algorithm is in privacy-preserving collaborative training. Suppose there are multiple parties, each possessing a private local dataset. These parties aim to jointly train a deep neural network on the combined data without exposing the sensitive information contained in their individual datasets. The following general framework demonstrates how Algorithm 1 can be integrated to achieve this goal:

1. All parties agree on a common preprocessing technique to be applied locally. For instance, in the context of image data, participants may choose to standardize the images or extract feature representations using a publicly available pretrained model, such as ResNet He et al. (2016).

2. Each party independently transforms its local dataset by applying Algorithm 1, thereby generating a set of mixup samples. Each sample consists of a mixup example and its corresponding mixup label.

3. The resulting data is then transmitted to a central server, which is responsible for training the deep learning model. Upon completion, the trained model is distributed back to the parties for local use.

The security of this protocol depends on the effectiveness of Algorithm 1 in protecting the privacy of local datasets. In particular, the central server, which only receives the mixup samples, should not be able to reconstruct the original data from these representations.

4 SECURITY ANALYSIS

In this section, we analyze the security of the proposed scheme and examine its resilience against three major attacks on the InstaHide system, as described in Carlini et al. (2021a), Chen et al. (2020), and Luo et al. (2022).

We first outline the attack strategy introduced by Carlini et al. (2021a), which applies directly to InstaHide without requiring additional assumptions and forms the basis for the subsequent attacks.

The attack described by Carlini et al. (2021a) targets the reconstruction of the noisy linear system of equations produced by the InstaHide algorithm. To accomplish this, the adversary must first identify which private images contribute to each encoded image. The core methodology involves generating encoded images from public data using InstaHide and training a neural network to predict whether pairs of encoded images share a common private image. Although one might expect that randomly flipping the sign of each pixel in the encoded images would hinder the training process, the authors demonstrate that this challenge can be circumvented by using the absolute values of the pixel intensities. Once the neural network is trained, the attacker can infer which private images are shared between encoded pairs. Since the weights used to generate each encoded image are revealed through the encoded labels, the adversary can then construct and solve a noisy linear system, where the noise results from the inclusion of public images. Notably, because different public images are used across encoded samples, this noise can be modeled as samples from a mean-zero Gaussian distribution, which averages out over many equations. This approach enables the attacker to recover the private images, up to the sign of each pixel. To further enhance the visual quality of the reconstructed images, a recoloring scheme is applied.

In contrast to the attack described by Carlini et al. (2021a), the approach proposed in Chen et al. (2020) assumes that the original data follows a Gaussian distribution. Under this assumption, the encoded dataset exhibits a folded Gaussian distribution when considering the absolute values of the pixel intensities. With access to a large number of mixup images, the attacker can estimate the covariance matrix of the encoded data, which corresponds to the Gram matrix formed by the weight vectors used in the mixup process. By analyzing this Gram matrix, the attacker can infer which private images are present in each mixup image. Similar to the method in Carlini et al. (2021a), the attacker can subsequently recover the private images, up to the sign of each pixel, by solving the resulting linear system of equations.

To counter the attack introduced by Carlini et al. (2021a), Luo et al. (2022) proposed employing image augmentation techniques—such as shifting, cropping, rotation, and translation—to disrupt geometric pixel alignment. This strategy aims to prevent attackers from accurately reconstructing the system of equations underlying the mixup process. Their method shares similarities with ours, as both approaches seek to avoid the repetition of identical images across multiple encoded samples. However, as demonstrated by Luo et al. (2022), such defenses can be circumvented by training a fusion-denoising network to reconstruct private images.

In their attack, similar to the approach in Carlini et al. (2021a), the attacker trains a comparison network to determine whether two encoded images share a common private image. Using this information, they cluster encoded images that contain the same private image, even if different augmented versions are used. To align features within each cluster, a fusion-denoising network is employed: a convolutional neural network (CNN) first downsamples the images to mitigate the effects of geometric transformations, followed by a transpose CNN for upsampling. The outputs from each encoded image are then fused, either by averaging or by selecting the maximum value, and the resulting image is passed through a denoising network to recover the private image.

The attacks proposed in Carlini et al. (2021a) and Chen et al. (2020) are not effective against our algorithm due to the addition of noise. With respect to the attack in Luo et al. (2022), our method does not rely on geometric transformations, making the fusion-denoising attack less applicable. To prevent situations in which two noisy samples of the same image share common features that could be exploited by image relaxation techniques, we ensure that the noise added to each private sample is several orders of magnitude greater than the average distance between two distinct private samples. This strategy significantly reduces the risk of feature overlap between noisy samples, thereby enhancing the security of our scheme.

Although our mechanism may prevent an attacker from forming clusters and reconstructing the system of equations described in (6), we adopt a conservative security assumption by considering a strong adversary who is capable of doing so. This approach is motivated by the observation that, in all three attacks, the adversary is able to determine whether two encoded images share a common private image. In Carlini et al. (2021a) and Chen et al. (2020), the sign mask applied by InstaHide is neutralized by taking the absolute value, while in Luo et al. (2022), the use of augmented images is still insufficient to prevent clustering.

In the context of the system in (6), we assume the adversary knows the matrix $\tilde{X}$ of encoded images and the matrix W of weights, and seeks to recover the matrix X of original samples. In the case of mean-zero isotropic Gaussian data, this intuition is formally captured by the following theorem:

Theorem 4.1. *Let $X = \{x_1, \ldots, x_n\}$ be points in $\mathbb{R}^d$ independently sampled from $\mathcal{N}(0, \sigma^2 I)$. Fix $x_i \in X$ and let $\varepsilon \in (0, 1)$. Then for*

$$r = \sigma \sqrt{2d + 4\sqrt{d \log\left(\frac{n-1}{\varepsilon}\right)} + 4\log\left(\frac{n-1}{\varepsilon}\right)}, \tag{7}$$

we have $\mathbb{P}\left(\forall j \neq i : \|x_j - x_i\| \leq r\right) \geq 1 - \varepsilon$.

Equivalently, $\mathbb{P}\left(\exists j \neq i : \|x_j - x_i\| > r\right) \leq \varepsilon$.

Proof. Since $x_i, x_j \sim \mathcal{N}(0, \sigma^2 I)$ independently, we have $x_j - x_i \sim \mathcal{N}(0, 2\sigma^2 I)$. Therefore,

$$\frac{\|x_j - x_i\|^2}{2\sigma^2} \sim \chi^2_d. \tag{8}$$

By the Chernoff bound for chi-squared random variables **?**, for any $t > 0$,

$$\mathbb{P}\left(\chi^2_d \geq d + 2\sqrt{dt} + 2t\right) \leq e^{-t}. \tag{9}$$

Setting $t = \log\left(\frac{n-1}{\varepsilon}\right)$ and substituting:

$$\mathbb{P}\left(\|x_j - x_i\|^2 \geq 2\sigma^2\left(d + 2\sqrt{d\log\left(\frac{n-1}{\varepsilon}\right)} + 2\log\left(\frac{n-1}{\varepsilon}\right)\right)\right) \leq e^{-\log\left(\frac{n-1}{\varepsilon}\right)} \tag{10}$$

$$= \frac{\varepsilon}{n-1}. \tag{11}$$

Taking square roots of the argument inside the probability:

$$\mathbb{P}\left(\|x_j - x_i\| \geq \sigma\sqrt{2d + 4\sqrt{d\log\left(\frac{n-1}{\varepsilon}\right)} + 4\log\left(\frac{n-1}{\varepsilon}\right)}\right) \leq \frac{\varepsilon}{n-1}. \tag{12}$$

Therefore, for each fixed $j \neq i$:

$$\mathbb{P}\left(\|x_j - x_i\| \leq r\right) \geq 1 - \frac{\varepsilon}{n-1}. \tag{13}$$

By the union bound:

$$\mathbb{P}\left(\exists j \neq i : \|x_j - x_i\| > r\right) = \mathbb{P}\left(\bigcup_{j \neq i}\{\|x_j - x_i\| > r\}\right) \tag{14}$$

$$\leq \sum_{j \neq i} \mathbb{P}\left(\|x_j - x_i\| > r\right) \tag{15}$$

$$\leq (n-1) \cdot \frac{\varepsilon}{n-1} = \varepsilon. \tag{16}$$

This establishes $\mathbb{P}\left(\forall j \neq i : \|x_j - x_i\| \leq r\right) \geq 1 - \varepsilon$. □

Remark 4.2. The theorem shows that the radius required for ε-coverage grows as $\mathcal{O}(\sigma\sqrt{\log n})$, which is significantly smaller than the typical inter-point distance $\mathcal{O}(\sigma\sqrt{d})$ in high dimensions. This reflects the concentration of measure phenomenon in Gaussian distributions.

What theorem tell us is that when the noise is large enough, i.e., $r \geq$ than the probability of finding a point *outside* the ball centered at the recovered point and with radius given by the norm of the error which was added during the miuxp computation in ALgorithm 1 is less than a chosen threshold ϵ.

5 EXPERIMENTS

Our experiments are designed to evaluate both the accuracy loss relative to InstaHide and the attack resilience of our proposed algorithm.

Setup: We conduct our experiments on three widely used benchmark datasets: MNIST LeCun (1998), CIFAR-10, and CIFAR-100 Krizhevsky et al. (2009). All implementations are carried out using the PyTorch framework Paszke et al. (2019). For the classification accuracy experiments, we utilize feature representations obtained from the output of the final convolutional layer of publicly available pretrained image models. In the adversarial attack experiments, the raw images serve as inputs to our proposed algorithm. Specifically, feature maps are extracted using a pretrained ResNet-18 model for MNIST and CIFAR-10, and a pretrained ResNet-34 model for CIFAR-100.

Accuracy: The input to Algorithm 1 consists of feature maps corresponding to images from the selected benchmark datasets. The subsequent output of the algorithm is then classified using a convolutional neural network. We evaluate our algorithm for $k = 4$ and $k = 6$, and for each value of k, we consider six distinct noise norms, each determined by a multiplication factor mf. Specifically, the noise norm r is calculated as the product of mf and the average distance between two randomly selected samples. The choice of the average inter-sample distance as a baseline is motivated by the findings of Luo et al. (2022), which demonstrate that attacks can be effective when private images are insufficiently perturbed, thereby enabling adversaries to exploit structural similarities between the transformed and original images. In our experiments, we set $mf \in 1, 2, 4, 8, 16, 32$. The results are presented in Table 1. As with InstaHide, the accuracy decreases as k increases, and for our algorithm, the accuracy also decreases as mf increases. This behavior with respect to mf is expected, since the norm of the noise is on the order of a constant factor mf times the average distance between pairs of different images. For large values of mf, the difference in accuracy between Singularized-Mixup and InstaHide increases, but for moderate values it remains within a maximum of 5%. This characterizes the tradeoff between security and privacy, which is common to all learnable obfuscation solutions, because mf directly influences the security of the scheme. A larger mf induces more noise, making it more difficult for the attacker to recover the original data.

Table 1: Comparison of InstaHide and Singularized Mixup (S-Mixup for short) over MNIST, CIFAR-10, and CIFAR-100 for $k \in \{4, 6\}$ and $mf \in \{1, 2, 4, 8, 16, 32\}$.

Setting		**MNIST**		**CIFAR-10**		**CIFAR-100**	
k	mf	InstaHide	S-Mixup	InstaHide	S-Mixup	InstaHide	S-Mixup
4	1	99.66	99.46	91.20	91.94	74.01	75.82
4	2	99.66	99.44	91.20	91.82	74.01	75.68
4	4	99.66	99.28	91.20	89.95	74.01	73.78
4	8	99.66	98.56	91.20	86.73	74.01	68.97
4	16	99.66	97.13	91.20	83.57	74.01	59.93
4	32	99.66	95.40	91.20	81.08	74.01	48.38
6	1	99.44	99.40	85.25	91.85	69.09	74.44
6	2	99.44	99.37	85.25	91.25	69.09	74.22
6	4	99.44	99.12	85.25	89.23	69.09	72.11
6	8	99.44	98.37	85.25	85.87	69.09	67.41
6	16	99.44	96.62	85.25	82.38	69.09	56.62
6	32	99.44	94.52	85.25	78.94	69.09	41.01

Security:

Table 2: Comparison between InstaHide and Singularized Mixup (S-Mixup) for noise levels selected according to the security analysis.

	mf	InstaHide	S-Mixup
MNIST	32	99.66	95.40
CIFAR-10	8	91.20	86.73
CIFAR-100	8	74.01	68.97

To recover the original images from their mixed representations, we employ a gradient descent-based optimization procedure inspired by the strategy in Luo et al. (2022). In this setting, we assume that the attacker has prior knowledge of the mixing process, specifically the mixing matrix, and leverages this information to guide the recovery. The attacker also incorporates prior assumptions about the structure and statistics of natural images by introducing regularization terms into the loss function. The composite loss consists of a reconstruction term, which measures the mean squared error between the observed mixed images and the reconstructed mixtures (obtained by linearly combining the recovered images using the mixing matrix), a total variation regularization term to encourage spatial smoothness and suppress noise, and an L_2 norm penalty to prevent implausibly large pixel values. These regularization terms encode prior knowledge about natural images, such as their tendency to be spatially smooth and to have bounded intensity values. The optimization is performed using the Adam optimizer, with the recovered images constrained to a fixed range after each update to ensure plausible pixel values.

Figure 1 shows the average distance between an original image and its corresponding recovered image for different noise levels and $k = 4$. To better illustrate the tradeoff between security and utility, Figure 2 presents the average recovered image from each dataset for each noise level. We observe that for the simpler MNIST dataset, a large noise multiplier is required; in this case, $mf = 32$ provides the best security. For more complex data, such as CIFAR-10 and CIFAR-100, a noise factor of $mf = 8$ is sufficient to prevent the recovery of the original image. This behavior is due to the fact that as more complex data is included in each mixup, it becomes more difficult for the attacker to reverse the operation, even at lower noise levels. Table 2 presents a comparison of the classification accuracy between InstaHide and our algorithm for the noise levels identified in the security analysis. As shown, the reduction in accuracy remains within a 5% margin.

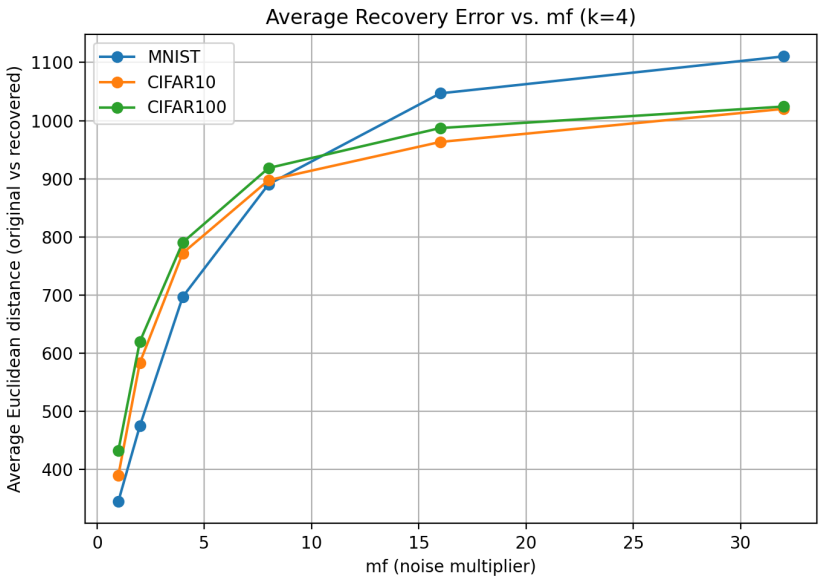

Figure 1: Average distance between original images and their recovered counterparts.

Figure 2: Each pair of rows shows the averaged recovered image for each dataset. In each pair, the first row displays the original images, while the second row shows the corresponding recovered images. The first two rows correspond to MNIST images, the next two rows to CIFAR-10 images, and the last two rows to CIFAR-100 images.

6 Conclusions

In this paper, we introduced a novel mixup algorithm within the singularization framework. Our approach adds noise of a specified magnitude to all but the target sample in each convex combination. This design is motivated by three major attacks on InstaHide, which exploited the repeated use of the same private image across multiple mixup images. Such repetition enables attackers to cluster mixup images that share a common private image and subsequently recover the private image by averaging out the other components as noise.

Similar to InstaHide, a necessary condition for any successful attack on our algorithm is the ability to cluster mixup images that share a common private image, even in the presence of noise. Without this clustering step, inverting individual mixup images is information-theoretically infeasible. While our Singularized-Mixup procedure may also hinder clustering, we conservatively assume a more powerful attacker who has access to the entire weight matrix. Our primary focus is to demonstrate that, even under this strong threat model, sufficiently large noise prevents reliable recovery of the original samples.

Experimentally, we compared our method to InstaHide and evaluated the trade-off between security and utility across different noise magnitudes. At the optimal balance point, we identified noise levels that effectively prevent attackers from recovering original samples, while incurring only a 5% accuracy loss compared to InstaHide.

Reproducibility Statement

All source code used in this study is publicly available, along with detailed instructions to reproduce the experiments described in this paper. The data utilized comes from publicly accessible benchmark datasets. For additional details regarding the experimental setup and procedures, please refer to the Appendix F.

REFERENCES

Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Abhradeep Thakurta, and Florian Tramèr. Is private learning possible with instance encoding? In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 410–427. IEEE, 2021a.

Nicholas Carlini, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Florian Tramer. Neuracrypt is not private. *arXiv preprint arXiv:2108.07256*, 2021b.

Sitan Chen, Xiaoxiao Li, Zhao Song, and Danyang Zhuo. On instahide, phase retrieval, and sparse matrix factorization. *arXiv preprint arXiv:2011.11181*, 2020.

Shaltiel Eloul, Fran Silavong, Sanket Kamthe, Antonios Georgiadis, and Sean J Moran. Mixing gradients in neural networks as a strategy to enhance privacy in federated learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3956–3965, 2024.

Yingwei Fu, Huaimin Wang, Kele Xu, Haibo Mi, and Yijie Wang. Mixup based privacy preserving mixed collaboration learning. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 275–2755. IEEE, 2019.

Chrystel Gaber, Gilles Macariot-Rat, Simona David, Jean-Philippe Wary, and Alain Cuaboz. Position paper: Strengthening applets on legacy sim cards with singularization, a new moving target defense strategy. In *International Conference on Mobile, Secure, and Programmable Networking*, pp. 71–74. Springer, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Qijian He, Wei Yang, Bingren Chen, Yangyang Geng, and Liusheng Huang. Transnet: Training privacy-preserving neural network over transformed layer. *Proceedings of the VLDB Endowment*, 13(12):1849–1862, 2020.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International conference on machine learning*, pp. 4507–4518. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Q Li, Y Zhang, J Ren, Q Li, and Y Zhang. You can use but cannot recognize: Preserving visual privacy in deep neural networks. arxiv 2024. *arXiv preprint arXiv:2404.04098*.

Zhijian Liu, Zhanghao Wu, Ligeng Zhu, Chuang Gan, and Song Han. Facemix: Privacy-preserving face attribute classification on the cloud. 2019.

Zhijian Liu, Zhanghao Wu, Chuang Gan, Ligeng Zhu, and Song Han. Datamix: Efficient privacy-preserving edge-cloud inference. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 578–595. Springer, 2020.

Xinjian Luo, Xiaokui Xiao, Yuncheng Wu, Juncheng Liu, and Beng Chin Ooi. A fusion-denoising attack on instahide with data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1899–1907, 2022.

Gilles Macario-Rat and Mihail-Iulian Plesa. Singularization: A new approach to designing block ciphers for resource-constrained devices. In *International Conference on Attacks and Defenses for Internet-of-Things*, pp. 155–167. Springer, 2024.

S Nythia, S Priyaa, R Priyanka, and S Saranya. Face image recognition and scrambling for privacy using neural networks. *International Journal of Scientific & Engineering Research (IJE), India*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 8024–8035. Curran Associates, Inc., 2019.

Andreea Bianca Popescu, Ioana Antonia Taca, Anamaria Vizitiu, Cosmin Ioan Nita, Constantin Suciu, Lucian Mihai Itu, and Alexandru Scafa-Udriste. Obfuscation algorithm for privacy-preserving deep learning-based medical image analysis. *Applied Sciences*, 12(8):3997, 2022.

Sifat Ut Taki and Spyridon Mastorakis. Amalgam: A framework for obfuscated neural network training on the cloud. In *Proceedings of the 25th International Middleware Conference*, pp. 238–251, 2024.

Yinggui Wang, Yuanqing Huang, Jianshu Li, Le Yang, Kai Song, and Lei Wang. Adaptive hybrid masking strategy for privacy-preserving face recognition against model inversion attack. *arXiv preprint arXiv:2403.10558*, 2024.

Yuexin Xiang, Tiantian Li, Wei Ren, Tianqing Zhu, and Kim-Kwang Raymond Choo. A lightweight privacy-preserving scheme using pixel block mixing for facial image classification in deep learning. *Engineering Applications of Artificial Intelligence*, 126:107180, 2023.

Hanshen Xiao, G Edward Suh, and Srinivas Devadas. Formal privacy proof of data encoding: The possibility and impossibility of learnable encryption. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1834–1848, 2024.

A Yala et al. Neuracrypt: hiding private health data via random neural networks for public training (2021).

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A Appendix

You may include other additional sections here.

B Chen attack

The attack described in Chen et al. (2020) simplifies the InstaHide problem by assuming that the matrix $X \in \mathbb{R}^{d\times n}$ of images is Gaussian, i.e., its entries are chosen i.i.d. from $\mathcal{N}(0,1)$. Let $p_1,\ldots,p_d \in \mathbb{R}^n$ be the rows of X. Consider $w_{i1}, dots, w_m \in \mathbb{R}^n$ the unknown selection vectors chosen from a distribution $\mathcal{D}$. $S \subset \{1,\ldots,m\}$ be the coorsinates of the public imges and $S^c = \{1,\ldots,n\}$ S be the coordinates of the private images. Let $[v]_S \in \mathbb{R}^{|S|}$ be the restriction of a vector v to the coordinates indexed by S. Each selection sector generates an encoded image as:

$$\tilde{x}_i = |Xw_i| \tag{17}$$

In InstaHide, the sign of each pixel from an encoded image is randomly flipped, but as the authors remarks, the two notations are interchangeable.

The attack goes like follows:

1. **Learning the public coordinates of any selection vector** In the first step, the attacker determines the weights associated with the public images from each selection vector. Considering the matrix
$$N = \mathbb{E}_{p,\tilde{x}}\left[\tilde{x}^2 \cdot ([p]_S[p]_S^\intercal - \mathrm{Id})\right]$$

where $\tilde{x} = |\langle w, p \rangle|$, $p \sim \mathcal{N}(0, \mathrm{Id})$. It can be proven that N is a rank-1 matrix proportional to $[w]_S[w]_S^\intercal$. Moreover, N can be approximated by

$$\hat{N} = \frac{1}{d} \sum_{i=1}^{d} \tilde{x}_i^2 \cdot ([p_i]_S[p_i]_S^\intercal - \mathrm{Id}) .$$

2. **Recovering the Gram Matrix** Since the previous steps recovers the coordinates of the public images in each selection vector, for the simplicty of the description, consider that all images are private, i.e., $S^c = \{1, \ldots, n\}$. Consider the matrix $\tilde{X} \in \mathbb{R}^{m \times d}$ where each line is an encoded images:

$$\tilde{X} = \begin{pmatrix} |\langle p_1, w_1 \rangle| & \cdots & |\langle p_d, w_1 \rangle| \\ \vdots & \ddots & \vdots \\ |\langle p_1, w_m \rangle| & \cdots & |\langle p_d, w_m \rangle| \end{pmatrix} \tag{18}$$

We can use the columns of $\tilde{X}$ to estimate the covariance matrix $\tilde{M}$ of the folded Gaussian distribution $\mathcal{N}^{\text{fold}}(0, M)$, since each column is drawn independently from this distribution. The covariance matrix M is, in fact, the rescaled $m \times m$ Gram matrix whose entries are proportional to the dot product of any two selection vectors; that is, the element at position (i, j) in the matrix M is given by $k \cdot \langle w_i, w_{ij} \rangle$.

3. **Floral submatrices**

The previous step of the attack shows the dot product between any two selection vectors, i.e., $\langle w_i, w_{ij} \rangle$, thus the attacker know how many private images are common between two encoded images. In order to identify which private images are common (not only how many), the attacker identifies in M floral submatrices. The rows/columns of a floral submatrix can be indexed by all subsets of size k of a set of $k + 2$ elements where its entries are the interscetion sizes between the subsets. More intuitively, the attacker exploits the fact that the subsets of size k of of the set $\{1, \ldots, k + 2\}$ are uniquily identified by their pairwise intersection sizes.

4. **Determining the private images**

uppose the attacker has identified a floral matrix in the previous steps, which corresponds to the selection vectors $w_{i1}, \ldots, w_{i_t}$, where $t = \binom{k+2}{k}$. The structure of the floral matrix encodes information about the indices of private images that are common between pairs of selection vectors. Specifically, the row and column indices of the matrix indicate which private images are shared. This allows the attacker to construct a system of equations of the form $|\langle w_{ij}, p_l \rangle| = \tilde{x}_l$ for all $l \in 1, \ldots, d$, where p_l denotes the private images and $\tilde{x}_l$ are known quantities.

From another perspective, each row or column of the floral submatrix can be indexed by a subset of size k from a set of size $k+2$. Each element in such a subset represents the index of a private image. For any given element in the floral matrix—which itself is a submatrix of the Gram matrix M—the position of the element along the rows provides the attacker with a set of k private image indices, while the position along the columns provides another set of k indices. By intersecting these two sets, the attacker can determine which private images are common between the selection vectors associated with the corresponding row and column. Solving the resulting system of equations enables the attacker to recover the indices of the private images.

C LUO ATTACK

In Luo et al. (2022), the authors observed that the method proposed by Carlini in Carlini et al. (2021a) can be mitigated by applying data augmentation before the mixup process. To address this, they introduce a new approach that successfully bypasses this mitigation strategy. Their method operates as follows:

1. In the first step, the attacker computes the absolute value of each pixel in every encoded image.

2. Next, a similarity score is calculated for every pair of encoded images to determine, with high probability, whether a given pair is derived from the same private image. To compute this score, the authors propose a comparative network that takes as input both high-resolution and low-resolution versions of the image pairs. This approach yields better results than the standard ResNet architecture used by Carlini. Based on the similarity scores, the attacker clusters the encoded images, with each cluster corresponding to a distinct private image.

3. For each cluster obtained in the previous step, the attacker re-weights all encoded images using the weights associated with the corresponding private image. These weights can be easily inferred from the associated encoded labels. Subsequently, a neural network is trained to perform image relaxation and fusion. This strategy counteracts the effects of geometric image augmentation by generating a set of features that are invariant to geometric transformations. An initial version of the private image is then constructed in the fusion step by combining these feature maps.

4. In the final step, the attacker trains an additional neural network to denoise the image produced in the previous stage.

D CARLINI ATTACK

The attack consists of two main stages. In the first stage, the attacker determines the two private images used to generate each encoded image during the mixing process. In the second stage, the attacker reconstructs the private images by solving a noisy linear system of equations:

1. The attacker computes the absolute value of each mixup encoding to counteract the random sign changes introduced by the mask σ_i in (3).

2. To identify whether two encoded images share at least one common private image, the attacker calculates a similarity function between each pair of encoded images. This similarity function is approximated using a neural network trained on public data transformed via the mixup algorithm. Using the similarity scores, the attacker constructs a weighted graph where vertices represent encoded samples, and edge weights correspond to the similarity function's output.

3. Based on the weighted graph, the attacker identifies densely connected cliques, enabling clustering of encoded samples that share a common private image. Each cluster is represented as a set S_i, $1 \leq i \leq n$, where each set contains encoded samples derived from the same private image.

4. Since each encoded image is generated by mixing two private images, the attacker constructs a bipartite similarity graph connecting encoded images to the sets identified in the previous step. Edge weights represent the distance between an encoded image x_i and a set S_i. This step determines, for each encoded image, the two sets corresponding to the private images used in its construction.

5. Using the bipartite graph, the attacker maps each encoded image to two sets, representing the private images involved in its generation during the mixup process.

6. The attacker recovers the weights used to generate each encoded image by analyzing the mixup of the labels, as described in (4). Since the labels are one-hot encoded, recovering the associated weights is straightforward.

7. Finally, the attacker constructs a matrix $B \in \mathbb{R}^{n \times d}$, where each row corresponds to an encoded image $\tilde{x_i}$, i.e., $B_i = \tilde{x_i}$. A sparse matrix $M \in \mathbb{R}^{n \times n}$ is also constructed, where each row contains two non-zero entries representing the weights $w^i 1$ and w_2^i associated with the private images used to compute the corresponding encoded image. Let $A \in \mathbb{R}^{n \times d}$ represent the matrix of private images, where each row $A_i = x_i$. The attacker solves the noisy linear system $B = M \cdot A + e$, where e represents the public images used in the mixup. This system can be efficiently solved using gradient descent.

E MORE RELATED WORK

Liu et al. (2020) proposed a different approach, where a classifier is trained on mixup samples and images to produce mixup results that can later be de-mixed. Unlike previous methods, this approach does not involve training on mixup samples followed by inference on original data. Instead, both training and inference are performed on mixup data, with the inference process generating mixup results that can then be used to recover the correct labels.

In a more recent study, Wang et al. (2024) proposed a mixup-like approach to mitigate model inversion attacks on face recognition systems. Instead of mixing images directly, the authors suggested mixing samples in the frequency domain. Additionally, they employed a reinforcement learning strategy to dynamically determine the number of images to mix, balancing privacy and utility. Similarly, Xiang et al. (2023) introduced a mixing strategy to preserve image privacy during training. Their method involves splitting each image into multiple blocks and replacing parts of these blocks with corresponding blocks from other images with the same label. In another study, Li et al. proposed a new privacy metric called Visual Feature Entropy (VFE), calculated for a region of an image as the sum of squared gradients with respect to both axes. This metric aims to quantify the amount of information that needs protection by analyzing the entropy of a region. The authors' mixing strategy involves shuffling pixels within an image based on the VFE metric. Although this method does not involve computing a weighted sum, it can be interpreted as a form of intra-image data mixing. Eloul et al. (2024) present the concept of mixing gradients in federated learning to enhance security against gradient inversion attacks. Although their method does not involve using random weights for gradient mixing, their straightforward approach of directly averaging gradients across a batch, combined with modifications to the loss function, significantly improves resistance to gradient inversion attacks.

The concept of data mixing is rooted in the broader idea of learnable obfuscation, which encompasses techniques designed to transform data in a way that allows algorithms to learn from the transformed data while safeguarding the privacy of the original data He et al. (2020); Yala et al.; Taki & Mastorakis (2024); Popescu et al. (2022); Nythia et al. (2017). For instance, in Nythia et al. (2017), the authors propose using the Arnold transformation to scramble images before inputting them into a face recognition system. This transformation rearranges image pixels by mapping each pixel to a new location determined by a linear transformation.

In Popescu et al. (2022), a method combining Variational Autoencoders (VAEs) with a substitution technique is introduced to protect medical images during neural network analysis. The approach involves training a VAE to reconstruct the image and then applying a substitution table to the latent space representation of the data. Similarly, Taki & Mastorakis (2024) presents a method to ensure the privacy of both training data and neural network architecture. For image data, the authors propose transforming it into a higher-dimensional space. To protect the architecture, they introduce random subnetworks with synthetic parameters that do not affect the network's accuracy or data flow.

The NeuraCrypt method, proposed in Yala et al., protects data privacy by transforming it with a random neural network. This approach is extended to enable privacy-preserving collaborative training, where all parties share transformed data with a central server. For the server to learn patterns from the combined datasets, all parties must use the same neural network for data transformation. Finally, He et al. (2020) introduces a privacy-preserving method that applies a linear transformation to each data sample. The authors also provide formal proofs demonstrating the information-theoretic security of their approach under specific conditions.

A common characteristic of learnable obfuscation techniques is that the same transformation—though potentially generated using independently chosen random parameters—must be applied to all samples in the dataset being protected. This creates a notable vulnerability: such techniques cannot provide security against chosen-plaintext attacks. This limitation, formally introduced and proven in Xiao et al. (2024), highlights an inherent weakness in these methods. Informally, learnable obfuscation can protect the privacy of plaintext data only under the assumption that the attacker does not have prior knowledge of the original data.

At first glance, this assumption may seem reasonable, as protecting data already known to an attacker might appear unnecessary. However, in practical scenarios, this assumption often fails. For instance, to improve the generalization capabilities of a machine learning model, private datasets

are frequently augmented with publicly available data. For example, a private image dataset might be enriched with images from CIFAR-100 Krizhevsky et al. (2009). To preserve the discriminative properties of the data and enable the model to generalize, the added public data must undergo the same transformation used to protect the private dataset.

This practice introduces a significant risk: an attacker with access to both the original public dataset and its transformed version could potentially design algorithms to reverse-engineer the transformation applied to the private data. An example of this vulnerability is described in Carlini et al. (2021b), where the authors successfully developed an algorithm to solve the NeuraCrypt challenge Yala et al., effectively bypassing the intended privacy protections.

F Experimental Details

All experiments were developed using the PyTorch framework and performed on an NVIDIA L4 GPU with 24 GB of available VRAM. Across all datasets, we consistently used a batch size of 128. For optimization, we employed the `AdamW` optimizer with a weight decay of 1×10^{-4}. The initial learning rate was set to 0.001 for all benchmarks, and we utilized a cosine annealing learning rate scheduler.

Our experimental evaluation was conducted on three distinct benchmarks using ResNet architectures (He et al., 2016), with specific configurations detailed in Table 3. A key aspect of our methodology is the exclusive use of the feature extraction layers from these architectures; the final classifier layers were omitted as our focus is on training features. For all datasets we apply similar transformations, which consist of a resize operation (224 pixels height and width) and a normalization. The MNIST dataset is also adjusted such that it has three channels, making it compatible with the chosen architectures.

Table 3: The configurations used within experiments for each dataset. The mean and standard deviation values for MNIST are for a single channel, while for CIFAR datasets they correspond to the (R, G, B) channels.

Dataset	Architecture	Epochs	Mean	Std.
MNIST	ResNet18	120	0.1307	0.3081
CIFAR-10	ResNet18	200	[0.4914, 0.4822, 0.4465]	[0.2470, 0.2435, 0.2616]
CIFAR-100	ResNet34	200	[0.5071, 0.4867, 0.4408]	[0.2675, 0.2565, 0.2761]

The training objective was to minimize the following loss function, which is designed for our Mixup implementation:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\text{softmax}(\mathbf{o}_i)_j) \tag{19}$$

where N is the batch size, C is the number of classes, $\mathbf{y}_i$ is the (potentially soft) label for sample i, and $\mathbf{o}_i$ is the model output for sample i. In PyTorch, this is implemented as:

```
loss = -(labels * torch.log_softmax(outputs, dim=1)).sum(dim=1).mean()
```

For the final classification step, we used a custom feed-forward neural network with three dense layers. The first and second of these dense layers are followed by batch normalization, GELU activation, and then a dropout. The precise structure of this neural network is described by the following equation:

$$\begin{aligned}\texttt{Classifier}(x; n_{\text{cls}}) = \text{Flatten}(x) &\rightarrow \text{Linear}_{in;1024} \rightarrow \text{BN}_{1024} \rightarrow \text{GELU} \rightarrow \text{Dropout}(0.5) \\ &\rightarrow \text{Linear}_{1024;512} \rightarrow \text{BN}_{512} \rightarrow \text{GELU} \rightarrow \text{Dropout}(0.5) \\ &\rightarrow \text{Linear}_{512;n_{\text{cls}}}\end{aligned} \tag{20}$$

The n_{cls} term represents the number of classes that the classification must be made on (e.g., 10 for MNIST). The in dimension of the flattened tensor x within the first linear layer is 25088 ($512 \times 7 \times 7$ for ResNet18 and ResNet34).

G THE USE OF LARGE LANGUAGE MODELS

We utilized Large Language Models (LLMs) in three specific ways during this work. First, after conducting a manual review of the state of the art using traditional search engines such as Google Scholar, we used LLMs to assist in identifying additional relevant papers. Second, LLMs were employed to help implement the experiments described in this study. Third, LLMs were used for grammar correction and minor improvements to the flow of the text. Importantly, LLMs were not used to generate or write any paragraphs; their role in writing was limited to minor edits and enhancements.