A Ethics Statement

This work demonstrates vulnerabilities in Federated Learning that could potentially allow malicious actors to exploit gradients to recover private batch labels. If deployed irresponsibly, such analytical attacks could seriously infringe on individuals' privacy and undermine trust in Federated Learning. However, we believe that with proper safeguards and oversight, the insights from this work can be used to improve accountability and integrity.

We recommend developers adopt differential privacy, trusted execution environments, and multi-party computation techniques to help cryptographically secure sensitive information like gradients and batch labels. Rigorous auditing and red team testing should be conducted before deployment to identify and patch vulnerabilities proactively. Policies and procedures governing the appropriate use of model insights should be established, clearly documenting purposes and ensuring transparency.

Furthermore, while we have developed proof-of-concept attacks in a simulated environment, we caution against reckless real-world testing which could cause serious harms. This work is meant to spur improved security practices, not enable adversaries. We advocate for an ethical approach centered on user empowerment and safeguarding rights. If deployed conscientiously with accountability checks, federated learning can offer privacy-preserving capabilities, but we must be vigilant against misuse. With care, insight and wisdom, we can work towards equitable and trustworthy AI.

B Related Label Recovery Attacks

We introduce the related label recovery attacks in this section, including iDLG (Zhao et al., 2020), GI (Yin et al., 2021), RLG (Dang et al., 2021), LLG (Wainakh et al., 2022), ZLG (Geng et al., 2021), and iLRG (Ma et al., 2023), and compare them with our proposed label attack.

iDLG. iDLG (Zhao et al., 2020) mathematically derives the relationship between the gradient of the cross-entropy loss w.r.t. the output logits $\nabla \boldsymbol{z} \in \mathbb{R}^K$ and the ground-truth label $\boldsymbol{y} \in \mathbb{R}^K$, which satisfies $\nabla \boldsymbol{z} = \boldsymbol{p} - \boldsymbol{y}$. This relationship reveals that:

- ∇z_j is negative ($\nabla z_j \in [-1, 0]$) for the input samples belonging to class j,
- ∇z_j is positive ($\nabla z_j \in [0, 1]$) for the samples belonging to other classes $k \neq j$.

Since $\nabla \boldsymbol{z}$ is unavailable in FL, iDLG uses the gradient of the cross-entropy loss w.r.t. the weight $\nabla \boldsymbol{W}$ in the last fully connected layer to estimate $\nabla \boldsymbol{z}$. If the non-negative activation function (e.g., ReLU or Sigmoid) is used in the model, $\nabla \boldsymbol{W}$ has the same sign as $\nabla \boldsymbol{z}$. By summing up the rows of $\nabla \boldsymbol{W}$, the negative row implies the ground-truth class of the target batch. However, iDLG is only applicable to single-batch training and non-negative activation functions.

GI. GI (Yin et al., 2021) follows the main idea of iDLG and extends it to the mini-batch training scenario. GI observes that for an untrained model, the negative values in gradient $\nabla \boldsymbol{z}$ possess larger magnitudes than the positive values, that is $\nabla z_j^- \gg \nabla z_j^+$. This observation indicates that when a training sample of class j is in the target batch, $\sum_{n=1}^{B} \nabla z_j^{(n)}$ is highly probable to be negative, where B is the batch size. Therefore, instead of summing up the rows of $\nabla \boldsymbol{W}$, GI obtains the minimum value in each row of $\nabla \boldsymbol{W}$ and then selects the rows with the minimum B values as the recovered classes of the target batch. However, GI is only applicable to non-repeating classes in the target batch and non-negative activation functions.

RLG. According to iDLG, for a sample belonging to class j, ∇z_j can be distinguished from $\nabla z_{k \neq j}$ by its sign. Since $\nabla \boldsymbol{z}$ can be estimated by $\nabla \boldsymbol{W}$, RLG (Dang et al., 2021) proposes to recover the ground-truth classes of the target batch by distinguishing each row of $\nabla \boldsymbol{W}$ from the other rows. RLG first decomposes $\nabla \boldsymbol{W}^\top$ into $\boldsymbol{P \Sigma Q}$, where $\boldsymbol{P} \in \mathbb{R}^{M \times S}$ and $\boldsymbol{Q} \in \mathbb{R}^{S \times K}$ are orthogonal matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{S \times S}$ is a diagonal matrix. For each column $\boldsymbol{q}_j$ of $\boldsymbol{Q}$, j corresponds to the target index c if a hyperplane can be found to separate $\boldsymbol{q}_j$ from the other columns. They transform the problem into finding a classifier to separate $\boldsymbol{q}_{j=c}$ from $\boldsymbol{q}_{j \neq c}$ through linear programming. Although RLG is suitable for all activation functions, it only applies to non-repeating classes in the batch.

ZLG. The aforementioned attacks exploit the distinguishability of corresponding rows in $\nabla \boldsymbol{W}$, where the target class is located, to recover the ground-truth classes of the training batch. Since $\nabla \boldsymbol{W} = \nabla \boldsymbol{z} \cdot \boldsymbol{o}^\top = (\boldsymbol{p} - \boldsymbol{y}) \cdot \boldsymbol{o}^\top$, ZLG (Geng et al., 2021) presents to restore batch labels by estimating the posterior probabilities $\boldsymbol{p} \in \mathbb{R}^K$ and input features $\boldsymbol{o} \in \mathbb{R}^M$ of the last layer. ZLG assumes that the summations of different features $\sum_{i=1}^{M} o_i$ are approximately equal to each other, that is $\hat{O} \approx \sum_{i=1}^{M} o_i^{(n)}$ for $n \in [1, B]$. By estimating $\hat{O}$ and $\hat{\boldsymbol{p}}$ from dummy data or auxiliary data, ZLG can restore the number of samples in each class j as follows:

$$\lambda_j = B \left(\hat{p}_j - \frac{1}{\hat{O}} \sum_{i=1}^{M} \nabla \boldsymbol{W}_{j,i} \right).$$

LLG. Similar to ZLG, LLG (Wainakh et al., 2022) rewrites $\nabla \boldsymbol{W}$ as $\nabla \boldsymbol{W} = -\boldsymbol{y} \cdot \boldsymbol{o}^\top + \boldsymbol{p} \cdot \boldsymbol{o}^\top$. For each class j, the restoration problem is formulated as $\sum_{i=1}^{M} \nabla \boldsymbol{W}_j = \lambda_j m + s_j$, where λ_j is the number of labels, m is the impact factor related to the input features, and s_j is a class-specific offset caused by misclassification. Instead of directly estimating the posterior probabilities $\boldsymbol{p}$ and input features $\boldsymbol{o}$, LLG embeds this information into the gradient and indirectly estimates m and s_j. By fitting dummy data or auxiliary data into the model and producing multiple sets of gradients, LLG then restores the class-wise labels in the target batch.

iLRG. iLRG (Ma et al., 2023) exploits both gradient $\nabla \boldsymbol{W}$ and gradient $\nabla \boldsymbol{b}$ of the bias terms in the last layer to recover the batch labels. According to $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{o} + \boldsymbol{b}$, it is easy to derive that $\nabla \boldsymbol{b} = \nabla \boldsymbol{z}$. Hence, it only needs to estimate the post-softmax probabilities $\boldsymbol{p}$ to recover the batch labels $\boldsymbol{y}$ through the conclusion $\nabla \boldsymbol{b} = \boldsymbol{p} - \boldsymbol{y}$. iLRG first restores the batch averaged features $\bar{\boldsymbol{o}}$ from $\nabla \boldsymbol{W} / \nabla \boldsymbol{b}$ (Geiping et al., 2020) and then calculates the posterior probabilities $\hat{\boldsymbol{p}}$ from $\bar{\boldsymbol{o}}$. For each class j, iLRG regards $(p_j - 1)$ and p_j as the coefficients and constructs these coefficients into a matrix $\boldsymbol{A}$. Then it can solve the label occurrence vector $\boldsymbol{\lambda}$ from the equation $\boldsymbol{A}\boldsymbol{\lambda} = \nabla \boldsymbol{b}$.

Our Attack. In terms of implementation, our attack leverages the gradient $\nabla \boldsymbol{b}$ and the estimated posterior probabilities to recover the batch labels. At a finer granularity, we divide the probabilities into positive and negative ones for each class j, which are denoted as $\boldsymbol{p}_j^+$ and $\boldsymbol{p}_j^-$, respectively. The samples belonging to class j output the $\boldsymbol{p}_j^+$, while the samples belonging to other classes $k \neq j$ output the $\boldsymbol{p}_j^-$. Based on the observation that the positive (negative) samples of a class j have approximate probability distributions, we can estimate the posterior probabilities of the target batch from an auxiliary dataset. The estimated positive and negative probability of the jth class are denoted as $\hat{p}_j^+$ and $\hat{p}_j^-$, respectively. Combined with our theoretical deductions, we can directly restore the number of labels λ_j for each class j as Equation (9).

C Definition and Proofs

C.1 Focal Loss in Multi-class Classification

According to the derivation of the binary Focal Loss in (Lin et al., 2017), we extend it into the multi-class scenarios. In a multi-class classification task using Cross-entropy (CE) Loss, the CE loss can be written as follows:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{i=1}^{K} y_i \log(p_i) = \begin{cases} -\log(p_1) & \text{if } y_1 = 1 \\ -\log(p_2) & \text{if } y_2 = 1 \\ \vdots & \\ -\log(p_K) & \text{if } y_K = 1, \end{cases}$$

where $\mathbf{y}$ is the one-hot embeded label.

For any class i, we use p_t to represent the confidence degree of the model's prediction as the following:

$$p_t = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{otherwise,} \end{cases}$$

where $t = i$. To be consistent with the original Focal Loss in (Lin et al., 2017), we use t to represent the class index instead of i, and t is actually identical to i.

In order to solve class imbalance, Focal Loss assigns an auto-determined weight $(1-p_t)^\gamma$ and a pre-determined weight α_t to each class t. Finally, we define the Focal Loss for multi-class classification tasks as:

$$\mathcal{L}_{\text{FL}}(p_t) = -\sum_{t=1}^{K} \alpha_t (1-p_t)^\gamma \log(p_t).$$

Some important points we would like to emphasize:

- **Difference in probabilities**: p_i is the post-softmax probability, while p_t represents the automatic determined confidence of the input at class i, where $t = i$.
- **Mechanism of weight**: For an easy-to-learn sample, p_t might be close to the target label. So, Focal Loss assigns a small coefficient $(1-p_t)^\gamma$ as the weight. However, for a hard-to-learn sample, p_t may be close to 0. Then $(1-p_t)^\gamma$ a relatively large weight to enhance the ratio of these samples in the total loss.
- **Summation sign**: Since the binary Focal Loss (Lin et al., 2017) just has one output, the summation is not necessary. In the multi-class case, we use the summation to cover all the classes $t \in [1, K]$, and aim to derive a general conclusion in Theorem 1.
- **Why Focal Loss**: To the best of our knowledge, Focal Loss has the general form in the cross-entropy loss variants and it can be converted to CE loss or BCE loss by setting different α and γ. We aim to derive a general form of label leakage from gradients, so we choose the Focal Loss.

C.2 Supplementary Definitions

In a multi-class classification problem, each instance in the dataset belongs to one of several classes. Let's denote the set of classes as K and a particular class of interest as $k \in K$. In this context, we can define positive and negative samples for class k.

- **Positive Samples** (X_k^+): The positive samples of class k satisfy that: $X_k^+ = \{x_i : y_i = k\}$, where x_i is the input and y_i is the corresponding label.
- **Negative Samples** (X_k^-): Similarly, the negative samples of class k satisfy that: $X_k^- = \{x_i : y_i \neq k\}$

According to the positive and negative samples, we can then get the positive and negative probability for class k.

- **Positive Probability** (p_k^+): When a positive instance is fed into the model, the predicted probability of class k is termed the positive probability. Since the Softmax activation function is used in the output layer, the output posterior probability p^+ is a vector of length k. Therefore, the positive probability for class k can be expressed as p_k^+.
- **Negative Probability** (p_k^-): Similarly, when a negative sample is input into the model, the kth element of the output probability vector represents the negative probability, denoted as p_k^-. It's essential to note that any negative sample associated with the other $(K-1)$ classes contributes to p_k^-.

When using an auxiliary dataset to estimate the probabilities of the target training batch in FL, we denote the estimated positive and negative probabilities as $\hat{p}_k^+$ and $\hat{p}_k^-$, respectively.

In a batch size of B, we aim to recover the labels of each instance within the batch, i.e., $\mathbf{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(B)}]$. As this is a multi-class classification problem, the ground-truth labels $\mathbf{y}$ can also be represented by the occurrences of each class: $\mathbf{y} = [n_1, n_2, \cdots, n_K]$, where n_k is the number of samples belonging to class k and K is the number of total classes.

- **Class-wise Labels**: The class-wise labels can be defined as: $n_k = \sum_{i=1}^{B} \delta(y^{(i)} = k)$. Here, n_k is the number of samples belonging to class k, B is the batch size, $y^{(i)}$ is the true class label of the ith instance in the batch, and $\delta(\cdot)$ is the Kronecker delta function, which equals 1 if the condition inside is true and 0 otherwise.

C.3 Proof of Theorem 1

Theorem 3 (Gradient of Focal Loss)**.** *For a K-class classification task using the **focal loss** function and **Softmax** activation, we can derive that the gradient of logit z_j as follows:*

$$\nabla_{z_j}\mathcal{L}_{FL} = \sum_{t=1}^{K} \Phi(\alpha_t, p_t, \gamma) \cdot (p_j - \delta_{tj}),$$

where $\Phi(\alpha_t, p_t, \gamma) = \alpha_t(1-p_t)^\gamma \left(1 - \gamma \frac{p_t \log p_t}{1-p_t}\right)$ and $\forall t \in K$, we have $\Phi(\alpha_t, p_t, \gamma) \geq 0$. Besides, δ_{tj} is the Kronecker delta, which equals 1 if $t = j$ and 0 otherwise.

Proof. According to Equation (1), we substitute the last p_t with its Softmax formula $p_t = \frac{e^{z_t}}{\sum_{k=1}^{K} e^{z_k}}$, and obtain the transformed focal loss function:

$$\begin{aligned}\mathcal{L}_{\text{FL}} &= -\sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma \log \frac{e^{z_t}}{\sum_{k=1}^{K} e^{z_k}} \\ &= \sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma \log \sum_{k=1}^{K} e^{z_k} - \sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma z_t.\end{aligned}$$

Let $\hbar = (1-p_t)^\gamma$, then we can deduce the gradient of logit z_j as follows:

$$\begin{aligned}\nabla_{z_j}\mathcal{L}_{\text{FL}} &= \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} \log \sum_{k=1}^{K} e^{z_k} + \sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma p_j - \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} z_t - \alpha_j(1-p_j)^\gamma \\ &= \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} \left(\log \sum_{k=1}^{K} e^{z_k} - z_t \right) + \sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma (p_j - \delta_{tj}) \\ &= \sum_{t=1}^{K} \alpha_t(1-p_t)^\gamma \left(1 - \gamma \frac{p_t \log p_t}{1-p_t}\right)(p_j - \delta_{tj}) \\ &= \sum_{t=1}^{K} \Phi(\alpha_t, p_t, \gamma) \cdot (p_j - \delta_{tj}).\end{aligned}$$

□

C.4 Proof of Theorem 2

Theorem 4 (Label Recovery Attack)**.** *For the attacker with an **auxiliary dataset**, he can recover the class-wise labels λ_j of the target batch according to the averaged gradient $\overline{\nabla b}_j$ and the estimated posterior probabilities $\hat{p}_j^+$ and $\hat{p}_j^-$ as follows:*

$$\lambda_j = B \cdot \frac{(\hat{p}_j^- - y_j^-) - \overline{\nabla b}_j / \hat{\varphi}_j}{(\hat{p}_j^- - y_j^-) - (\hat{p}_j^+ - y_j^+)},$$

where y_j^+ and y_j^- are the pre-set label embeddings of class j, $\hat{\varphi}_j = \frac{1}{\tau}\Phi(\alpha_j, \hat{p}_j^+, \gamma)$ is an coefficient related to the jth class, and B is the batch size.

Proof. Since $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, we can deduce that $\nabla \boldsymbol{b} = \nabla \boldsymbol{z}$ and $\overline{\nabla b}_j = \overline{\nabla z}_j$. We expand the averaged gradient $\overline{\nabla z}_j$ as a summation of B terms and replace the posterior probability $p_j^{(n)}$ of each sample

n with its estimated probabilities $\hat{p}_j^+$ and $\hat{p}_j^-$. Because $\Phi(\alpha_j, p_j^{(n)}, \gamma)$ is only related to the positive samples of the jth class, we can replace $p_j^{(n)}$ with $\hat{p}_j^+$. So we have:

$$\hat{\varphi}_j = \frac{1}{\tau}\Phi(\alpha_j, \hat{p}_j^+, \gamma) = \frac{1}{\tau}\alpha_j(1-\hat{p}_j^+)^\gamma \left(1 - \gamma\frac{\hat{p}_j^+ \log \hat{p}_j^+}{1-\hat{p}_j^+}\right).$$

Assume that the first λ_j samples belong to the jth class, and the rest $(B-\lambda_j)$ samples belong to other classes. Then from the first row of Table 1, we can derive that:

$$\begin{aligned}\overline{\nabla b}_j = \frac{1}{B}\sum_{n=1}^{B}\nabla b_j^{(n)} &= \frac{1}{B}\left\{\sum_{n=1}^{\lambda_j}\varphi_j^{(n)}\left[p_j^{(n)} - y_j^{(n)}\right] + \sum_{n=\lambda_j+1}^{B}\varphi_j^{(n)}\left[p_j^{(n)} - y_j^{(n)}\right]\right\} \\ &\approx \frac{1}{B}\left\{\lambda_j\,\hat{\varphi}_j\left(\hat{p}_j^+ - y_j^+\right) + (B-\lambda_j)\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right)\right\}.\end{aligned}$$

Therefore, we can finally derive that:

$$\lambda_j = B\cdot\frac{\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right) - \overline{\nabla b}_j}{\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right) - \hat{\varphi}_j\left(\hat{p}_j^+ - y_j^+\right)} = B\cdot\frac{(\hat{p}_j^- - y_j^-) - \overline{\nabla b}_j/\hat{\varphi}_j}{(\hat{p}_j^- - y_j^-) - (\hat{p}_j^+ - y_j^+)}.$$

□

C.5 Interpretation of Theoretical Analysis from Exponential Family

The standard form of exponential family distribution is:

$$f_{\mathrm{x}}(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left[\boldsymbol{\eta}(\boldsymbol{\theta})\cdot\boldsymbol{T}(\boldsymbol{x}) - A(\boldsymbol{\theta}) + B(\boldsymbol{x})\right].$$

We know that the likelihood is the joint probability of all samples occurring:

$$\begin{aligned}L(\boldsymbol{\theta};\boldsymbol{x}) &= f(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} f(\boldsymbol{x}_i|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{N}\exp\left[\boldsymbol{\eta}(\boldsymbol{\theta})\cdot\boldsymbol{T}(\boldsymbol{x}_i) - A(\boldsymbol{\theta}) + B(\boldsymbol{x}_i)\right] \\ &= \exp\left[\boldsymbol{\eta}(\boldsymbol{\theta})\cdot\sum_{i=1}^{N}\boldsymbol{T}(\boldsymbol{x}_i) - NA(\boldsymbol{\theta}) + \sum_{i=1}^{N}B(\boldsymbol{x}_i)\right].\end{aligned}$$

Now let's add logarithms to the likelihood function to get the log-likelihood function:

$$\begin{aligned}\ell(\boldsymbol{\theta};\boldsymbol{x}) &= \log L(\boldsymbol{\theta};\boldsymbol{x}) \\ &= \boldsymbol{\eta}(\boldsymbol{\theta})\cdot\sum_{i=1}^{N}\boldsymbol{T}(\boldsymbol{x}_i) - NA(\boldsymbol{\theta}) + \sum_{i=1}^{N}B(\boldsymbol{x}_i).\end{aligned}$$

For the exponential family, parameter $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ are reversible. Hence, the derivative of canonical parameter $\boldsymbol{\eta}$ is denote as:

$$\nabla_{\boldsymbol{\eta}}\ell(\boldsymbol{\theta};\boldsymbol{x}) = \sum_{i=1}^{N}\boldsymbol{T}(\boldsymbol{x}_i) - N\nabla_{\boldsymbol{\eta}}A(\boldsymbol{\theta}).$$

For the categorical distribution, we know that $\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}) = \boldsymbol{\theta}, \boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{x}$ and $\mathcal{L}_{\text{CE}}(\boldsymbol{\theta}, \boldsymbol{x}) = -\ell(\boldsymbol{\theta}; \boldsymbol{x})$. Then we can derive the final expression in the following:

$$\begin{aligned}\nabla_{\boldsymbol{\eta}} \mathcal{L}_{\text{CE}}(\boldsymbol{\theta}, \boldsymbol{x}) &= -\nabla_{\boldsymbol{\eta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}) \\ &= N \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}) - \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i) \\ &= N\boldsymbol{\theta} - \sum_{i=1}^{N} \boldsymbol{x}_i.\end{aligned}$$

From this formula, we can observe that the characteristic of the Exponential Family gives an efficient way for CE loss to calculate the loss of canonical parameter $\boldsymbol{\eta}$ by just doing a subtraction rather than complex calculations. In the multi-class scenario, after substituting $\boldsymbol{\theta}$ with post-softmax probability $\boldsymbol{p}$, $\boldsymbol{\eta}$ with logits $\boldsymbol{z}$, and $\boldsymbol{x}_i$ with target label $\boldsymbol{y}_i$, we can derive $\nabla_{\boldsymbol{z}} \mathcal{L}_{\text{CE}}$ (reduction in summation) as follows:

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{y}) = N\boldsymbol{p} - \sum_{i=1}^{N} \boldsymbol{y}_i.$$

This is what we mean that from an exponential family perspective, the combination of the softmax and cross-entropy would have reduced computation, but opened a back door to leaking labels from shared gradients.

D Ablation Studies

D.1 Attack on Focal Loss

In this section, we present additional experiments conducted on the Focal Loss. We mainly test the parameters of τ, γ and ε on an untrained model, and average the experiments over 10 trials. Focusing on temperature τ, we present several cases where the accuracy is not 100%. In addition, by varying γ and ε on these settings, the accuracies are not affected. The following table shows the ClsAcc and InsAcc of our attack on the Focal Loss with different temperatures τ (batch size=64, activation=ReLU).

Table 4: Label recovery accuracies on Focal Loss ($\gamma = 2$, $\epsilon = 0$).

$\boldsymbol{\tau}$	**MNIST (LeNet)**		**CIFAR-10 (ResNet-18)**		**CIFAR-100 (ResNet-50)**	
	Our ClsAcc	Our InsAcc	Our ClsAcc	Our InsAcc	Our ClsAcc	Our InsAcc
0.5	0.980	0.906	0.990	0.960	1.000	1.000
0.75	0.980	0.945	1.000	0.994	1.000	1.000
0.9	0.990	0.954	1.000	1.000	1.000	1.000
1.25	0.990	0.983	1.000	1.000	1.000	1.000
1.5	1.000	0.998	1.000	1.000	1.000	1.000

We have observed that the temperature parameter, τ, significantly impacts smaller datasets. When τ is smaller, the space of logits expands, complicating the estimation of batch posterior probabilities. Consequently, as τ decreases, label accuracy deteriorates. For the large datasets with more classes, such as CIFAR-10, the logit space is hardly influenced by changing different τ.

D.2 Auxiliary Dataset with Different Attacks

In this section, we compare the label recovery accuracies of ZLG, LLG and our attack on different settings. We use the training dataset to sample the target batch for label recovery and the validation dataset to simulate the auxiliary dataset. This ensures that the auxiliary dataset has the same data distribution as the training dataset. We compare the label recovery accuracies (ClsAcc and InsAcc) of different attacks on three groups of experiments, averaging results over 10 trials.

The first group of experiments set the batch size to 64 and the activation function to ReLU. The second group of experiments set the batch size to 256 and the activation function to ReLU. Finally, the third group of experiments set the batch size to 64 and the activation function to ELU. The results are presented in the following tables.

Table 5: Label recovery accuracies (batch size=64, activation=ReLU).

Dataset	**Model**	**ZLG**		**LLG**		**Our**	
		ClsAcc	InsAcc	ClsAcc	InsAcc	ClsAcc	InsAcc
MNIST	LeNet	**1.000**	0.996	**1.000**	**1.000**	0.995	0.955
CIFAR-10	LeNet	**1.000**	0.986	**1.000**	**0.988**	**1.000**	0.969
CIFAR-10	ResNet-18	**1.000**	0.916	**1.000**	0.914	**1.000**	**1.000**
CIFAR-100	ResNet-50	0.985	0.893	0.816	0.633	**1.000**	**1.000**

Table 6: Label recovery accuracies (batch size=256, activation=ReLU).

Dataset	**Model**	**ZLG**		**LLG**		**Our**	
		ClsAcc	InsAcc	ClsAcc	InsAcc	ClsAcc	InsAcc
MNIST	LeNet	**1.000**	0.977	**1.000**	**0.982**	**1.000**	0.951
CIFAR-10	LeNet	**1.000**	0.961	**1.000**	**0.970**	**1.000**	0.956
CIFAR-10	ResNet-18	**1.000**	0.905	**1.000**	0.919	**1.000**	**0.977**
CIFAR-100	ResNet-50	0.998	0.881	**1.000**	0.868	**1.000**	**1.000**

Table 7: Label recovery accuracies (batch size=64, activation=ELU).

Dataset	**Model**	**ZLG**		**LLG**		**Our**	
		ClsAcc	InsAcc	ClsAcc	InsAcc	ClsAcc	InsAcc
MNIST	LeNet	**1.000**	0.948	0.580	0.283	0.990	**0.969**
CIFAR-10	LeNet	**1.000**	0.923	0.990	0.743	**1.000**	**0.939**
CIFAR-10	ResNet-18	**1.000**	0.902	0.980	0.697	**1.000**	**0.972**
CIFAR-100	ResNet-50	0.985	0.897	0.845	0.603	**1.000**	**1.000**

From these tables, we can observe that our attack is more robust than ZLG and LLG on complex datasets like CIFAR-10 or CIFAR-100 trained with ResNet-18 or ResNet-50.

- For the LeNet model with ReLU activation, when the batch size increases from 64 to 256, all the attacks have a slight decrease in InsAcc. It is straightforward to understand that a larger batch size is much more difficult to attack than a smaller batch size.
- For the CIFAR-10 dataset trained on LeNet or ResNet-18, we can observe that the InsAcc of ZLG and LLG is lower than our attack. When changing the shallow LeNet model to the deep ResNet-18 model, the InsAcc of ZLG and LLG decreases significantly. However, the InsAcc of our attack is improved, which indicates that our attack is more robust than ZLG and LLG on deep neural networks.

- For the CIFAR-100 dataset, it is shown that our attack reaches 100% InsAcc on all the settings. We conclude that a dataset with more classes makes it easy for our attack to recover the labels. This is because the more classes there are, the estimated probabilities are more accurate and concentrated around the ground-truth value.
- Finally, when changing the activation function from ReLU to ELU, the InsAcc of LLG drops dramatically. This is because LLG is based on the assumption that the activation function is non-negative. When the activation function is changed to ELU, the assumption is violated, and the InsAcc of LLG drops significantly.

D.3 Auxiliary Dataset with Distribution Shift

To address concerns about significant distribution shifts, we conducted supplementary experiments, averaging results over 10 trials.

In the first set of experiments, we alternately used CIFAR-10 and CINIC-10 (Darlow et al., 2018) as the training and auxiliary datasets. CINIC-10 extends CIFAR-10 by incorporating downsampled ImageNet images. While there is some overlap in the distributions due to similar object categories, there is a notable bias in the data features. We employed an untrained VGG model with a batch size of 64.

Similarly, the second set of experiments involves alternating between MNIST and Fashion-MNIST as the training and auxiliary datasets. Despite both datasets having 10 classes, the objects they represent are entirely different. MNIST dataset contains a lot of handwritten digits, while Fashion-MNIST represents the article of clothing. Employing an untrained LeNet model with a batch size of 64, the results are presented in the table below.

Table 8: Distribution shift between training dataset and auxiliary dataset.

Training Dataset	**Auxiliary Dataset**	**Model**	**Our ClsAcc**	**Our InsAcc**
CIFAR-10	CINIC-10	VGG	1.000	1.000
CINIC-10	CIFAR-10	VGG	1.000	1.000
MNIST	FMNIST	LeNet	1.000	0.969
FMNIST	MNIST	LeNet	1.000	0.948

This phenomenon can be explained as follows: In the initial phase of model training, the model lacks the ability to differentiate between samples from each class, assigning similar output probabilities to all fitted samples (i.e., $1/K$). For different training datasets like Fashion-MNIST and MNIST, the model projects input figures to similar output probabilities with slight variations. This benefits our label recovery attack as it becomes easier to estimate the posterior probabilities of the target batch.

Given that Fashion-MNIST is more intricate than MNIST, utilizing MNIST as the auxiliary dataset poses challenges in accurately estimating the probability distribution of batch training samples. This difficulty accounts for the marginal decrease in InsAcc. However, since CIFAR-10 and CINIC-10 exhibit similarities, there is no difference in InsAcc. The outcomes of these experiments illustrate that if an attacker initiates an attack during the early training stages, having an auxiliary dataset with an identical distribution to the training dataset may not be essential.