Differential Privacy: Gradient Leakage Attacks in Federated Learning Environments

Miguel Fernandez-de-Retana [1 2] **Unai Zulaika** [2] **Rubén Sánchez-Corcuera** [2] **Aitor Almeida** [2]

Abstract

Federated Learning (FL) allows for the training of Machine Learning models in a collaborative manner without the need to share sensitive data. However, it remains vulnerable to Gradient Leakage Attacks (GLAs), which can reveal private information from the shared model updates. In this work, we investigate the effectiveness of Differential Privacy (DP) mechanisms — specifically, DP-SGD and a variant based on explicit regularization (PDP-SGD) — as defenses against GLAs. To this end, we evaluate the performance of several computer vision models trained under varying privacy levels on a simple classification task, and then analyze the quality of private data reconstructions obtained from the intercepted gradients in a simulated FL environment. Our results demonstrate that DP-SGD significantly mitigates the risk of gradient leakage attacks, albeit with a moderate trade-off in model utility. In contrast, PDP-SGD maintains strong classification performance but proves ineffective as a practical defense against reconstruction attacks. These findings highlight the importance of empirically evaluating privacy mechanisms beyond their theoretical guarantees, particularly in distributed learning scenarios where information leakage may represent an unassumable critical threat to data security and privacy.

Keywords: Data Leakage · Differential Privacy · Federated Learning · Input Reconstruction · ML Privacy

1. Introduction

Federated Learning (FL) has recently emerged as a crucial paradigm in *distributed* training, enabling multiple entities (or *nodes*) to collaboratively train Machine Learning (ML) models without the need to share their local data. This approach is especially valuable in scenarios where data privacy is of utmost importance, such as in health-related applications. By keeping data on local nodes and only sharing updates through the exchange of gradients, FL seeks to mitigate the risks associated with exposing *sensitive data*. This strategy not only reduces the need to transfer large volumes of data but also minimizes the risk of privacy breaches, since data never leaves the local host and model updates are securely aggregated on a central server. This allows models to learn from a wide variety of data without compromising individual privacy, as the updates do not directly reveal information about the local data of each node. In this sense, FL presents itself as a promising solution to address the challenges of privacy and security in ML, e.g., enabling organizations to collaborate in developing robust shared models without compromising the confidentiality of their sensitive data.

However, despite its inherent privacy advantages, FL is not immune to vulnerabilities. One of its most prominent threats are **Gradient Leakage Attacks** (GLAs), in which an adversary — potentially a malicious participant or an entity intercepting communications as a *man-in-the-middle* — can infer sensitive information about a client's private training data by analyzing the gradients, or parameter updates, shared with the central server (Zhu et al., 2019; Zhao et al., 2020). It has been shown that, under certain conditions, original data instances can be reconstructed with high fidelity from these gradients (Wei et al., 2020), posing a considerable privacy breach and, thus, a significant risk to data security.

To counter these threats and strengthen the privacy guarantees of FL, **Differential Privacy** (DP) has become the gold standard (Dwork et al., 2006; 2014). DP provides a rigorous framework for quantifying and limiting the information that an algorithm's output reveals about any individual record. In the context of ML, and more specifically FL, this is typically achieved through mechanisms such as Differentially-Private Stochastic Gradient Descent (DP-SGD) (Abadi et al., 2016), which introduce calibrated noise into the training process (specifically, by clipping the norm of individual gradients

---

[1]*Modeling & Simulation in Life and Materials Sciences*, Basque Center for Applied Mathematics (BCAM) [2]*Faculty of Engineering*, University of Deusto (Bilbao, Spain). Correspondence to: Miguel Fernandez-de-Retana <m.fernandezderetana@deusto.es>.

and adding Gaussian noise to the aggregated sum) to obscure the contribution of each instance. Nevertheless, while effective in guaranteeing privacy, the application of DP often entails a *trade-off* in model performance (e.g., in reduced accuracy) and may require careful tuning of its hyperparameters (such as the privacy budget ε and the failure probability δ) (Abadi et al., 2016). Moreover, most recently, alternatives and refinements to DP-SGD have emerged, such as regularization-based approaches (Aguilera-Martínez & Berzal, 2024), which aim to offer a different balance between privacy, utility, and computational efficiency.

This work explores the intersection of federated learning, gradient leakage attacks, and differential privacy. Our main objective is to investigate and evaluate the effectiveness of DP mechanisms, particularly DP-SGD and related approaches, as defenses against GLAs in a simulated FL environment. To this end, we first introduce the fundamental concepts of FL, GLAs, and DP. We then compare the performance of several computer vision models trained with, and without, differential privacy protection on a moderately complex classification task. Then, we simulate a representative example of a gradient leakage attack and assess, both quantitatively and qualitatively, the impact of DP on the quality of the reconstructed data. Finally, we discuss the results, highlighting the inherent trade-off between privacy and utility, and outline potential limitations in this critical area of information security within distributed ML.

In summary, the main contributions of our work[1] are the following:

- To present the *theoretical foundations* behind differential privacy and gradient leakage attacks.

- To compare the performance of models trained with and without *privacy protection* on a moderately complex classification task.

- To simulate a representative example of a *gradient leakage attack* and investigate the impact of differential privacy on the quality of the reconstructed data.

2. Related Work

The study of security and privacy in distributed Machine Learning builds upon several key concepts. Below, we provide a brief snapshot of the *state-of-the-art* in the areas most relevant to this work, with technical details to be further elaborated later in Section 3.

Federated Learning (FL): Originally introduced by McMahan et al. (2017) and further developed in subsequent works such as those by Kairouz et al. (2021) and Li et al. (2020),

Federated Learning is a decentralized ML paradigm that allows multiple clients to collaboratively train a *shared model* without the need to exchange their local data. This distributed approach is particularly well-suited for privacy-sensitive applications, such as those involving medical or biometric information, since the raw data remains on the user devices or within the infrastructure of participating organizations. FL aims to leverage the computational resources and data heterogeneity distributed across clients, while maintaining a foundational level of local data privacy and security.

Gradient Leakage Attacks (GLA): However, it soon became evident that the *de facto* privacy offered by standard FL was limited against certain adversaries. Recent studies have demonstrated that an attacker with access to gradients, or parameter updates shared during federated training, can reconstruct, with surprising fidelity in some cases, the private data samples used to compute those gradients (Geiping et al., 2020; Huang et al., 2021; Zhu et al., 2019; Zhao et al., 2020). These attacks, generically referred to as Gradient Leakage Attacks (GLA), expose a critical vulnerability that can compromise the confidentiality of the obfuscated data.

Differential Privacy (DP): Introduced by Dwork et al. (2006) and formalized in subsequent works such as (Dwork et al., 2014), Differential Privacy has become the reference framework for providing robust and quantifiable privacy guarantees in data analysis and Machine Learning. DP offers protection against a wide range of inference attacks, including GLAs, by mathematically ensuring that the output of an algorithm (e.g., model parameters or aggregated gradients) does not reveal excessive information about any individual entry in the original dataset. In Machine Learning, DP is commonly implemented through mechanisms such as Differentially-Private Stochastic Gradient Descent **(DP-SGD)** (Abadi et al., 2016), or via explicit **regularization methods** (Aguilera-Martínez & Berzal, 2024), which introduce calibrated noise during training to obscure individual contributions, although this often entails a *trade-off* in the final model quality. Moreover, more recent approaches explore alternative pathways, such as generating *synthetic datasets* that intrinsically comply with differential privacy, e.g., by leveraging Large Language Models (LLMs) to create private synthetic textual data (Kurakin et al., 2023).

This work lies at the intersection of these three areas, specifically evaluating the effectiveness of Differential Privacy as a countermeasure against Gradient Leakage Attacks in Federated Learning environments. However, it is worth noting that, although the context of this work is framed within federated learning, we will not delve deeply into its specific aspects. Instead, we assume that model gradients have been intercepted in an FL setting and focus *exclusively* on the issues related to DP and the aforementioned attacks.

---

[1]Code available at https://github.com/miguelfrndz/Differential-Privacy-GL-Attacks

3. Background

In this section, we detail the essential theoretical foundations related to the concepts of Federated Learning (Section 3.1), Gradient Leakage Attacks (Section 3.2), and Differential Privacy (Section 3.3), providing the necessary groundwork to understand the analysis and results presented later.

3.1. Federated Learning (FL)

Federated Learning (FL) is a distributed Machine Learning paradigm designed to enable multiple *clients* (e.g., mobile devices, institutions, local servers) to collaboratively train a global model without the need to share their local training data (McMahan et al., 2017; Kairouz et al., 2021; Li et al., 2020). This allows for addressing critical concerns related to privacy, security, and communication costs typically associated with centralizing data in traditional ML workflows. Additionally, in the context of increasing regulatory constraints on data protection and governance (e.g., GDPR, HIPAA), FL offers a viable solution for complying with legal and ethical requirements imposed by current legislation.

Typically, a standard FL system consists of the following two main components:

- A set of **client nodes** (or *workers*), denoted by $k \in \{1, \ldots, N\}$, each with its own local dataset D_k, which is not shared with other clients or with the server.

- A **central server** (or *coordinator*) that orchestrates the training process and maintains the state of the global model.

The training process itself is iterative and generally unfolds over multiple *communication rounds*. In each round t:

1. **Distribution:** The central server selects a subset of available clients (e.g., $S_t \subseteq \{1, \ldots, N\}$) and sends them the current parameters of the global model θ_t.

2. **Local Training:** Each client $k \in S_t$ performs a *local training* using the received model θ_t and its own dataset D_k. This local training usually consists of running one or more steps (epochs) of a standard optimization algorithm, such as Stochastic Gradient Descent (SGD), to minimize a local loss function $\mathcal{L}(\theta_t^{(k)}; D_k)$.

   The result is an updated set of local parameters $\theta_{t+1}^{(k)}$, or alternatively, a "*computed*" update such as the difference $\Delta\theta_t^{(k)}$ or the average gradient $\nabla\mathcal{L}(\theta_t^{(k)}; D_k)$.

$$\theta_{t+1}^{(k)} = \theta_t - \eta^{(k)}\nabla\mathcal{L}(\theta_t^{(k)}; D_k) \qquad (1)$$

3. **Communication:** Clients $k \in S_t$ send their local updates (i.e., $\theta_{t+1}^{(k)}$ or $\Delta\theta_t^{(k)}$) back to the central server.

It is important to note that the raw data D_k *never* leaves the client during this process.

4. **Aggregation:** The central server collects the updates received from the clients participating in round t and *aggregates* them to produce an improved global model θ_{t+1}. Several aggregation strategies exist, the most common being *Federated Averaging (FedAvg)* (McMahan et al., 2017). In FedAvg, assuming clients return their local parameters $\theta_{t+1}^{(k)}$, the server computes the new global model as a weighted average as in (2).

$$\theta_{t+1} = \sum_{k \in S_t} \frac{n_k}{N_t} \theta_{t+1}^{(k)} \qquad (2)$$

where $n_k = |D_k|$ is the number of data samples on client k, and $N_t = \sum_{k \in S_t} n_k$ is the total number of samples across the participating clients in round t. This weighting ensures that clients with more data have a greater influence on the global model. The new global model θ_{t+1} then becomes the reference for the next training round.

Other aggregation alternatives include:

- Variants of FedAvg with different weighting schemes or adaptive learning rates.

- Algorithms that aggregate gradients or parameter differences instead of absolute parameters.

- Use of *Secure Aggregation* protocols (Bonawitz et al., 2017), which employ cryptographic techniques to allow the server to compute the sum (or weighted average) of updates without accessing the individual updates $\theta_{t+1}^{(k)}$ or $\Delta\theta_t^{(k)}$ from each client. This provides an extra layer of privacy, protecting the updates even from the central server itself.

This four-step cycle is repeated for a predefined number of rounds or until the global model θ_t converges according to some criterion (e.g., accuracy on a global validation set or the stabilization of the loss function).

While the federated learning paradigm improves privacy by avoiding the need to centralize raw data, the updates shared by clients may still **reveal sensitive information** about their local data. Therefore, it is essential to implement additional privacy and security mechanisms to protect data confidentiality throughout the federated training process.

3.2. Gradient Leakage Attacks (GLA)

Although Federated Learning proposes a framework that avoids sharing raw local training data, the model updates (i.e., typically the gradients) that clients send to the central server are not free from privacy concerns. It has been shown
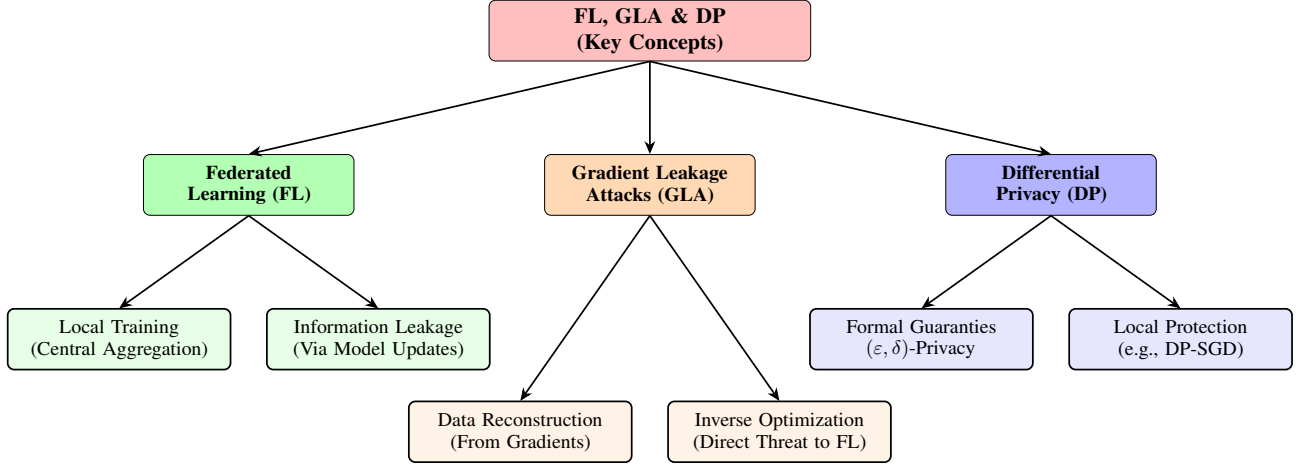
Figure 1. Conceptual Summary of the Three Pillars of This Work: Federated Learning (FL), Gradient Leakage Attacks (GLA), and Differential Privacy (DP)

that these updates, while seemingly obfuscated, can contain enough information for an adversary to infer, or even reconstruct with relative fidelity, the private data samples used to compute them (Zhu et al., 2019; Zhao et al., 2020; Geiping et al., 2020; Gong et al., 2023). These methods are collectively known as **Gradient Leakage Attacks (GLA)**, and we provide below a brief description of how they work.

A typical GLA scenario assumes an adversary with specific capabilities. Generally, a *white-box* attack is considered, where the attacker knows the model architecture, the loss function $\mathcal{L}$, and the model parameters θ_t prior to the client's local update. The goal of the attacker is to *intercept* the local update from client k before it is securely aggregated with the updates from other clients. This adversary could be an external entity intercepting communications (i.e., a *man-in-the-middle* attack), or in some cases, the central server itself if it has been compromised and no secure aggregation has been implemented (Bonawitz et al., 2017), or even compromised client nodes acting maliciously (Gong et al., 2023).

The core idea behind GLAs is to formulate data reconstruction as an *inverse optimization problem*. The attacker initializes a "*fake*" or "*seed*" input $(x_{\text{rec}}^{(0)}, y_{\text{rec}}^{(0)})$ with random values (e.g., Gaussian noise for $x_{\text{rec}}^{(0)}$ and a random or inferred label for $y_{\text{rec}}^{(0)}$). The attacker then computes the gradient $\nabla\theta_{\text{att}}^{(0)}$ that this fake input would produce in the model f with the known parameters θ_t. The goal is to iteratively adjust $(x_{\text{rec}}, y_{\text{rec}})$ to minimize the discrepancy between the calculated gradient $\nabla\theta_{\text{att}}$ and the *real* intercepted gradient $\nabla\theta_t^{(k)}$ as in (3).

$$(x_{\text{rec}}^*, y_{\text{rec}}^*) = \underset{x', y'}{\arg\min} \|\nabla_{\theta_t}\mathcal{L}(f(x'; \theta_t), y') - \nabla\theta_t^{(k)}\|_2^2 \quad (3)$$

where $\|\cdot\|_2^2$ denotes the squared Euclidean norm.

If the attacker intercepts the updated parameters $\theta_{t+1}^{(k)}$ instead of the gradient directly, they can estimate the original gradient, assuming they know, or can accurately estimate, the local learning rate $\eta^{(k)}$ used by the client:

$$\nabla\theta_t^{(k)} \approx \frac{\theta_{t+1}^{(k)} - \theta_t}{\eta^{(k)}} \quad (4)$$

The optimization in (3) is typically carried out using gradient-based methods, where the update is applied directly to the variables x_{rec} and y_{rec}. However, optimizing over y_{rec} (which often belongs to a discrete space) can be challenging. A major breakthrough, proposed by Zhao et al. (2020), showed that the correct label y_{rec} can often be *inferred directly* by analyzing the signs or magnitudes of the components of the intercepted gradient $\nabla\theta_t^{(k)}$ corresponding to the weights in the final layer of the model. This allows y_{rec} to be fixed, simplifying the optimization problem to focus solely on x_{rec}.

Algorithm 1 summarizes the general process of a gradient-based reconstruction attack. After obtaining the target gradient $\nabla\theta_t^{(k)}$ (either directly or by deriving it from $\theta_{t+1}^{(k)}$), a seed $x_{\text{rec}}^{(0)}$ is initialized, and a label y_{rec} is either inferred or initialized. Then, in an iterative loop governed by a stopping condition T:

1. The attacker computes the gradient $\nabla\theta_{\text{att}}^{(\tau)}$ using the current seed $(x_{\text{rec}}^{(\tau)}, y_{\text{rec}})$.

2. A reconstruction loss $D^{(\tau)}$ is computed to measure the distance between $\nabla\theta_{\text{att}}^{(\tau)}$ and $\nabla\theta_t^{(k)}$, potentially including *regularization terms*, weighted by α, such as the

distance between the model output $f(x_{\text{rec}}^{(\tau)}; \theta_t)$ and the label y_{rec} (e.g., classification loss or regression equivalent), or the total variation loss of $x_{\text{rec}}^{(\tau)}$ (Geiping et al., 2020).

3. The seed $x_{\text{rec}}^{(\tau)}$ is updated by taking a gradient descent step on the loss $D^{(\tau)}$ with respect to $x_{\text{rec}}^{(\tau)}$, using a learning rate η' for the attack. As noted earlier, the same can be done for $y_{\text{rec}}^{(\tau)}$ if it has not been directly inferred from the gradients.

This process continues for a fixed number of iterations T, or until the reconstruction loss $D^{(\tau)}$ converges below a predefined threshold (both are valid termination criteria). The choice of optimizer for updating $(x_{\text{rec}}, y_{\text{rec}})$ is important; although the algorithm shows simple gradient descent, more sophisticated optimizers such as L-BFGS (Liu & Nocedal, 1989) or Adam (Kingma & Ba, 2014) are commonly used in practice due to their efficiency and ability to handle high-dimensional problems (Wei et al., 2020). Hyperparameters such as the number of iterations T, the learning rate η', and the regularization weight α must be carefully tuned to achieve high-quality reconstructions.

The demonstrated existence and effectiveness of GLAs underlines a fundamental vulnerability in standard Federated Learning, and strongly motivates the need for more robust privacy-preserving mechanisms, such as Differential Privacy, which we discuss in the next section.

3.3. Differential Privacy (DP)

Differential Privacy (DP) (Dwork et al., 2006; 2014) has become the *de facto* standard for providing reliable and quantifiable privacy guarantees in the training of Machine Learning models, where its goal is to limit the amount of information an adversary can infer about the dataset used to train a model. We present below some basic definitions and theorems related to differential privacy. The two theorems presented, following the notation commonly used across the literature, are fundamental for understanding the concept of DP and its application in the context of information security. In this case, both theorems are stated according to the definition of *differential privacy* introduced by Dwork et al. (2006) and in the seminal work of Abadi et al. (2016). In essence, differential privacy is based on the idea that the output of an algorithm should not depend *excessively* on any single individual instance, ensuring that private information cannot be inferred from the algorithm's output.

Formally, differential privacy is defined in terms of the *indistinguishability* of the outputs of a model $\mathcal{M}$ — or more generally, an algorithm $\mathcal{A}$ — when applied to *neighboring* datasets, i.e., datasets that differ by a single record (see Definition 3.1). Based on this notion of neighborhood, the two

---

Algorithm 1 Gradient-Based Reconstruction Attack (Wei et al., 2020; Gong et al., 2023)

1: **Input:** differentiable model $f(x; \theta_t)$, local gradient from node k $\nabla\theta_t^{(k)}$, client learning rate $\eta^{(k)}$, stopping condition T, attack learning rate η', regularization weight α
2: **if** $\theta_t^{(k)}$ updated $\theta_{t+1}^{(k)}$ is available **then**
3:
$$\nabla\theta_t^{(k)} \leftarrow \frac{\left(\theta_{t+1}^{(k)} - \theta_t\right)}{\eta^{(k)}}$$

4: **end if**
5: *Initialize attack seed:* $x_{\text{rec}}^{(0)}$
6: *Infer label:* $y_{\text{rec}} \leftarrow \text{argmin}_i \left(\nabla_i \theta_t^{(k)}\right)$
7: **for** $\tau = 1$ **to** T **do**
8:     *Compute attack gradient:*
$$\nabla\theta_{\text{att}}^{(\tau)} \leftarrow \frac{\partial \mathcal{L}(f(x_{\text{rec}}^{(\tau)}; \theta_t), y_{\text{rec}})}{\partial\theta_t}$$

9:     *Compute reconstruction loss:*
$$D^{(\tau)} \leftarrow \|\nabla\theta_{\text{att}}^{(\tau)} - \nabla\theta_t^{(k)}\|^2 + \alpha\|f(x_{\text{rec}}^{(\tau)}; \theta_t) - y_{\text{rec}}\|^2$$

10:     *Update seed:* $x_{\text{rec}}^{(\tau+1)} \leftarrow x_{\text{rec}}^{(\tau)} - \eta'\frac{\partial D^{(\tau)}}{\partial x_{\text{rec}}^{(\tau)}}$

11: **end for**
12: **Output:** Reconstructed data $(x_{\text{rec}}, y_{\text{rec}})$

---

main variants of differential privacy are defined in Theorems 3.2 and 3.3.

Definition 3.1 (Neighborhood in $\mathcal{D}$). Let $D, D' \in \mathcal{D}$ be two *datasets* defined over the same domain $\mathcal{D}$. We say that D and D' are *neighbors* or *adjacent* if they differ in exactly one instance, that is, if D can be obtained from D' (or vice versa) by adding or removing a single element.

Theorem 3.2 (ε-Differential Privacy). *An algorithm $\mathcal{A}\colon \mathcal{D} \to \mathcal{R}$ is ε-differentially private if, for every pair of neighboring datasets $D, D' \in \mathcal{D}$ and for any subset $E \subseteq \mathcal{R}$, it holds that:*

$$P\left(\mathcal{A}(D) \in E\right) \leq e^{\varepsilon} \cdot P\left(\mathcal{A}(D') \in E\right) \tag{5}$$

That is, the probability that algorithm $\mathcal{A}$ produces an output in the set E does not change significantly regardless of whether dataset D or D' is used.

Theorem 3.3 ((ε, δ)-Differential Privacy). *An algorithm $\mathcal{A}\colon \mathcal{D} \to \mathcal{R}$ is (ε, δ)-differentially private if, for every pair of neighboring datasets $D, D' \in \mathcal{D}$ and for any subset $E \subseteq \mathcal{R}$, it holds that:*

$$P\left(\mathcal{A}(D) \in E\right) \leq e^{\varepsilon} \cdot P\left(\mathcal{A}(D') \in E\right) + \delta \tag{6}$$

Algorithm 2 Differentially-Private Stochastic Gradient Descent (DP-SGD)

1: **Input:** loss function $\mathcal{L}(\theta)$, mini-batch size L, learning rate η, clipping parameter C, noise σ, number of iterations T
2: **Initialize:** θ_0 randomly
3: **for** $t = 1$ **to** T **do**
4:     Sample mini-batch $\mathcal{B}_t$ of L instances
5:     **for** each $x_i \in \mathcal{B}_t$ **do**
6:         *Compute gradient:* $g_i = \nabla_\theta \mathcal{L}(\theta_{t-1}, x_i)$
7:         *Clip:* $\bar{g}_i \leftarrow g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right)$
8:     **end for**
9:     *Compute aggregated gradient:*

$$\tilde{g}_t \leftarrow \frac{1}{L}\left(\sum_{i=1}^{L} \bar{g}_i\right) + \mathcal{N}(0, \sigma^2 C^2 I)$$

10:     *Update parameters:* $\theta_t \leftarrow \theta_{t-1} - \eta \tilde{g}_t$
11: **end for**
12: **Output:** final parameters θ_T

Here, ε is the **privacy budget**, which measures the upper bound on per-record privacy leakage. A lower ε implies stronger privacy. The parameter δ, typically chosen to be a small value (e.g., less than the inverse of the dataset size), represents the probability that the ε-DP guarantee may fail. In other words, with probability $(1 - \delta)$, the algorithm behaves as ε-DP.

Moreover, this definition can be naturally extended to the case where the two datasets $D, D' \in \mathcal{D}$ differ in more than one instance. In this case, we talk about *group privacy*, and although it is not commonly used in practice, we include its definition in Lemma 3.4 (Gupta & Oakley, 2021; Jagielski et al., 2020).

Lemma 3.4 (Group Differential Privacy). *An algorithm* $\mathcal{A}: \mathcal{D} \to \mathcal{R}$ *is group* (ε, δ)-*differentially private if, for every pair of neighboring datasets* $D, D' \in \mathcal{D}$ *and for any subset* $E \subseteq \mathcal{R}$*, it holds that:*

$$P\left(\mathcal{A}(D) \in E\right) \leq e^{k\varepsilon} \cdot P\left(\mathcal{A}(D') \in E\right) + \frac{e^{k\varepsilon} - 1}{e^\varepsilon - 1}\delta \quad (7)$$

where k is the number of instances by which the datasets differ.

In summary, the main idea behind DP is that the inclusion, or exclusion, of certain *private* training instances should not significantly affect the outcome of our model. In this setting, ε represents the *privacy budget*, which quantifies the level of privacy achieved, and δ corresponds to a small probability of error in maintaining or guaranteeing this privacy.

The most widely extended mechanism for achieving differential privacy is **Differentially-Private Stochastic Gradi-**

Algorithm 3 Basic Differential Privacy Accounting

1: **Input:** number of steps T, per-step privacy parameters $(\varepsilon_t, \delta_t)$, model state $\mathcal{M}_t$
2: **Initialize:** $(\varepsilon_t, \delta_t) \leftarrow (\varepsilon_0, \delta_0)$
3: **for** $t = 1$ **to** T **do**
4:     $\varepsilon_t \leftarrow \texttt{ExtractEpsilon}(\mathcal{M}_t)$
5:     $\delta_t \leftarrow \texttt{ExtractDelta}(\mathcal{M}_t)$
6: **end for**
7: **Output:** accumulated privacy $\{(\varepsilon_t, \delta_t)\}_{t=0}^{T}$

---

ent Descent (DP-SGD), proposed by Abadi et al. (2016) and presented in Algorithm 2. DP-SGD modifies the standard SGD procedure at two key steps in each training iteration over a mini-batch $\mathcal{B}_t$ of size L:

1. **Gradient Clipping:** Before aggregating the mini-batch gradients, the individual gradients g_i for each sample $x_i \in \mathcal{B}_t$ are computed as in (8).

$$g_i = \nabla_\theta \mathcal{L}(\theta_{t-1}, x_i), \quad x_i \in \mathcal{B}_t \quad (8)$$

Then, the L_2 norm of each gradient is clipped to a maximum threshold C according to (9).

$$\bar{g}_i \leftarrow g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right) \quad (9)$$

This step is crucial because it limits the *sensitivity* of the aggregated gradient to any individual sample. Bounding the maximum influence of each data point is a fundamental requirement for calibrating the added noise properly.

2. **Noise Addition:** After computing the average of the clipped gradients $\bar{g}_i$, random noise is added to the result, typically Gaussian. The noise is sampled from $\mathcal{N}(0, \sigma^2 C^2 I)$, where I is the identity matrix, C is the clipping threshold, and σ is the *noise multiplier*. The noise magnitude ($\propto \sigma C$) is calibrated to obscure the individual contribution of each sample in the aggregated gradient $\tilde{g}_t$, thereby providing the DP guarantee. A larger σ implies more noise, thus stronger privacy (i.e., smaller ε), but potentially lower practical utility.

Finally, the model parameters are updated using this noisy aggregated gradient as in (10).

$$\theta_t \leftarrow \theta_{t-1} - \eta \tilde{g}_t \quad (10)$$

A fundamental aspect of iterative algorithms with DP, such as DP-SGD, is the privacy budget tracking (*privacy accounting*, see Algorithm 3). Each step of DP-SGD consumes part of the total budget (ε, δ) which must be tracked across the

T training iterations. Simple composition (e.g., aggregating ε and δ across steps) provides valid but often overly conservative bounds. In practice, more refined techniques are used, such as Advanced Composition (Dwork et al., 2014), or more commonly, accounting based on Rényi Differential Privacy (RDP) (Mironov, 2017) or Privacy Loss Distribution (PLD) analysis (Sommer et al., 2018), among others (Zhu et al., 2022). These methods, often referred to as *privacy accountants*, incorporate mini-batch sampling probabilities and the noise mechanism (e.g., Gaussian) to provide much tighter bounds on the accumulated budget (ε, δ) after T steps. Tools such as Opacus (Yousefpour et al., 2021) for PyTorch implement these advanced accountants as a *de facto* standard.

The application of DP, and in particular DP-SGD, inevitably introduces a **privacy-utility trade-off**. Adding more noise (larger σ, smaller ε) to strengthen privacy guarantees, typically degrades model performance in classification or regression tasks. Finding the right balance requires careful tuning of both DP hyperparameters (e.g., C, σ) and training hyperparameters (e.g., η, L, T) for a target budget (ε, δ).

Additionally, Lomurno & Matteucci (2022) showed that some classical regularization methods (e.g., *dropout* or L_2) behave similarly to DP-SGD in terms of privacy levels, with a substantial reduction in computational cost and impact on prediction quality. However, these techniques *do not* provide the formal, quantifiable guarantees of differential privacy.

More recently, following this same line of thought, Aguilera-Martínez & Berzal (2024) demonstrated that the noise injection and clipping of DP-SGD can be interpreted as an implicit regularization of the loss function $\mathcal{L}$. However, it is important to note that this regularization is *independent* of the model parameters θ. To address this, the authors introduced a new method for differential privacy via **explicit regularization** (PDP-SGD, *Proportional DP-SGD*), where the injected noise is *proportional* to each parameter. In principle, this yields better privacy guarantees more efficiently, since any traditional optimization scheme can be used to train our neural network.

In the context of federated learning, differential privacy is typically applied *locally*. Each client runs DP-SGD, or the chosen method, on its local data before sending updates to the server. This protects client data not only from external adversaries but also from the central server itself should it be compromised. Likewise, it also acts as a direct defense, at least *in theory*, against the gradient leakage attacks discussed earlier in Section 3.2, since the gradient intercepted by an attacker would have already been previously perturbed and thus should not reveal *sensitive information* about the underlying training data.
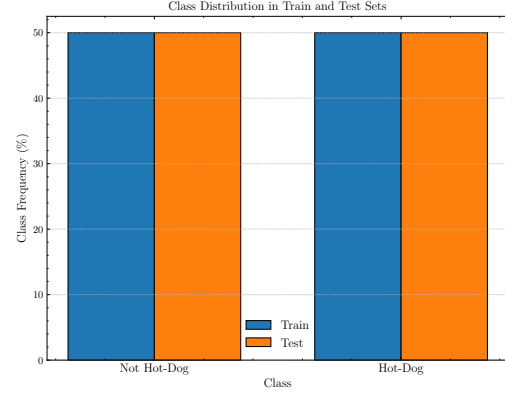

Figure 2. Class Distribution in the Training and Test Splits

4. Methodology

In this section, we detail the experimental methodology followed to evaluate the impact of Differential Privacy (DP) on the performance of both image classification models and on their ability to mitigate gradient leakage attacks (GLA). Our objective is twofold:

- First, to compare the performance of different Computer Vision (CV) architectures trained with, and *without*, differential privacy guarantees under various privacy budgets;

- Second, to analyze qualitatively and quantitatively how these mechanisms affect an adversary's ability to reconstruct private training data from shared gradients, simulating a GLA attack scenario.

4.1. Classification Using Differentially Private Models

To evaluate the *trade-off* between model privacy and utility, we select a binary classification task with enough complexity to observe significant performance differences under different training conditions. An overly simple task, such as the MNIST dataset used for the GLA simulation, may fail to reveal performance degradations introduced by DP; while an overly complex task may hinder the training of useful models even without privacy.

Dataset: We use an outcome-balanced subset of the well-known Food-101 dataset (Bossard et al., 2014) focused on binary classification with an equal distribution of classes. This dataset, with representative examples shown in Figure 3, consists of images belonging to two classes: *hot-dog* and *not hot-dog*. As seen in Figure 2, the dataset is perfectly balanced between the two classes in both the training and evaluation sets, thus avoiding class imbalance biases.

Compared Models: In order to assess the impact of different differential privacy methods across model architectures

Figure 3. Examples of Training Images (*Hot-Dog vs Not Hot-Dog*)

and training strategies, we compare three distinct computer vision models:

- **Simple Convolutional Neural Network (Custom-CNN):** A relatively small and standard CNN architecture (two convolutional layers with *max-pooling*, followed by two linear layers), trained entirely from scratch. This serves as a baseline for evaluating performance without the benefit of pretraining.

- **ResNet50:** We employ a ResNet50 (He et al., 2016) pretrained on ImageNet (Deng et al., 2009). We apply a *fine-tuning* strategy where all pretrained convolutional layers have been frozen, and only a new final classification head is trained for our task. The ResNet50 architecture is based on *residual blocks*, which enable training deeper networks without suffering from vanishing gradient problems. In this case, we use the *pretrained backbone* as a feature extractor and train only the final classification head. Figure 4 illustrates the general architecture of ResNet50.

- **Vision Transformer (ViT) DINOv2 with Registers:** Finally, we employ an advanced Vision Transformer (ViT) model, specifically DINOv2 (Oquab et al., 2023), incorporating the *registers* technique (Darcet et al., 2023). Registers are additional tokens introduced during pretraining that help the model capture global scene information, improving downstream performance. More specifically, they *remove artifacts* in ViT feature maps, as shown in Figure 5, and improve classification performance. As with ResNet50, we freeze the pretrained ViT backbone and train only the final classification head for our task.

Training Methods: Each of the three models described above has been trained under different regimes to compare their performance with and without differential privacy protection:

1. **Standard Training (No DP):** Models have been trained using the Adam optimizer (Kingma & Ba, 2014) and the standard binary classification loss, *Binary Cross-Entropy* (BCE) in (11). This serves as a baseline scenario without any formal privacy guarantees ($\varepsilon = \infty$).

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \quad (11)$$

where N is the total number of instances, y_i is the true label, and $\hat{y}_i = f(x_i; \theta)$ is the model prediction for instance x_i.

2. **DP-SGD:** We train the models using the DP-SGD algorithm, as described in Section 3.3 (Algorithm 2). To assess the impact of different protection levels, we use three privacy budgets ($\varepsilon = 8$, $\varepsilon = 25$, and $\varepsilon = 50$), keeping δ fixed and small ($\delta \approx 1/N_{train}$, where N_{train} is the size of the training set). For the implementation, we use the Opacus library by Meta (Yousefpour et al., 2021).

3. **PDP-SGD (Explicit Regularization):** We also train our models using the Proportional DP-SGD (PDP-SGD) approach (Aguilera-Martínez & Berzal, 2024), which implements DP via explicit loss function regularization, where added noise is **proportional** to model parameters. This method is included in order to compare an alternative to DP-SGD which, theoretically, should offer better results in terms of model performance. The regularized loss function proposed by PDP-SGD is given by (12).

$$\mathcal{L}_{\text{PDP}} = \mathcal{L} + \kappa \sum_i \theta_i^2 x_i^2 \quad (12)$$

where $\mathcal{L}$ is the original loss function (e.g., binary cross-entropy in our case), and κ is a regularization constant related to the learning rate η and the noise parameter σ by $\kappa = \eta^2 \sigma^2$. Moreover:

$$\nabla_{\theta_i} \mathcal{L}_{\text{PDP}} = \nabla_{\theta_i} \mathcal{L} + 2\kappa x_i^2 \theta_i \quad (13)$$

Regarding training hyperparameters, all models have been trained with the Adam optimizer (Kingma & Ba, 2014), an image size of 224×224 pixels (in RGB), an initial learning rate of $\eta = 0.001$, and batch size $L = 32$. Additionally, we use *early-stopping* as a training halting criterion where training is terminated if no improvement on the validation loss is observed over 20 consecutive epochs. For DP-SGD, clipping size has been set to $C = 1.2$ with a default noise multiplier. Likewise, for the explicit PDP-SGD regularization, we use a noise multiplier of $\sigma = 0.1$ and a learning rate of $\eta = 0.1$. In all cases, the validation was carried out on a set comprising 20% of the total training data.
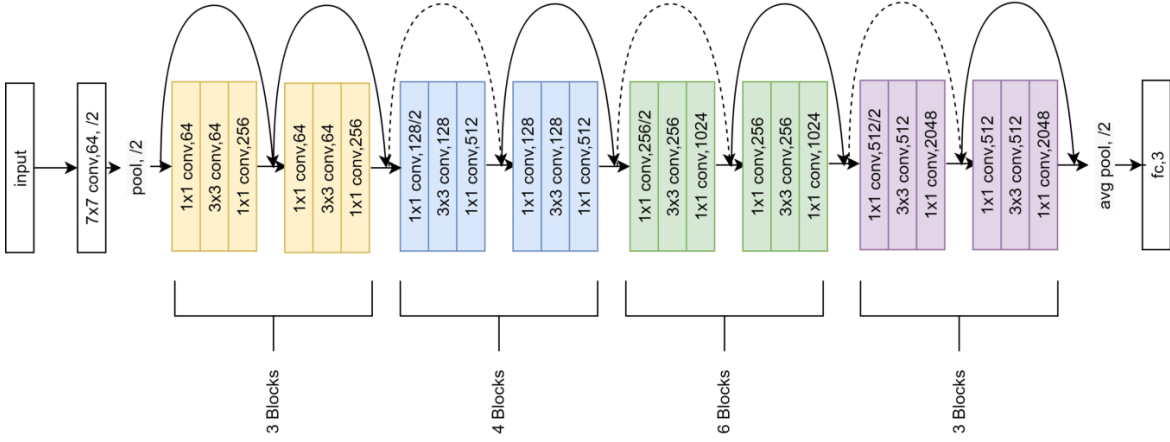
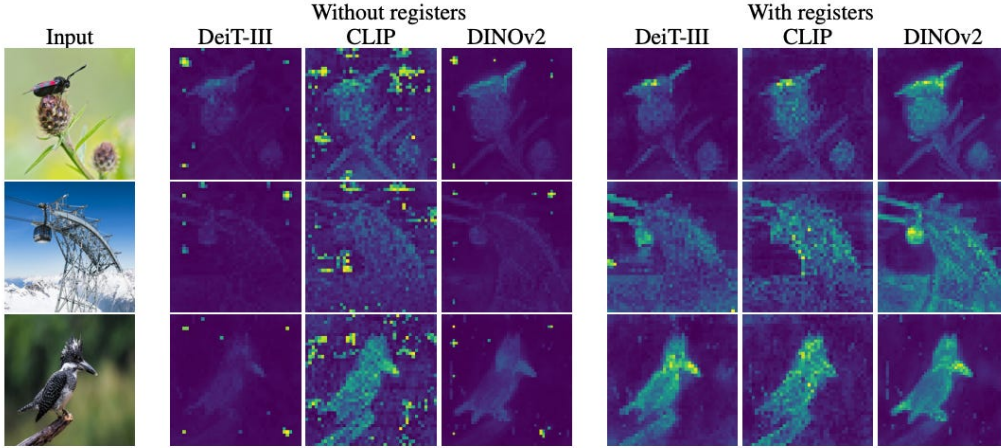*Figure 4.* ResNet50 Model Architecture (He et al., 2016)

Figure 5. Example of Artifact Removal in ViT Models with Registers (Darcet et al., 2023)

4.2. Reconstruction Attacks via Gradient Leakage

The second experimental component of this work focuses on the simulation and analysis of reconstruction attacks based on gradient leakages. Our goal is to investigate the feasibility and limitations of recovering private training data — specifically, images used to train a computer vision model, for example — from gradients intercepted in a federated learning environment under the influence of differential privacy methods.

Dataset & Model: For this stage, we use the **MNIST** dataset (LeCun et al., 1998), a standard benchmark for image classification tasks consisting of grayscale handwritten digits. In this case, the images have been resized to 32×32 pixels, and the victim model architecture is a simple Convolutional Neural Network (CNN), composed of four convolutional layers followed by a linear (dense) layer for classification. This architecture, while simple, is sufficient

to demonstrate the feasibility of gradient leakage attacks in a controlled environment.

Reconstruction Process: The attack aims to reconstruct an image x_{rec} and its corresponding label y_{rec} such that the gradients generated by this pair $(\nabla\theta_{\text{att}})$ match the original intercepted gradients $(\nabla\theta_t^{(k)})$ as closely as possible, as described in Equation 3. In our case, we *simultaneously optimize* both the reconstructed image and the label by employing an L-BFGS second-order optimizer (Liu & Nocedal, 1989). As mentioned in Section 3.2, the label can also be inferred directly from the intercepted gradients, but in our experiments, we found that jointly optimizing both the image and label yields better reconstruction results.

Victim Model Training Scenarios: To evaluate the impact of differential privacy methods as a safeguard against GLAs, the reconstruction attack is performed on gradients obtained from the victim CNN, trained (more on this later) under

9

three different conditions, analogous to those used in the classification task (Section 4.1):

1. **Standard Training:** Gradients from a model trained in a standard manner, without any differential privacy protection, are used. This serves as a baseline to determine the maximum possible information leakage in this scenario.

2. **DP-SGD Training:** Gradients are generated from a model trained using DP-SGD. The same privacy budgets (with $\delta \approx 1/N_{train}$) and DP hyperparameters as in the classification experiments are explored.

3. **PDP-SGD Training (Explicit Regularization):** Gradients are obtained from a model trained with the Proportional DP-SGD approach. The same regularization hyperparameters (e.g., κ derived from $\eta = 0.1$ and $\sigma = 0.1$) used in the classification task are employed.

Considerations on Attack Stability: One of the main challenges of gradient leakage reconstruction attacks is their *instability*. The reconstruction optimization process is highly sensitive to initialization and may converge to suboptimal critical points (local minima or saddle points) which do not correspond to the original data, or even *diverge* completely, especially if the initialization is inadequate or the search space is too complex. This lack of reliability is a significant limitation and was a determining factor in choosing a relatively simple dataset and model architecture (MNIST, small CNN), with small, black-and-white images.

To partially mitigate these stability and convergence issues, we initialize the neural network with a weight configuration that leads to a *stable reconstruction*. In practice, this is equivalent to assuming that the network has already been trained for a few epochs and that its weights are in a region of the parameter space that is conducive to convergence. This assumption allows us to focus on the information leakage mechanism through gradients without the results being obscured by an early divergence of the attack's optimizer due to poor model initialization.

Finally, although the general context of this work is framed within federated learning, for this specific GLA simulation, we assume that the gradients have been *intercepted* in a federated learning environment, but in practice, the attack is performed on a single, centrally trained model. This allows us to study the information leakage process from an individual client's gradient without the added complexity of multi-client aggregation or federated communication protocols. The goal is to demonstrate the potential for information leakage at the level of a single gradient, which is the fundamental unit of information shared in many FL schemes.

5. Results & Discussion

In this section, we present and analyze the results derived from the experiments described in Sections 4.1 and 4.2. The content is organized as follows: first, we describe the *evaluation metrics* used to quantify both model performance and the quality of reconstructions (Section 5.1). Next, we report the results obtained in the *classification task* under several differential privacy regimes, comparing the utility of various privacy levels and protection mechanisms on the performance of vision models (Section 5.2). Finally, we present the results of the reconstruction attacks via *gradient leakage*, where we assess both qualitatively and quantitatively the quality of the reconstructed images under the influence of DP-SGD and PDP-SGD, and discuss their effectiveness as defense strategies against the recovery of sensitive information (Section 5.3).

All experiments were run using PyTorch and Python 3.12. Given the modest scale of the models and datasets, the experiments can also be reproduced on any standard consumer-grade GPUs.

5.1. Evaluation Metrics

To quantitatively evaluate the results obtained in the two main experimental tasks of this work — image classification and gradient-based reconstruction attacks — we employ different sets of metrics adapted to each objective.

For the **binary classification task** (*Hot-Dog vs Not Hot-Dog*), we use the standard metrics commonly applied to classification problems, as presented in Table 1. These metrics are derived from the entries of the confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Specifically, these metrics are:

- **Accuracy:** The overall proportion of correct predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (14)$$

- **Precision:** Among all instances predicted as positive, how many were truly positive?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (15)$$

- **Recall (Sensitivity):** Of all instances that were truly positive, how many were correctly identified?

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (16)$$

- **Specificity:** Of all instances that were truly negative, how many were correctly identified?

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (17)$$

- F_1**-Score:** The harmonic mean of Precision and Recall, providing a balance between the two. This is especially useful in cases of class imbalance (although that is not the case in this dataset).

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (18)$$

- **MCC (Matthews Correlation Coefficient):** A correlation coefficient between the observed and predicted classifications. Considered a robust and balanced metric even with imbalanced classes. It ranges from -1 (total disagreement) to $+1$ (perfect agreement), with 0 indicating random correlation.

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \quad (19)$$

For the task of **image reconstruction** from the gradient leakage (GLA), the goal is to evaluate the similarity between the original image x and the reconstructed image x_{rec}. While metrics such as the Mean Squared Error (MSE) are common, they do not always reflect *human-perceived similarity*. Therefore, in this work, we focus on the **Structural Similarity Index Measure (SSIM)** (Wang et al., 2004), specifically designed to capture structural similarity between two images in a way that is more *consistent* with human visual perception.

SSIM evaluates similarity based on three comparative components between the two images (x and x'), typically computed over local windows and then averaged:

1. **Luminance:** Compares the mean brightness of the images, based on pixel intensity means ($\mu_x, \mu_{x'}$):

$$l(x, x') = \frac{2\mu_x \mu_{x'} + c_1}{\mu_x^2 + \mu_{x'}^2 + c_1} \quad (20)$$

2. **Contrast:** Compares the variation in intensity (dynamic range) around the mean, based on the standard deviations ($\sigma_x, \sigma_{x'}$):

$$c(x, x') = \frac{2\sigma_x \sigma_{x'} + c_2}{\sigma_x^2 + \sigma_{x'}^2 + c_2} \quad (21)$$

3. **Structure:** Compares the structural correlation between the images, based on covariance ($\sigma_{xx'}$):

$$s(x, x') = \frac{\sigma_{xx'} + c_3}{\sigma_x \sigma_{x'} + c_3} \quad (22)$$

where $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, and $c_3 = c_2/2$ are stabilization constants to avoid division by zero, with L being the dynamic range of the pixel values (e.g., 255 for 8-bit images), and k_1, k_2 are two small constants (typically $k_1 = 0.01, k_2 = 0.03$).

The global SSIM index is then defined as a combination of these three components:

$$\text{SSIM}(x, x') = [l(x, x')]^\alpha \cdot [c(x, x')]^\beta \cdot [s(x, x')]^\gamma \quad (23)$$

where $\alpha > 0, \beta > 0, \gamma > 0$ are parameters controlling the relative importance of each component. Commonly, $\alpha = \beta = \gamma = 1$, simplifying the formula to:

$$\text{SSIM}(x, x') = \frac{(2\mu_x \mu_{x'} + c_1)(2\sigma_{xx'} + c_2)}{(\mu_x^2 + \mu_{x'}^2 + c_1)(\sigma_x^2 + \sigma_{x'}^2 + c_2)} \quad (24)$$

The SSIM value ranges between -1 and 1, where 1 indicates perfect structural similarity between the two images. Higher SSIM values indicate visually higher-quality reconstructions in the context of GLA, meaning that the attacker has recovered an image more closely resembling the original.

5.2. Classification with Differentially Private Models

The quantitative results of the **image classification** task (*Hot-Dog vs Not Hot-Dog*), using the different architectures and training methods with and without differential privacy, and evaluated with the metrics presented in Section 5.1, are summarized in Table 1. The analysis of these results reveals some general trends regarding the *trade-off* between the strength of privacy guarantees and model utility (or performance), as well as some noteworthy differences between DP-SGD and explicit regularization (PDP-SGD).

Overall, and as expected, the introduction of differential privacy through **DP-SGD tends to degrade model performance** compared to the traditionally trained alternatives (without DP, $\varepsilon = \infty$). This degradation manifests itself in most evaluation metrics (Accuracy, F_1-Score, MCC) and is generally more pronounced the stricter the privacy budget (i.e., the smaller the value of ε). This phenomenon is a direct consequence of the two core mechanisms behind DP-SGD: gradient clipping, which limits the magnitude of individual updates; and Gaussian noise injection, which obscures the contribution of each data point, potentially interfering with optimal model learning.

For the Simple Convolutional Neural Network (Custom-CNN) trained from scratch, the standard training established a modest baseline (e.g., Accuracy 0.584, MCC 0.168). Applying DP-SGD with $\varepsilon = 8$ results in a significant performance drop (Accuracy 0.562, MCC 0.131), particularly driven by an extremely low Recall (0.342), suggesting that the model becomes overly conservative in positive predictions under strong privacy constraints. However, we observe a *notable exception* with this architecture: for $\varepsilon = 25$ and $\varepsilon = 50$, performance (in terms of Accuracy and MCC) is slightly higher or comparable to that of standard training (e.g., Accuracy 0.598, MCC 0.198 for $\varepsilon = 25$). This anomaly may be attributable to the simplicity of the network

Table 1. Classification Results Across 10 Independent Executions (Values Reported as Mean $\pm$ Standard Error)

	ε	**Accuracy**	**Precision**	**Recall**	**Specificity**	F_1	**MCC**
Convolutional Neural Network (Custom-CNN)							
Standard Training	∞	$0.584_{\pm 0.005}$	$0.586_{\pm 0.006}$	$0.572_{\pm 0.010}$	$0.596_{\pm 0.011}$	$0.579_{\pm 0.006}$	$0.168_{\pm 0.009}$
DP-SGD ($\varepsilon = 8$)	8	$0.562_{\pm 0.013}$	$0.629_{\pm 0.030}$	$0.342_{\pm 0.151}$	$0.682_{\pm 0.152}$	$0.428_{\pm 0.088}$	$0.131_{\pm 0.015}$
DP-SGD ($\varepsilon = 25$)	25	$0.598_{\pm 0.015}$	$0.612_{\pm 0.024}$	$0.536_{\pm 0.118}$	$0.660_{\pm 0.153}$	$0.571_{\pm 0.038}$	$0.198_{\pm 0.018}$
DP-SGD ($\varepsilon = 50$)	50	$0.588_{\pm 0.010}$	$0.655_{\pm 0.019}$	$0.372_{\pm 0.019}$	$0.804_{\pm 0.033}$	$0.475_{\pm 0.009}$	$0.195_{\pm 0.022}$
Explicit Regularization	–	$0.614_{\pm 0.013}$	$0.609_{\pm 0.019}$	$0.636_{\pm 0.011}$	$0.592_{\pm 0.030}$	$0.622_{\pm 0.009}$	$0.228_{\pm 0.028}$
Pretrained ResNet50 (*Fine-Tuning* of Classification Head Only)							
Standard Training	∞	$0.886_{\pm 0.003}$	$0.873_{\pm 0.005}$	$0.904_{\pm 0.002}$	$0.868_{\pm 0.006}$	$0.888_{\pm 0.003}$	$0.773_{\pm 0.007}$
DP-SGD ($\varepsilon = 8$)	8	$0.844_{\pm 0.006}$	$0.831_{\pm 0.015}$	$0.864_{\pm 0.018}$	$0.824_{\pm 0.020}$	$0.847_{\pm 0.006}$	$0.689_{\pm 0.012}$
DP-SGD ($\varepsilon = 25$)	25	$0.868_{\pm 0.005}$	$0.871_{\pm 0.008}$	$0.864_{\pm 0.012}$	$0.872_{\pm 0.009}$	$0.868_{\pm 0.005}$	$0.736_{\pm 0.009}$
DP-SGD ($\varepsilon = 50$)	50	$0.866_{\pm 0.003}$	$0.877_{\pm 0.011}$	$0.852_{\pm 0.015}$	$0.880_{\pm 0.011}$	$0.864_{\pm 0.004}$	$0.732_{\pm 0.004}$
Explicit Regularization	–	$0.892_{\pm 0.002}$	$0.883_{\pm 0.004}$	$0.904_{\pm 0.003}$	$0.880_{\pm 0.005}$	$0.893_{\pm 0.002}$	$0.784_{\pm 0.004}$
Pretrained DINOv2 w/ Registers (Vision Transformer, *Fine-Tuning* of Classification Head Only)							
Standard Training	∞	$0.972_{\pm 0.004}$	$0.992_{\pm 0.002}$	$0.952_{\pm 0.007}$	$0.992_{\pm 0.001}$	$0.971_{\pm 0.004}$	$0.945_{\pm 0.007}$
DP-SGD ($\varepsilon = 8$)	8	$0.964_{\pm 0.002}$	$0.992_{\pm 0.002}$	$0.936_{\pm 0.003}$	$0.992_{\pm 0.001}$	$0.963_{\pm 0.002}$	$0.929_{\pm 0.003}$
DP-SGD ($\varepsilon = 25$)	25	$0.968_{\pm 0.003}$	$0.996_{\pm 0.001}$	$0.940_{\pm 0.007}$	$0.996_{\pm 0.001}$	$0.967_{\pm 0.004}$	$0.938_{\pm 0.007}$
DP-SGD ($\varepsilon = 50$)	50	$0.968_{\pm 0.001}$	$0.996_{\pm 0.001}$	$0.942_{\pm 0.004}$	$0.996_{\pm 0.001}$	$0.968_{\pm 0.001}$	$0.939_{\pm 0.003}$
Explicit Regularization	–	$0.976_{\pm 0.003}$	$0.990_{\pm 0.002}$	$0.962_{\pm 0.005}$	$0.990_{\pm 0.002}$	$0.976_{\pm 0.003}$	$0.953_{\pm 0.005}$

and the nature of the dataset; it is possible that the noise and clipping inherent to DP-SGD act as an unintended form of regularization. On the other hand, **explicit regularization (PDP-SGD) consistently improves all metrics** for the Custom-CNN (Accuracy 0.614, F_1-Score 0.622, MCC 0.228), outperforming both the baseline and all DP-SGD variants. This suggests that its mechanism, which in practice resembles an adaptive L_2 regularization, is beneficial in helping the model generalize better.

In the case of ResNet50 (pretrained with fine-tuning of the classification head), the standard model already achieves considerably high performance (Accuracy 0.886, MCC 0.773), owing to the pretrained features learned from ImageNet. Here, the introduction of DP-SGD also follows the general trend of degrading performance. For example, with $\varepsilon = 8$, Accuracy falls to 0.844 and MCC to 0.689. As ε increases (*weaker privacy*), performance progressively recovers, approaching but not fully reaching the baseline (e.g., Accuracy 0.866 with $\varepsilon = 50$). This behavior is expected,

as the added noise has a more noticeable effect when the model is already operating at high accuracy. Similarly to the Custom-CNN, explicit regularization (PDP-SGD) achieves a slight improvement over the baseline (Accuracy 0.892, MCC 0.784).

Finally, for the DINOv2 with Registers (pretrained Vision Transformer), the *most advanced* architecture, the standard baseline delivers exceptional performance (Accuracy 0.972, MCC 0.945). The application of DP-SGD, as in the case of ResNet50, consistently reduces performance, although the absolute magnitude of the drop is smaller in percentage terms given the very high starting point (e.g., Accuracy 0.964 with $\varepsilon = 8$). Improvement is observed when moving from $\varepsilon = 8$ to $\varepsilon = 25$, but the results for $\varepsilon = 25$ and $\varepsilon = 50$ are nearly identical (Accuracy 0.968 in both cases, MCC 0.938 and 0.939 respectively), suggesting that, for this model and with the DP hyperparameters used, a *privacy-utility plateau* is reached around these privacy budgets. Interestingly, explicit regularization (PDP-SGD)

achieves a *slight improvement* over standard training (Accuracy 0.976 vs 0.972, MCC 0.953 vs 0.945), demonstrating that parameter-proportional regularization can still provide benefits even for such advanced pretrained models. In fact, the additional L_2-type regularization introduced by PDP-SGD appears to consistently aid generalization across all architectures, likely by preventing overfitting and promoting smoother decision boundaries. However, its relative benefit diminishes as the base model's performance increases (e.g., DINOv2), indicating that highly capable architectures may already possess sufficient implicit regularization through their design and pretraining.

In summary, the classification results presented in Table 1 confirm that DP-SGD generally leads to a depreciation in model utility. The exception observed in the Custom-CNN for certain ε values highlights that the interaction between DP-SGD, the architecture of the model, and complexity of the task at hand can be non-trivial. Conversely, explicit regularization (PDP-SGD) emerges as a promising alternative, improving or at least maintaining performance in most cases, likely because its effect is analogous to well-calibrated L_2 regularization that aids *generalization*. However, its benefit may be null or even slightly negative in models that are already highly effective and robust due to their architecture and pretraining. We now turn to how these classification results translate into the effectiveness of reconstruction attacks via gradient leakage (GLA) in the next section.

5.3. Reconstruction Attacks via Gradient Leakage

In this section, we delve into the results of the reconstruction attack via gradient leakage (GLA), evaluating the ability of an adversary to recover images from the MNIST dataset using gradients obtained from a victim CNN. We analyze how the different training regimes — traditional (no privacy), DP-SGD, and PDP-SGD — impact the quality of such reconstructions. As mentioned in Section 4.2, it is crucial to recall that these experiments start from a neural network initialization that, in the baseline case without privacy, leads to a *stable reconstruction trajectory*. In this way, we aim to evaluate the effectiveness of the reconstruction attacks without any optimization instability interfering with the results.

First, Figure 6 illustrates the progression of the *reconstruction process* over 200 iterations for a sample image when gradients from a traditionally trained model (without any privacy protection) are used. Starting from an image of random noise, the attack optimizer (L-BFGS in our case) progressively refines the *seed image* until obtaining a reconstruction that visually resembles the original. The final reconstruction quality in this *privacy-free* scenario can be observed in the leftmost image of Figure 7. While the reconstruction is not identical to the original, the attack successfully recovers a vi-
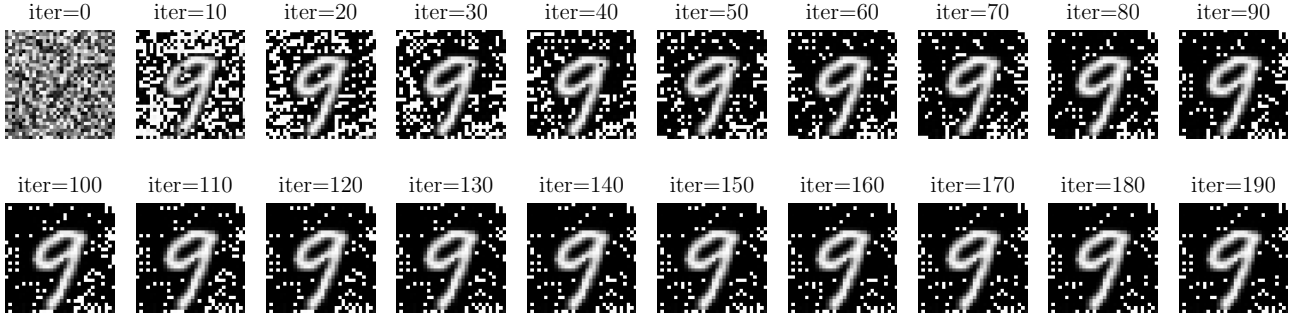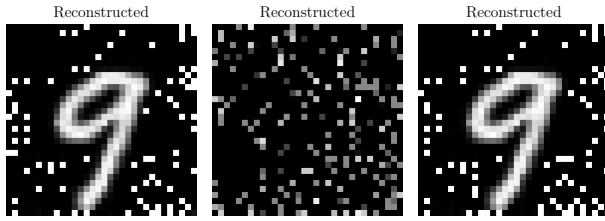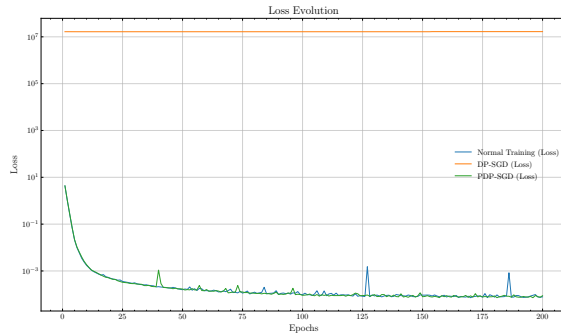
sually similar image with recognizable details and features, albeit containing clearly visible artifacts and noise.

The attack optimization behavior is quantitatively reflected in Figures 8 and 9. For traditional training (blue curve), the *gradient reconstruction loss* in Figure 8 decreases exponentially towards values close to zero, indicating that the attack optimizer successfully aligns the gradients of the seed image with the intercepted ones. At the same time, the SSIM index (Figure 9, blue curve) rises rapidly, reaching values near 1, confirming a high structural similarity between the original and reconstructed images. This clearly demonstrates the vulnerability of sharing unprotected gradients, as any potentially malicious attacker can recover sensitive information from training data.

When introducing DP-SGD during the victim model's *training* (using the same privacy budgets and hyperparameters as in the classification section, though results shown correspond to a representative ε, e.g., $\varepsilon = 8$, which already provides strong protection), the situation changes radically. The central image in Figure 7 displays the result of an attempted reconstruction under DP-SGD. It is evident that *the attack fails* to recover any meaningful information about the original image; the output is essentially indistinguishable noise. This failure is corroborated quantitatively: the reconstruction loss for DP-SGD (orange curve in Figure 8) stagnates at a much higher value, while the SSIM (orange curve in Figure 9) remains very low, indicating no structural similarity. This is an expected and desirable outcome, as it demonstrates the effectiveness of DP-SGD in perturbing gradients to obscure sensitive information over individual inputs, thereby preventing their reconstruction.
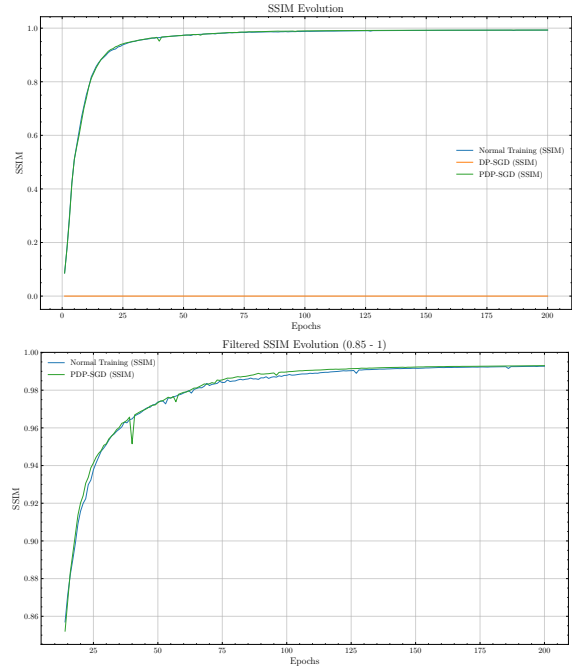
The most *surprising* and noteworthy finding arises when analyzing the results of attacks on gradients protected with PDP-SGD. Contrary to what might be expected from a mechanism designed to provide differential privacy, the image reconstructed under PDP-SGD (rightmost image in Figure 7) is *practically identical* to that obtained in the no-privacy scenario. The reconstruction loss and SSIM curves for PDP-SGD (green curves in Figures 8 and 9) follow trajectories almost indistinguishable from those of traditional training. The attack minimizes gradient distance and maximizes structural similarity just as effectively as if no protection were in place. This result is particularly curious when contrasted with the claims of the original work introducing PDP-SGD (Aguilera-Martínez & Berzal, 2024), which suggests that PDP-SGD, acting as explicit parameter-proportional regularization, not only offers differential privacy guarantees but may do so more efficiently than DP-SGD.

However, our gradient reconstruction experiments indicate that, at least under the tested configuration and in the context of image reconstruction, PDP-SGD does not provide *any discernible protection* against this type of attack, behaving

Figure 6. Reconstruction Progression of a Training Example from Model Gradients (*No Privacy*)

Figure 7. Final Reconstruction of a Training Instance from the Leaked Model Gradients (*From Left to Right:* No Privacy vs DP-SGD vs PDP-SGD)

Figure 9. Evolution of the *Structural Similarity Index* (SSIM) Across Attack Iterations (*Note:* The lower plot zooms in on the [0.85, 1] range to better visualize convergence in the region)

Figure 8. Evolution of the *Gradient Reconstruction Loss* Across Attack Iterations

very similarly to standard training. A possible explanation, though requiring deeper investigation, may lie in the nature of the regularization imposed by PDP-SGD. While it formally satisfies the definition of DP, the way it perturbs (or fails to sufficiently perturb) the information seems fundamentally different from the clipping and noise addition of DP-SGD. Because PDP-SGD injects noise proportional to parameters, for small parameters the effective perturbation may be minimal, or the noise structure may fail to obscure the crucial features exploited by the attack.

In short, image reconstruction results from gradients reveal a significant vulnerability in traditional training, where an attacker can recover sensitive data. DP-SGD, on the other hand, proves highly effective in *mitigating* this threat, at least in the context of image reconstruction; whereas PDP-SGD, while promising in terms of regularization and differential privacy, does not appear to provide effective protection against gradient reconstruction attacks in its current form. This raises serious questions about the applicability and effectiveness of PDP-SGD as a robust privacy-preserving mechanism in practical scenarios.

Finally, beyond the effectiveness of the tested attacks, the *sensitivity* of these methods to getting stuck in local optima or diverging during attack optimization is noteworthy.

While the literature documents GLA reconstruction attacks as robust and convergent; in our experiments, we frequently observe optimization trajectories diverging or stalling in local optima, despite using a second-order optimizer such as L-BFGS. Moreover, in multiple additional tests not documented in this paper, we consistently found that these attacks face serious limitations with relatively large or multi-channel images. In fact, Zhu et al. (2019) had already documented the *limitations* of these attacks for images larger than 64×64 pixels. In any case, we believe this does not diminish the significance of our results; rather, it highlights the need for further investigation into the effectiveness of gradient leakage reconstruction and other adversarial attacks in the context of differential privacy.

6. Conclusion

In this work, we have focused on the critical intersection between Federated Learning (FL), Gradient Leakage Attacks (GLA), and Differential Privacy (DP), with the main objective of evaluating the effectiveness of DP mechanisms, in particular DP-SGD and an alternative based on explicit regularization (PDP-SGD), as defenses against GLAs, as well as their impact on the utility of ML models.

The results obtained in the image classification task broadly confirm a well-known *privacy-utility trade-off* when employing DP-SGD. As stricter privacy constraints are imposed, model performance tends to degrade; although exceptions have been observed with simpler architectures, where DP-SGD may act as a beneficial form of regularization under certain privacy budgets. On the other hand, PDP-SGD emerges as an interesting alternative, improving or maintaining classifier performance in all cases.

With respect to the simulation of gradient leakage attacks on the MNIST dataset, the findings are somewhat revealing. As expected, in the absence of protection mechanisms, GLAs are capable of reconstructing recognizable images from intercepted gradients. The application of DP-SGD proves, however, to be a highly effective countermeasure, successfully preventing reconstruction and resulting in outputs that amount to little more than noise, with very low structural similarity scores.

By contrast, the most surprising and *concerning* result comes from the evaluation of PDP-SGD in the context of GLAs. Despite its formal differential privacy guarantees, PDP-SGD does *not* provide *discernible protection* against this type of attack in our experiments. Reconstructed images under PDP-SGD were practically indistinguishable from those obtained without protection, with SSIM values close to the *non-privacy* scenario. This suggests that, although a mechanism may satisfy the theoretical definition of (ε, δ)-DP, the specific way in which privacy is introduced

(e.g., parameter-proportional noise instead of *clipping* and noise addition) may not suffice to obfuscate the information exploited by certain attacks.

Ultimately, we emphasize the importance of not relying solely on theoretical privacy guarantees, but also on *empirically evaluating* the resilience of DP mechanisms against attack vectors that are relevant and specific to the application setting. The apparent disconnect between the formal protection of PDP-SGD and its practical ineffectiveness against GLAs, in our view, deserves deeper investigation.

Acknowledgement

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Aguilera-Martínez, F. and Berzal, F. Differential privacy regularization: Protecting training data through loss function regularization. *arXiv preprint arXiv:2409.17144*, 2024.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101– mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via dis-

tributed noise generation. In *Advances in cryptology-EUROCRYPT 2006: 24th annual international conference on the theory and applications of cryptographic techniques, st. Petersburg, Russia, May 28-June 1, 2006. proceedings 25*, pp. 486–503. Springer, 2006.

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

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Gong, H., Jiang, L., Liu, X., Wang, Y., Gastro, O., Wang, L., Zhang, K., and Guo, Z. Gradient leakage attacks in federated learning. *Artificial Intelligence Review*, 56 (Suppl 1):1337–1374, 2023.

Gupta, A. and Oakley, L. Cy 7790 lecture 17: Differentially private sgd and auditing dp-sgd. 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems*, 34:7232–7241, 2021.

Jagielski, M., Ullman, J., and Oprea, A. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kurakin, A., Ponomareva, N., Syed, U., MacDermed, L., and Terzis, A. Harnessing large-language models to generate private synthetic text. *arXiv preprint arXiv:2306.01684*, 2023.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Lomurno, E. and Matteucci, M. On the utility and protection of optimization with differential privacy and classic regularization techniques. In *International Conference on Machine Learning, Optimization, and Data Science*, pp. 223–238. Springer, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Mironov, I. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pp. 263–275. IEEE, 2017.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Sommer, D., Meiser, S., and Mohammadi, E. Privacy loss classes: The central limit theorem in differential privacy. *Cryptology ePrint Archive*, 2018.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Wei, W., Liu, L., Loper, M., Chow, K.-H., Gursoy, M. E., Truex, S., and Wu, Y. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

Yousefpour, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

Zhu, Y., Dong, J., and Wang, Y.-X. Optimal accounting of differential privacy via characteristic function. In *International Conference on Artificial Intelligence and Statistics*, pp. 4782–4817. PMLR, 2022.

A. Reconstruction Attack via Gradient Leakage

A.1. Attack in a Non-Differentially Private Setting
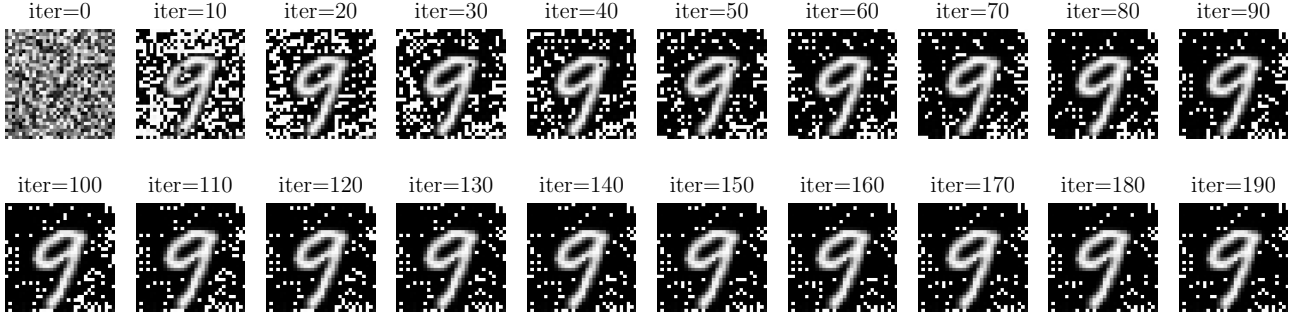

Figure 10. Reconstruction Progression of a Training Example from Model Gradients (**No Privacy**)

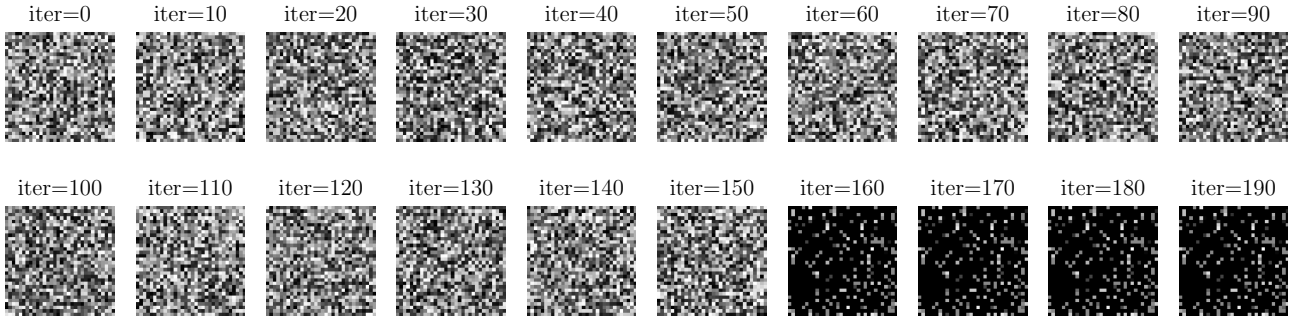## A.2. Attack in a Differentially Private Setting (DP-SGD)

Figure 11. Reconstruction Progression of a Training Example from Model Gradients (**DP-SGD**)

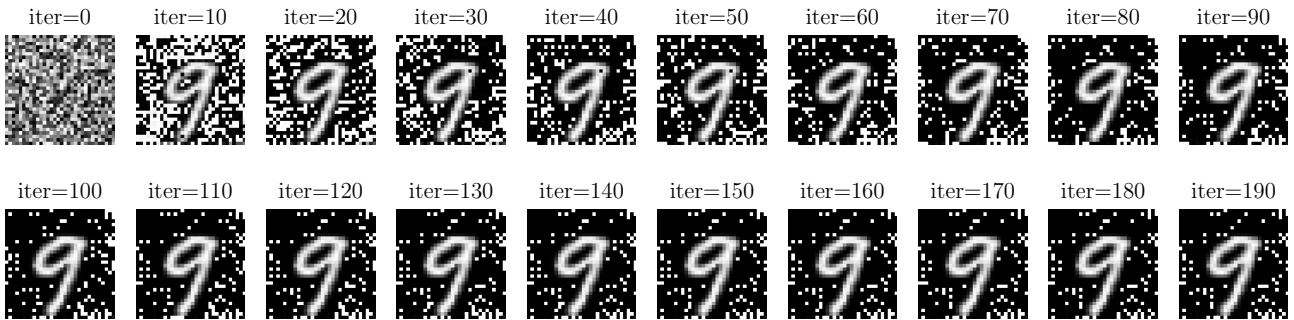## A.3. Attack in a Differentially Private Setting (PDP-SGD)

Figure 12. Reconstruction Progression of a Training Example from Model Gradients (**PDP-SGD**)