IN THE NAME OF GOD


## Introduction to Machine Learning

# PROJECT PHASE-0

**Seyed Hamid Ghasemi (400109768)**

**Parsa Novrouzinezad (400102182)**

**Theory Question 1.** Name 3 aggregation methods you would propose for this algorithm. To your best knowledge, reason each method's strengths and weaknesses.

Based on the information provided in the Article, here are three aggregation methods that can be proposed for SISA training along with their strengths and weaknesses:

**1. Majority Vote:**

- **Strengths**: The majority vote is a simple and intuitive aggregation method. It works well when there is a clear majority decision among the constituent models, providing a robust prediction.

- **Weaknesses**: However, it may not capture the uncertainty or diversity of predictions when there is disagreement among constituent models. It also does not consider the confidence levels of individual models.

**2. Averaging:**

- **Strengths**: Averaging provides a more nuanced and probabilistic prediction by taking into account the predicted probabilities or scores from all constituent models. It can represent uncertainty in predictions

- **Weaknesses**: Averaging assumes that all constituent models have equal credibility or performance, which might not be true in practice if some models are consistently outperforming others. It may also give equal weight to poorly performing or unreliable models.

**3. Weighted Average:**

- **Strengths**: Weighted average allows assigning different weights to each model based on their performance measures (e.g., accuracy). This flexibility enables giving higher weight to more accurate or reliable models, resulting in improved prediction quality.

- **Weaknesses**: Determining appropriate weights can be challenging without prior knowledge of model performance metrics or validation data representing real-world scenarios.

It should be noted that these aggregation methods are general techniques used in ensemble learning approaches but may need further adaptation and evaluation specifically for SISA training considering its unique characteristics and requirements. Additionally, specific datasets and task complexities might influence the choice of aggregation method based on their specific properties and expected behavior during training and inference phases.

**Theory Question 2**. When will this method be inefficient to use? What do you suggest to mitigate this issue?

Based on the information provided in the Article, there are several scenarios where the SISA training method may be inefficient to use. Here are some potential situations and suggestions to mitigate these issues:

1. **Large Number of Shards**:

If the number of shards (S) is significantly large, it can lead to a decrease in accuracy due to reduced volumes of data available for each shard. This occurs when S is greater than a certain threshold that depends on various factors such as dataset size and task complexity

- **Mitigation**:

To address this issue, it is important to ensure that each shard has a sufficient number of data points for accurate training. If increasing the number of shards leads to significant accuracy degradation, reducing the number of shards or using alternative strategies such as federated learning can help mitigate this issue.

2. **Large Dataset Size:**

When dealing with a very large dataset, the computational cost of training multiple constituent models simultaneously can be prohibitive.

- **Mitigation:**

To address this issue, one possible solution is to utilize distributed computing frameworks or cloud-based infrastructures that can effectively handle the computational requirements of training multiple models in parallel.

3. **Limited Computational Resources:**

If there are limitations in terms of storage capacity or computation power, storing and managing multiple models and their associated parameters can become challenging.

- **Mitigation:**

Model compression techniques such as quantization or sparsity regularization can be applied to reduce storage requirements without significantly sacrificing performance. Additionally, utilizing distributed computing frameworks or cloud-based infrastructures with sufficient resources can help alleviate these constraints.

4. **Complex Learning Tasks with Insufficient Data:**

In situations where the learning task is complex and requires a significant amount of data for accurate modeling, dividing the data into shards (or subsets) may result in each constituent model becoming a weak learner due to limited data availability per shard.

- **Mitigation:**

To address this issue, increasing the number of samples per class within each shard during sharding can help improve accuracy by providing more representative training data for each constituent model.

5. **Heterogeneous Shards/Slices:**

When different shards/slices contain significantly different distributions of data points (e.g., imbalanced datasets), it may lead to biased constituent models and reduced overall performance.

- **Mitigation:**

To mitigate this issue, ensuring that each shard/slice has a relatively balanced distribution of data points from different classes can help avoid bias in individual constituent models and improve overall performance.

It's important to carefully analyze specific scenarios while considering factors like dataset characteristics, computational resources available, and task requirements when determining if SISA training is efficient and how best to leverage its benefits while mitigating potential inefficiencies accordingly. The suggestions provided above serve as general guidelines but may need further customization based on specific needs and constraints in practical applications.

**Theory Question 3.** What metric would you use to evaluate the performance of your unlearning algorithm? Reason your choice!

In the given Article, the metric used to evaluate the performance of the unlearning algorithm is accuracy. The choice of accuracy as a metric is reasonable given that the goal is to ensure that unlearning does not significantly degrade model performance. By measuring accuracy, we can assess how well the model performs on test data after unlearning specific training points.
Accuracy measures the proportion of correct predictions made by the model out of all predictions. It provides an overall assessment of prediction correctness and reflects both how well the model has learned from training data and its ability to generalize to unseen test data.

To compare the time run of different unlearning algorithms, we can use retraining time as a metric. This measures how long it takes for each algorithm to retrain models after unlearning requests. By comparing retraining times, we can assess which algorithm provides more efficient and faster unlearning capabilities.

It's important to consider both accuracy and retraining time when evaluating unlearning algorithms because they capture different aspects of performance. Accuracy reflects prediction correctness, while retraining time directly impacts computational efficiency and resource utilization during unlearning process.

By considering both metrics in evaluating an unlearning algorithm, we can gain insights into both prediction quality and computational efficiency trade-offs provided by different approaches.

**Theory Question 4.** Discuss the effect of the models' architecture on learning and unlearning time, and performance in both learning and unlearning.

The architecture of machine learning models significantly impacts both learning and unlearning time, as well as performance in both learning and unlearning. Here are some key points to consider:

- **Learning Time and Performance**
  - **Model Complexity:**

    More complex models, such as deep neural networks, typically require longer training times and more computational resources compared to simpler models like linear regression. This is because complex models have more parameters to optimize and require more iterations to converge.

  - **Model Size:**

    Larger models, whether complex or simple, generally require more time and resources to train. This is because they have more parameters to optimize and more data to process.

  - **Optimization Algorithms:**

    The choice of optimization algorithm can significantly influence learning time and performance. For example, stochastic gradient descent (SGD) is often faster and more efficient than batch gradient descent, but may not converge as quickly.

  - **Data Size and Distribution:**

    The size and distribution of the training data also impact learning time and performance. Larger datasets or datasets with complex distributions may require more time and resources to train.

- **Unlearning Time and Performance**
  - **Model Structure:**

    The structure of the model can affect unlearning time and performance. For example, models with simpler architectures, such as linear regression, may be easier and faster to unlearn compared to complex models like deep neural networks.

  - **Data Selection:**

    The selection of data for unlearning can significantly impact unlearning time and performance. For example, selecting only the most informative data points for unlearning can reduce the computational overhead and improve performance.

  - **Unlearning Algorithms:**

    The choice of unlearning algorithm can influence unlearning time and performance. For example, approximate unlearning methods, such as gradient update, may be faster and more efficient than exact unlearning methods, such as retraining from scratch.

- **Data Distribution:**

  The distribution of the training data can also impact unlearning time and performance. For example, datasets with complex distributions may require more time and resources to unlearn.

- **Performance in Learning and Unlearning**
  - **Model Accuracy:**

    The accuracy of the model during learning and unlearning is critical. Models with high accuracy during learning are more likely to have high accuracy during unlearning, and vice versa.

  - **Model Robustness:**

    The robustness of the model during learning and unlearning is also important. Models that are robust during learning are more likely to be robust during unlearning, and vice versa.

  - **Model Interpretability:**

    The interpretability of the model during learning and unlearning is crucial. Models that are interpretable during learning are more likely to be interpretable during unlearning, and vice versa.

The architecture of machine learning models has a significant impact on both learning and unlearning time, as well as performance in both learning and unlearning. Understanding these factors is essential for designing and implementing efficient and effective machine learning models that can learn and unlearn accurately and robustly.

**Theory Question 5.** Explain differentially private algorithms and their techniques for training a differentially private model.

Differentially private algorithms are designed to protect the privacy of individuals by ensuring that the output of a machine learning model does not reveal any information about a specific individual. This is achieved by introducing noise into the model's parameters or predictions, making it difficult for an attacker to infer the presence or absence of a specific individual in the training data.

- **Differentially Private Algorithms**
  - **Differential Privacy:**

    This is the most widely used technique for ensuring differential privacy. It works by introducing noise into the model's parameters or predictions, making it difficult for an attacker to infer the presence or absence of a specific individual in the training data.

  - **Local Differential Privacy:**

    This technique is used when the data is distributed across multiple parties, and each party wants to ensure that their data is private. It works by introducing noise into the data before it is shared with the other parties.

  - **Centralized Differential Privacy:**

    This technique is used when the data is collected centrally and then used to train a machine learning model. It works by introducing noise into the model's parameters or predictions, making it difficult for an attacker to infer the presence or absence of a specific individual in the training data.

- **Techniques for Training a Differentially Private Model**
  - **Noise Addition:**

    This technique involves adding noise to the model's parameters or predictions. The noise is typically drawn from a distribution that is symmetric around zero, such as the Laplace distribution.

  - **Clipping:**

    This technique involves clipping the model's parameters or predictions to a certain range. This helps to prevent the model from learning too much about the training data.

  - **Truncation:**

    This technique involves truncating the model's parameters or predictions to a certain range. This helps to prevent the model from learning too much about the training data.

  - **Differentially Private Stochastic Gradient Descent:**

This technique involves using a differentially private version of stochastic gradient descent to train the model. This involves adding noise to the gradient updates and clipping the model's parameters to a certain range.

- **Differentially Private Batch Gradient Descent:**

  This technique involves using a differentially private version of batch gradient descent to train the model. This involves adding noise to the gradient updates and clipping the model's parameters to a certain range.

- **Advantages and Disadvantages of Differentially Private Algorithms**
  - **Advantages:**
    - **Privacy:**

      Differentially private algorithms provide strong privacy guarantees, making it difficult for an attacker to infer the presence or absence of a specific individual in the training data.

    - **Robustness:**

      Differentially private algorithms are robust to attacks, making it difficult for an attacker to compromise the model's privacy.

  - **Disadvantages:**
    - **Accuracy:**

      Differentially private algorithms can introduce noise into the model's parameters or predictions, which can affect the model's accuracy.

    - **Computational Overhead:**

      Differentially private algorithms can introduce additional computational overhead, which can affect the model's training time.

Differentially private algorithms are designed to protect the privacy of individuals by ensuring that the output of a machine learning model does not reveal any information about a specific individual. These algorithms introduce noise into the model's parameters or predictions, making it difficult for an attacker to infer the presence or absence of a specific individual in the training data. While these algorithms provide strong privacy guarantees, they can also introduce additional computational overhead and affect the model's accuracy.

**Theory Question 6.** Explain the regularization and normalization techniques used in training a private model. Are these techniques similar to the method of adding noise to the model in differential privacy?

The regularization and normalization techniques used in training a private model are designed to ensure that the model is robust and generalizable, while also protecting the privacy of the individuals whose data is used to train the model.

- **Regularization Techniques**
  - **L1 and L2 Regularization:**

    These techniques involve adding a penalty term to the loss function to discourage large weights. L1 regularization (Lasso) adds a term proportional to the absolute value of the weights, while L2 regularization (Ridge) adds a term proportional to the square of the weights. This helps to prevent overfitting and reduce the impact of noisy or sensitive data, making the model more robust and less prone to memorizing individual data points.

  - **Dropout:**

    This technique involves randomly dropping out neurons during training to prevent overfitting. By randomly masking out a portion of the network during each training step, the model is forced to learn more robust and generalizable features, reducing its reliance on specific data points and making it less susceptible to privacy attacks.

  - **Early Stopping:**

    This technique involves stopping the training process early to prevent overfitting. By monitoring the model's performance on a validation set and halting training when the performance stops improving, the model is prevented from memorizing the training data and becomes more generalizable.

- **Normalization Techniques**
  - **Batch Normalization:**

    This technique involves normalizing the inputs to each layer by subtracting the mean and dividing by the standard deviation. This helps to reduce the internal covariate shift, making the model more stable and less sensitive to the specific values of the training data. By reducing the model's reliance on individual data points, batch normalization can improve the model's privacy and robustness.

  - **Layer Normalization:**

    Similar to batch normalization, this technique involves normalizing the inputs to each layer by subtracting the mean and dividing by the standard deviation. However, it operates on the individual samples rather than the entire batch, making it more suitable for tasks with variable-length inputs or when the batch size is small. Like batch normalization, layer normalization can also improve the model's privacy and robustness.

- **Comparison with Differential Privacy**

The regularization and normalization techniques used in training a private model are similar to the method of adding noise to the model in differential privacy, as they both aim to reduce the model's reliance on individual data points and improve its generalization.
However, there are some key differences:

- **Purpose:**

  The purpose of regularization and normalization techniques is to improve the model's generalizability and prevent overfitting, while the purpose of differential privacy is to provide a formal guarantee of privacy, ensuring that the model's output does not reveal information about any individual data point.

- **Mechanism:**

  Regularization and normalization techniques work by modifying the model architecture and the training process, while differential privacy involves adding carefully calibrated noise to the model's parameters or outputs.

- **Guarantees:**

  Regularization and normalization techniques provide guarantees about the model's generalization and robustness, but do not offer the same formal privacy guarantees as differential privacy. Differential privacy provides a quantifiable measure of privacy (the privacy budget ε) that can be tuned to achieve the desired level of privacy protection.

In summary, while regularization and normalization techniques can improve the privacy and robustness of a machine learning model, they do not provide the same formal privacy guarantees as differential privacy. The two approaches can be used in conjunction to create private and robust models, with differential privacy providing the formal privacy protection and regularization/normalization techniques improving the model's generalization and robustness.

## **Theory Question 7.** Find other techniques for training a private model.

here are some additional techniques for training a private model that do not involve differential privacy, regularization, or normalization:

- **Sharding, Isolation, Slicing, and Aggregation (SISA) Training:**

  This technique involves dividing the training data into multiple disjoint shards, training models in isolation on each shard, and then aggregating the predictions from each shard.
  The key advantage of SISA training is that it can significantly reduce the time required for unlearning a data point, as only the affected shard needs to be retrained.
  SISA training is designed to achieve the largest improvements for stateful algorithms like stochastic gradient descent for deep neural networks.

- **Statistical Query Learning:**

  This technique involves modeling unlearning in the statistical query (SQ) learning framework, where the learning algorithm queries the data in a predetermined order.
  By doing so, it is possible to know exactly how individual training points contributed to model parameter updates.
  However, this approach is not generalizable to more complex models, such as deep neural networks, which are trained using adaptive statistical query algorithms.

- **Decremental Learning:**

  This technique involves considering the problem of data erasure from a data protection regulation standpoint.
  It presents a formal definition of complete data erasure, which can be relaxed into a distance-bounded definition.
  Deletion time complexity bounds are provided, but it is unclear if the approach presented (Quantized k-Means) is applicable and scalable for all model classes.

- **Certified Removal Mechanisms:**

  This technique involves relaxing the definition of differential privacy to provide certificates of data removal.
  The mechanism by Guo et al. uses a one-step Newton update, but this introduces a small residue that is masked by adding noise.
  However, this approach is not generalizable to complex models, such as deep neural networks, and requires pretraining models on public data or using differentially-private feature extractors.

- **Stochasticity in Training:**

This technique involves introducing randomness in the training process, such as using small batches of data or randomizing the ordering of batches.
This can help to reduce the impact of individual data points on the model's parameters.

- **PAC Learning Theory:**

  This technique involves using the PAC learning theory to model the learning process.
  PAC learning theory suggests that the learned hypothesis is one of many hypotheses that minimize the empirical risk.
  This can help to provide guarantees about the model's performance and robustness.

## **Theory Question 8.** Explain three ways of generating training data for shadow models.

To train shadow models, the attacker needs training data that is distributed similarly to the target model's training data. We developed several methods for generating such data.

- **Model-based synthesis:**

If the attacker does not have real training data nor any statistics about its distribution, he can generate synthetic training data for the shadow models using the target model itself. The intuition is that records that are classified by the target model with high confidence should be statistically similar to the target's training dataset and thus provide good fodder for shadow models. The process and algorithm of this method are provided in the answer to question 9.

- **Statistics-based synthesis:**

This method leverages statistical information about the population from which the target model's training data was drawn. For example, the attacker may know the marginal distributions of different features. Synthetic training records for the shadow models can be generated by independently sampling the value of each feature from its own marginal distribution.

- **Noisy real data:**

The attacker may have access to some data similar to the target model's training data and can be considered a "noisy" version thereof. This can be simulated by introducing random perturbations to the data. For example, flipping the binary values of a certain percentage of features to create a noisy dataset for training shadow models.

**Theory Question 9.** One of the method for generating training data for shadow models is using the model to generate synthetic data. Explain the Algorithm of synthetic data generation.

The synthesis process runs in two phases: (1) search, using a hill-climbing algorithm, the space of possible data records to find inputs that are classified by the target model with high confidence; (2) sample synthetic data from these records. After this process synthesizes a record, the attacker can repeat it until the training dataset for shadow models is full.

**Algorithm 1** Data synthesis using the target model

```
procedure SYNTHESIZE(class : c)
    x ← RANDRECORD( )                          ▷ initialize a record randomly
    y*_c ← 0
    j ← 0
    k ← k_max
    for iteration = 1 ··· iter_max do
        y ← f_target(x)                        ▷ query the target model
        if y_c ≥ y*_c then                     ▷ accept the record
            if y_c > conf_min and c = arg max(y) then
                if rand() < y_c then           ▷ sample
                    return x                   ▷ synthetic data
                end if
            end if
            x* ← x
            y*_c ← y_c
            j ← 0
        else
            j ← j + 1
            if j > rej_max then                ▷ many consecutive rejects
                k ← max(k_min, ⌈k/2⌉)
                j ← 0
            end if
        end if
        x ← RANDRECORD(x*, k)                  ▷ randomize k features
    end for
    return ⊥                                   ▷ failed to synthesize
end procedure
```

First, fix class c for which the attacker wants to generate synthetic data. The first phase is an iterative process. Start by randomly initializing a data record x. Assuming that the attacker knows only the syntactic format of data records, sample the value for each feature uniformly at random from among all possible values of that feature. In each iteration, propose a new record. A proposed record is accepted only if it increases the hill-climbing objective: the probability of being classified by the target model as class c.

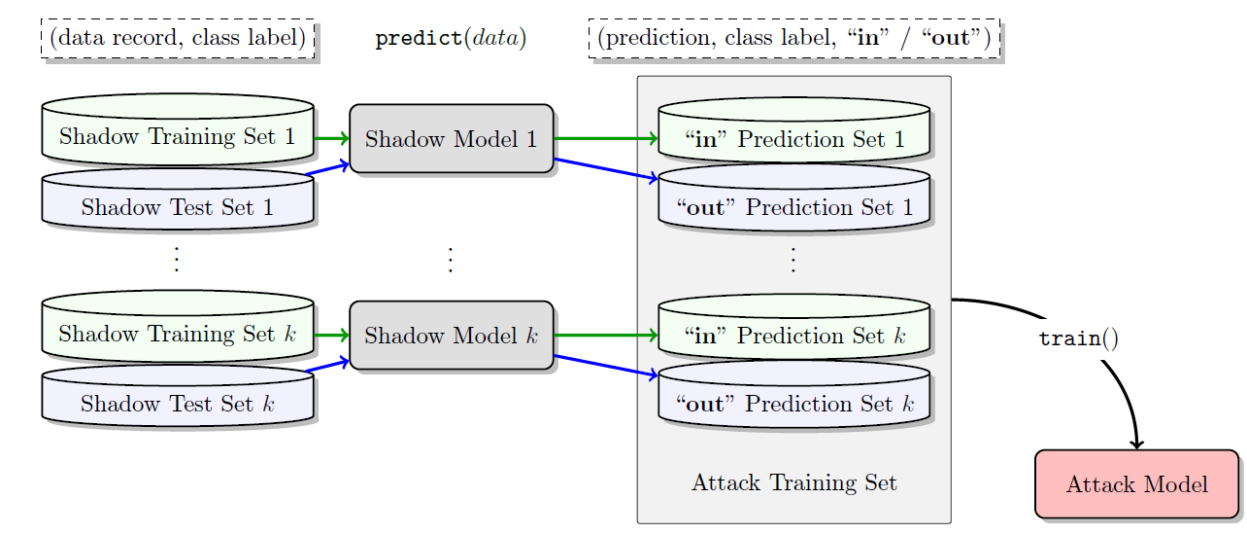
Each iteration involves proposing a new candidate record by changing k randomly selected features of the latest accepted record x*. This is done by flipping binary features or resampling new values for features of other types. We initialize k to kmax and divide it by 2 when $rej_{max}$ subsequent proposals are rejected. This controls the

diameter of search around the accepted record in order to propose a new record. We set the minimum value of k to $k_{min}$. This controls the speed of the search for new records with a potentially higher classification probability $y_c$.

The second, sampling phase starts when the target model's probability $y_c$ that the proposed data record is classified as belonging to class c is larger than the probabilities for all other classes and also larger than a threshold $conf_{min}$. This ensures that the predicted label for the record is c, and that the target model is sufficiently confident in its label prediction. We select such record for the synthetic dataset with probability $y_c^*$ and, if selection fails, repeat until a record is selected.

**Theory Question 10.** Explain how attack model is trained using shadow models.

query each shadow model with its own training dataset and with a disjoint test set of the same size. The outputs on the training dataset are labeled "in," the rest are labeled "out." Now, the attacker has a dataset of records, the corresponding outputs of the shadow models, and the in/out labels. The objective of the attack model is to infer the labels from the records and corresponding outputs.




For all (x; y) ∈ $D^{train}_{shadow\ i}$ compute the prediction vector y =$f^{i}_{shadow}$(x) and add the record (y; **y**; in) to the attack training set $D^{train}_{attack}$. Let $D^{test}_{shadiw\ i}$ be a set of records disjoint from the training set of the ith shadow model. Then, $\forall(x;\ y) \in D^{test}_{shadow\ i}$ compute the prediction vector y = $f^{i}_{shadow}$(x) and add the record (y; **y**; out) to the attack training set $D^{train}_{attack}$. Finally, split $D^{train}_{attack}$ into $c_{target}$ partitions, each associated with adifferent class label. For each label y, train a separate model that, given y, predicts the in or out membership status for x.
The attack model is essentially a binary classifier trained to predict membership status based on the target model's outputs.

**Theory Question 11.** Explain the effect of following concepts in accuracy of the attack model:
1. Effect of the shadow training data generated using the three methods you explained earlier.
2. Effect of the number of classes and training data per class.
3. Effect of overfitting.

- **Effect of the Shadow Training Data Generated Using the Three Methods**
  - **Model-based synthesis:**

    Generates high-quality synthetic data that closely resembles the target model's training data, resulting in higher accuracy of the attack model. This method ensures that the synthesized data is highly representative of the target data, thus leading to more effective attack models.
    Achieves a precision of 0.895, balancing between high precision and some vulnerabilities in underrepresented classes.

  - **Statistics-based synthesis:**

    This method may not capture the joint distribution of the features accurately, leading to moderate accuracy. The generated data might miss some of the complex relationships between features present in the real training data, resulting in less effective attack models compared to model-based synthesis.
    Precision drops to 0.795 but remains effective, demonstrating robustness even with less accurate data generation methods.

  - **Noisy real data:**

    If the noise level is not too high, this method can produce reasonably accurate shadow models. The generated data retains much of the structure of the original data, making the attack model effective. However, too much noise can degrade the accuracy of the shadow models and, consequently, the attack model.

- **Effect of the Number of Classes and Training Data per Class**
  - **Number of Classes:**
    - **Process**:

      The target model's number of output classes impacts how much information it leaks. Models with more output classes need to extract more distinctive features to classify inputs accurately.

    - **Accuracy Impact**:

      The more classes the target model has, the more signals about its internal state are available to the attacker. This makes the model more vulnerable to membership inference attacks. For example, CIFAR-100, with more classes, is more vulnerable than CIFAR-10

because the model needs to remember more about the training data, increasing the leakage of information.

- **Training Data per Class:**
  - **Process:**

    The amount of training data associated with each class affects the attack's precision.

  - **Accuracy Impact:**

    Sparse data per class can lead to lower accuracy as the attack model struggles to learn the distribution effectively. Conversely, a larger amount of training data per class can lower the attack precision for that class because the model does not overfit as much. Overfitting to smaller classes provides more distinctive patterns for the attack model to exploit.

- **Effect of Overfitting**
  - **Process:**

    Overfitting occurs when a model performs significantly better on its training data than on unseen test data. This typically happens when the model memorizes the training data rather than learning generalizable patterns

  - **Accuracy Impact:**

    Overfitting in the target model increases the success of membership inference attacks. Overfitted models tend to memorize the training data, making it easier for the attack model to distinguish between in and out samples. The model's confident predictions on training data versus less confident predictions on unseen data provide a clear signal for the attack model. However, overfitting is not the only factor contributing to vulnerability; the model's structure and type also play significant roles.
    Experiments show that attacks on overfitted models, such as CIFAR-10 and CIFAR-100, yield higher precision and recall. For instance, the test accuracy for overfitted neural-network models was 0.6 and 0.2 for CIFAR-10 and CIFAR-100, respectively, indicating high overfitting and thus high vulnerability to membership inference attacks.