Unlearning Inversion Attacks

Berkay Guler (ID: 13609520) Kashan Saeed (ID: 42921899)

June 13, 2025

1 Introduction

Machine unlearning (MU) is a recent concept that refers to removing the effect of a certain subset of training samples on model parameters [1, 2]. MU methods can be analyzed in two groups: (i) exact unlearning and (ii) approximate unlearning. The former is conceptually simple and involves retraining after data removal, leading to undesirable computational complexity. The latter is performed through several gradient updates and has been a thriving research area due to its feasibility compared to exact unlearning.

A typical use case for MU naturally arises in a scenario where users request the removal of their private data used during the training of a machine learning (ML) model. However, with two versions of the model avaliable, the original model and the unlearned model, MU results in a security vulnerability through which an adversary can gain sensitive information about the removed data. The authors of [3] analyze this vulnerability in a rather simplistic setting through two unlearning inversion attacks. (i) They demonstrate that an adversary can generate data samples with similar properties to the removed data sample(s) under white box access to the model. (ii) They also demonstrate that an adversary can also predict the removed data labels with black box access. The authors also propose and analyze the effectiveness of three simple defense mechanisms.

This project briefly summarizes the threat model and methodology of the aforementioned work. We aim to reimplement certain aspects of the work and verify the results. The reimplemented elements are detailed in the corresponding sections.

2 Threat Model

The goal of MU is to remove the influence of specific training samples from a trained ML model. Unlearning inversion attacks on the other hand are utilized to reveal information about the removed samples. The paper considers a machine learning-as-a-service (MLaaS) setting in which a service provider offers inference through a trained ML model exposed via a public API.

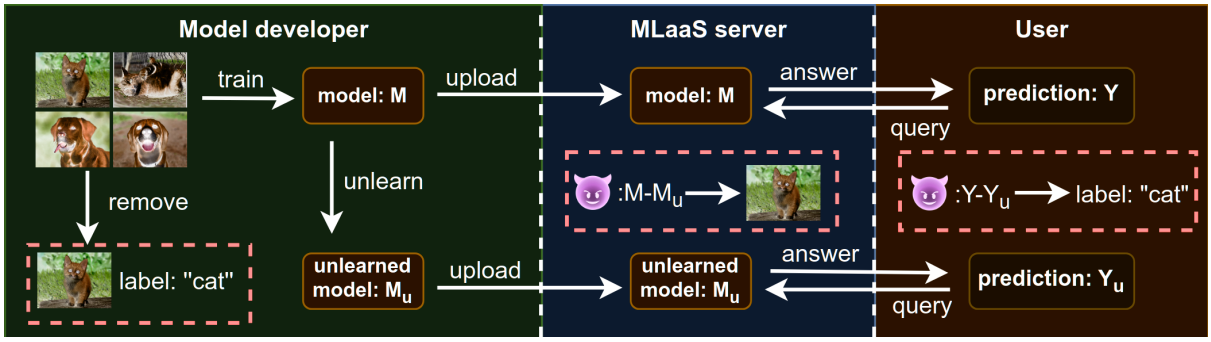

Figure 1: Overview of the threat model [3]

MLaaS Scenario This setting includes three entities: the model developer, the MLaaS server, and the users. The model developer is responsible for training the ML model. After training, the developer uploads the model to the MLaaS server, which hosts it and provides an API interface to end users. Users interact with the model through this API, sending input queries and receiving prediction results in response. The structure of this interaction is illustrated in Figure 1.

Machine unlearning occurs when a user requests their data to be removed from the trained model. The developer applies a selected unlearning technique, in the case of the original paper either exact unlearning or approximate unlearning, to eliminate the influence of the specified data subset, resulting in a modified model. The unlearned model is then uploaded to the server and made available through the same API.

Although unlearning is intended to improve privacy, the paper shows that this process can unintentionally leak information about the removed data. Specifically, it introduces unlearning inversion attacks, where an attacker compares the behavior of the original and unlearned models to infer information about deleted samples. The paper defines two adversarial settings.

Server as an Attacker A server-side attacker, who has white-box access to the original and unlearned model parameters, but no access to the training data or details of the unlearning method.

User as an Attacker A user-side attacker, who interacts with the model through the API and has black-box access. This attacker can only submit input queries and observe the output probabilities, making it a more constrained but realistic threat model.

Reimplemented Aspects

Our project focuses on the user-side attacker with black-box access. The adversary is assumed to be a standard user who can send queries to the model and observe the returned probabilities but has no access to model parameters or training data.

3 Methodology

The unlearning inversion attack framework proposed in the paper investigates how machine unlearning in MLaaS settings can leak sensitive information about deleted training data. These attacks rely on comparing the behavior of the original model before unlearning with the unlearned model after specific data samples have been removed.

The paper explores two types of attacks:

- Feature Recovery Attack: A white-box attack where the server estimates gradients from parameter differences to reconstruct the deleted input samples.
- Label Inference Attack: A black-box attack where the user infers the class label of the deleted data by analyzing changes in output probabilities assigned to different classes in a classification task.

In addition to evaluating the success of these attacks, the paper introduces several defense mechanisms designed to mitigate information leakage.

- Parameter Obfuscation: Adds Gaussian noise to gradients during unlearning, inspired by differential privacy.
- Model Pruning: Deletes a fraction of model parameters post-unlearning to limit memorization capacity.
- Fine-tuning: Fine-tunes the unlearned model on a held-out dataset to obscure the information contained in the unlearned model.

Each defense is evaluated in the paper in terms of how it affects the success rate of unlearning inversion attacks.

Reimplemented Aspects

Our implementation targets the black-box label inference attack scenario, where an adversary queries both the original and unlearned models through an API. We also implement model pruning as a defense mechanism and examine its impact on the success of label inference.

Label Inference Attack

In the black-box setting, the adversary has no access to the model's internal state but can query and observe output predictions assigned to each class. The attacker first constructs a probing dataset $\mathcal{D}_p = \{x'_i\}_{i=0}^{|\mathcal{D}_p|-1}$ where each x_i is assigned a target label y_t with high confidence. This is achieved by solving:

$$\arg\min_{x'} \; g(x', y_t) \quad \text{subject to } x' \in [0,1]^N$$

where

$$g(x', y_t) = \max_{i \neq y_t} [Z(x')]_i - [Z(x')]_{y_t}$$

and $[Z(x')]_i$ denotes the probability assigned to class i by the model. The optimization objective aims to maximize the confidence difference assigned to class y_t with all other classes $i \neq y_t$. The above optimization problem is solved with Zeroth Order Optimization (ZOO) technique introduced in [4], which utilizes an iterative procedure to obtain an approximate solution.

Each probing sample $x'_i \in \mathcal{D}_p$ is evaluated by both original and unlearned model, and the average difference in prediction vectors is computed as:

$$\Delta_p = \frac{1}{m} \sum_{i=1}^{m} \left(f_\theta(x_i') - f_{\theta_u}(x_i') \right)$$

where $f_\theta(.)$ and $f_{\theta_u}(.)$ are the original and unlearned models that output a probability distribution over all classes.

The attacker predicts the class with largest confidence change as the removed class, as the unlearned model becomes less confident for classes with deleted examples. Without loss of generality, the attacker can pick N different classes with the largest confidence changes to conclude that samples from those N classes are removed. Attack success is measured by the accuracy of correctly predicting the removed class.

Model Pruning

To reduce information leakage, we implement global unstructured model pruning following the methodology described in the paper. This defense involves removing a proportion p of the smallest-magnitude parameters in the unlearned model before deployment.

We apply pruning after retraining on $\mathcal{D}_u \setminus \mathcal{X}$ and before evaluating the model against label inference attacks. We use PyTorch's built-in pruning utilities to perform global pruning across all layers.

The original paper shows that larger pruning rates (e.g., $p = 0.8$ or $p = 0.9$) significantly reduce feature recovery success while sacrificing some validation accuracy. Our experiments assess whether pruning also degrades the accuracy of label inference, indicating improved privacy.

By evaluating both attack performance and accuracy retention, we explore the trade-off between privacy and utility introduced by model pruning.

4 Simulation Results

We evaluate the label inversion attack and model pruning defense mechanism on the CIFAR-10 [5] dataset. We split the entire dataset $\mathcal{D}$ into disjoint sets $\mathcal{D}_0$ and $\mathcal{D}_u$ with a 0.8/0.2 split ratio, corresponding to the public training dataset and the dataset with sensitive user information, respectively.

We train a convolutional neural network denoted as ConvNet on $\mathcal{D}_0$ and then fine-tune it on $\mathcal{D}_u$ to obtain the original model with parameters θ. We also fine-tune the model on $\mathcal{D}_u \setminus \mathcal{X}$ to obtain the unlearned model with parameters θ_u. We denote by p_i the proportion of the unlearned data with class label i in $\mathcal{X}$ to the total number of data samples with class label i in $\mathcal{D}_u$.

We evaluate the label inversion attack performance using the accuracy in predicting the labels of the removed data samples. We implement retraining as a method of unlearning. In our experiments, samples from only one data class are unlearned for the label inversion attack. We generate the probing samples for the label inference attack as explained in Section 3. Specifically, we generate 20 probing samples for each class label $i \in \{0, 1, \ldots, 9\}$. We experiment with $p_i \in \{0.003, 0.01, 0.1, 0.25, 0.5, 0.9\}$. We also note that $p_i = 0.003$ corresponds to unlearning 2 or 3 samples depending on the choice of i, while $p_i = 0.9$ leads to unlearning of around 900 images with the exact number changing with choice of i.

We experiment with pruning ratios $p \in \{0.1, 0.2, 0.5, 0.9\}$. We classify the defense successful if the adversary fails to predict the removed class correctly after pruning but predicts it correctly before pruning. We also report the change in model performance on the test set after pruning to observe if the model retains its discriminative capabilities after pruning.

We implemented our solution in Python and PyTorch and trained/fine-tuned more than 60 models on several GPUs over the span of this project. We referred to the authors' codebase[1] only for the implementation of ZOO method and ConvNet. We made our implementation, together with all the models and probing samples, publicly available[2].

5 Analysis and Discussion

The experimental results in Figures 2–4 reveal critical insights into the effectiveness of label inference attacks under different unlearning scenarios. We also include results for other experiments in appendix A.

Attack Effectiveness vs. Removal Proportion

The label inference attack success exhibits a strong correlation with the proportion of removed data. When minimal data is unlearned ($p_i = 0.003$), the attack shows limited effectiveness due to small confidence differences between original and unlearned models. However, as the removal proportion increases to $p_i = 0.25$ and $p_i = 0.9$, the target class becomes easily identifiable through pronounced negative confidence changes, while non-target classes show minimal variation.

This pattern demonstrates that the privacy vulnerability scales directly with the amount of data removed, suggesting that large-scale unlearning operations pose significant privacy risks in MLaaS environments.

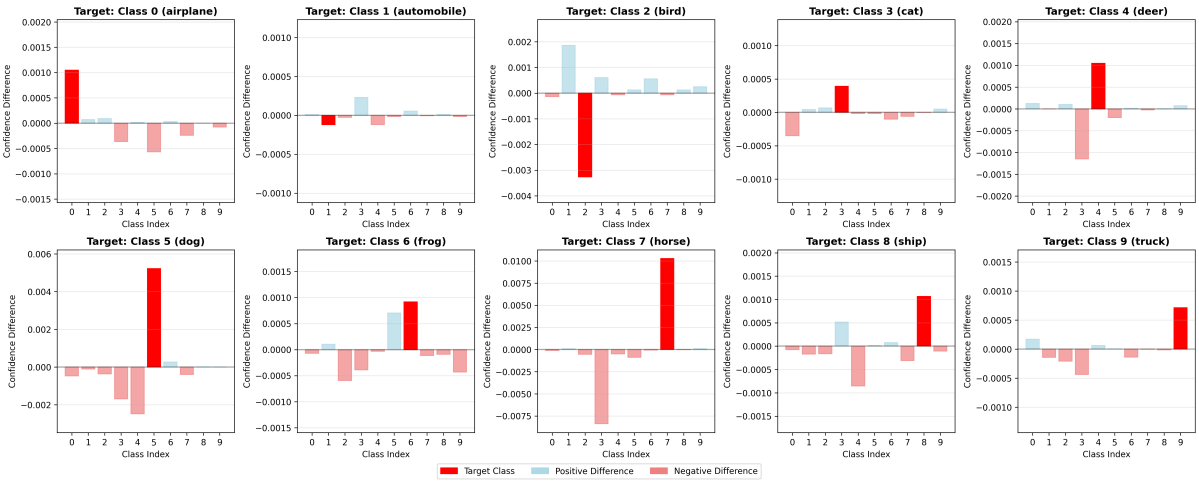

Figure 2: Change in Confidence Per Class with $p_i = 0.003, i = 0, 1, \ldots, 9$

[1] https://github.com/TASI-LAB/Unlearning-inversion-attacks
[2] https://github.com/BerkIGuler/UnlearningInversionAttacks

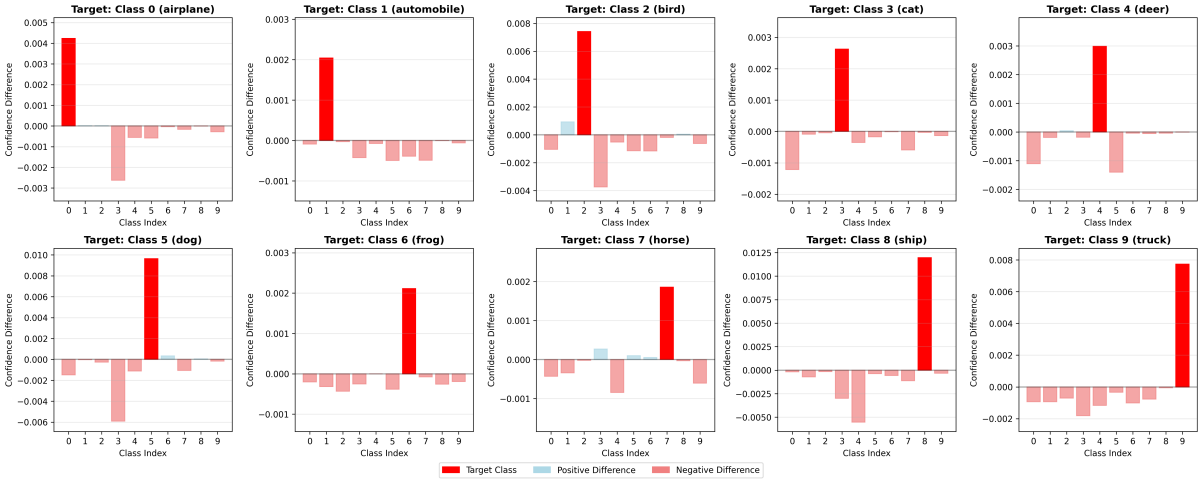

Figure 3: Change in Confidence Per Class with $p_i = 0.25, i = 0, 1, \ldots, 9$

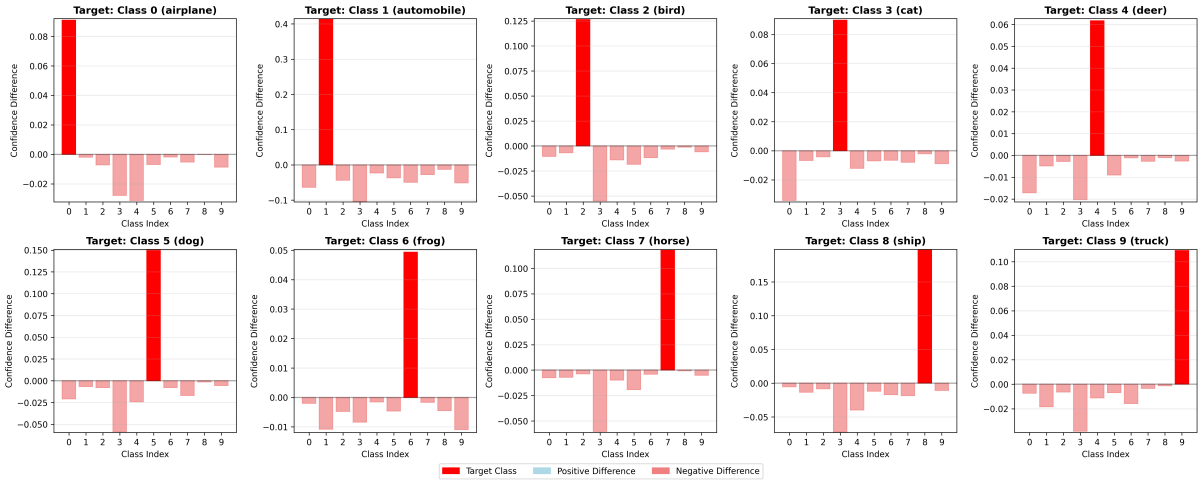

Figure 4: Change in Confidence Per Class with $p_i = 0.9, i = 0, 1, \ldots, 9$

Defense Mechanism Analysis

Figure 5 reveals a counterintuitive relationship between pruning intensity and defense effectiveness. While moderate pruning rates ($P = 0.1, 0.2$) fairly successfully defend against approximately 5% of the attacks with minimal accuracy loss, higher pruning rates ($P = 0.5, 0.9$) show dramatically reduced defense effectiveness with substantial accuracy degradation.

At $P = 0.9$, the defense success drops to nearly zero while accuracy decreases by approximately 75%. This suggests that excessive parameter removal may amplify remaining signals or create distinctive sparse patterns that actually facilitate attack success.

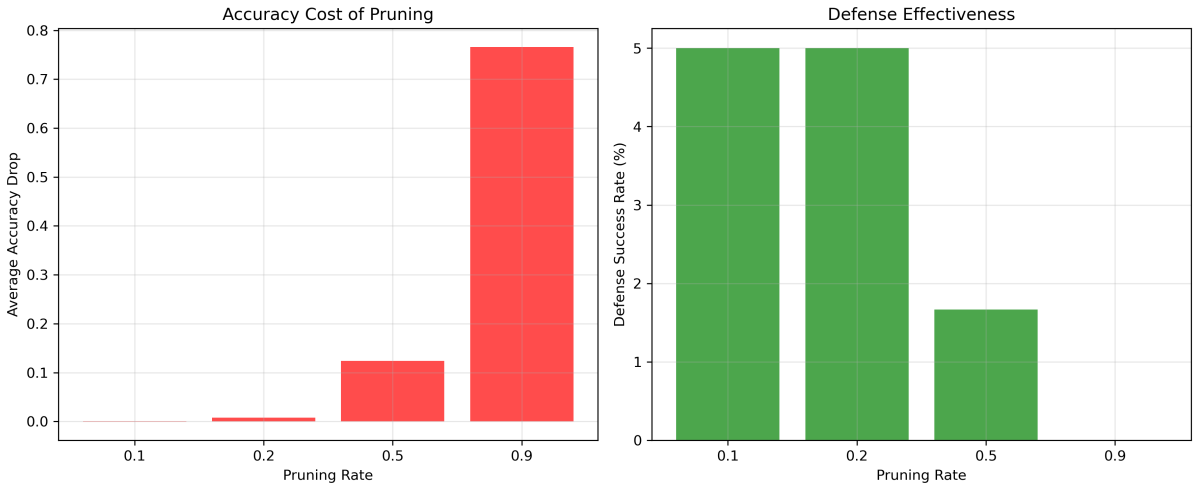

Figure 5: Effectiveness of Pruning and its Effect on Accuracy

6 Conclusion

In this project, we reimplemented model unlearning through retraining from the work *Learn What You Want to Unlearn* [3] and evaluated its vulnerability to black-box label inference attacks. We simulated a realistic MLaaS environment in which a user attempts to infer which class of data has been removed by comparing the outputs of original and unlearned models.

To strengthen privacy, we implemented global unstructured model pruning as a defense and explored its effectiveness. Our findings reveal that moderate pruning rates (0.1-0.2) provide a weak defense with minimal accuracy loss, while higher rates reduce privacy protection despite severe performance degradation.

Our experiments confirm that label inference attacks are highly effective when substantial data is removed, with attack success scaling directly with removal proportion. The counter-intuitive pruning results highlight the complex privacy-utility trade-offs in machine unlearning and emphasize the need for careful defense calibration in practical MLaaS deployments.

References

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020.

[2] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2150–2168, June 2024.

[3] Hongsheng Hu, Shuo Wang, Tian Dong, and Minhui Xue. Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning, 2024.

[4] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, CCS '17, page 15–26. ACM, November 2017.

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

APPENDIX

A Additional Experimental Results

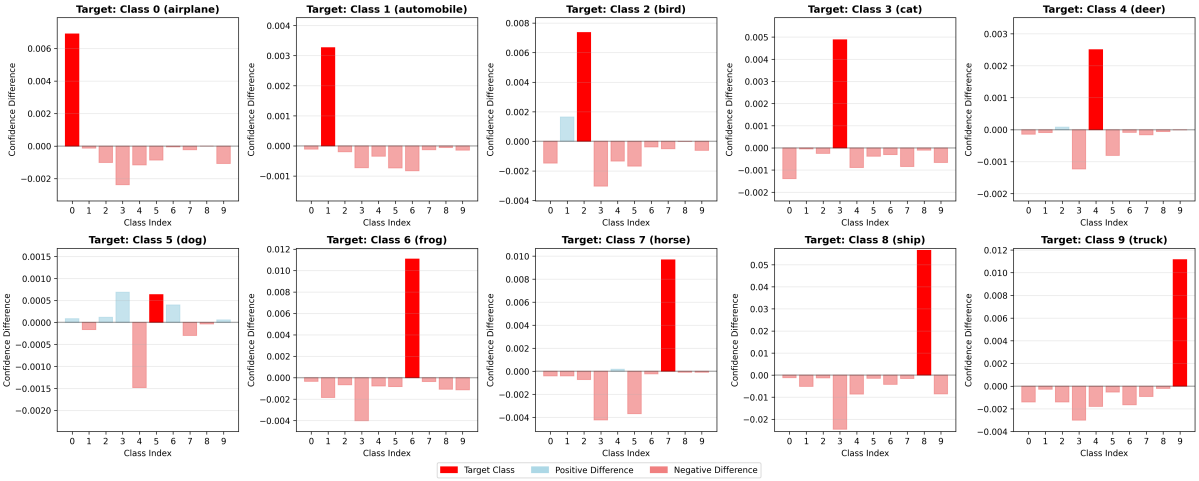

Figure 6: Change in Confidence Per Class with $p_i = 0.01, i = 0, 1, \ldots, 9$

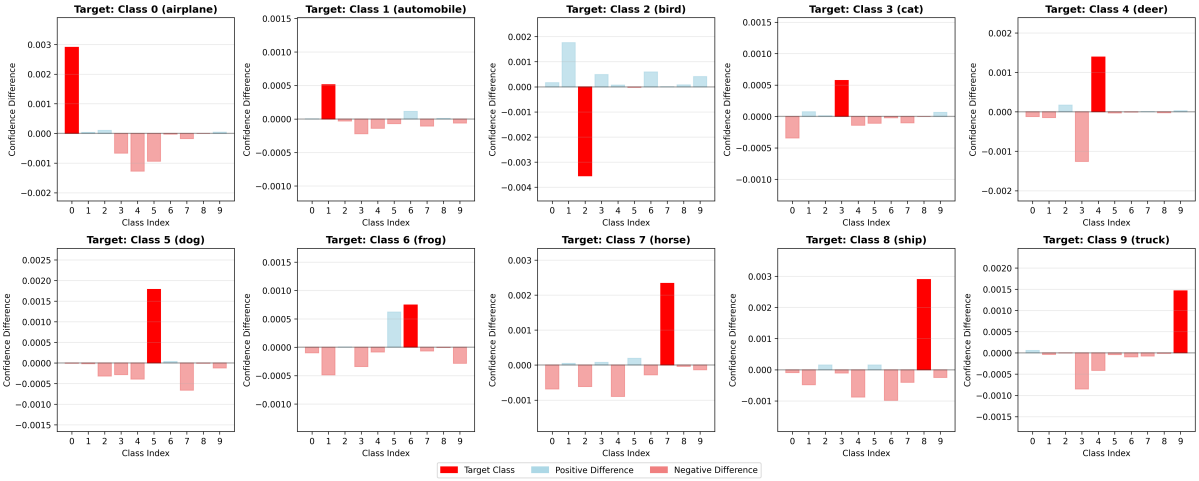

Figure 7: Change in Confidence Per Class with $p_i = 0.1, i = 0, 1, \ldots, 9$

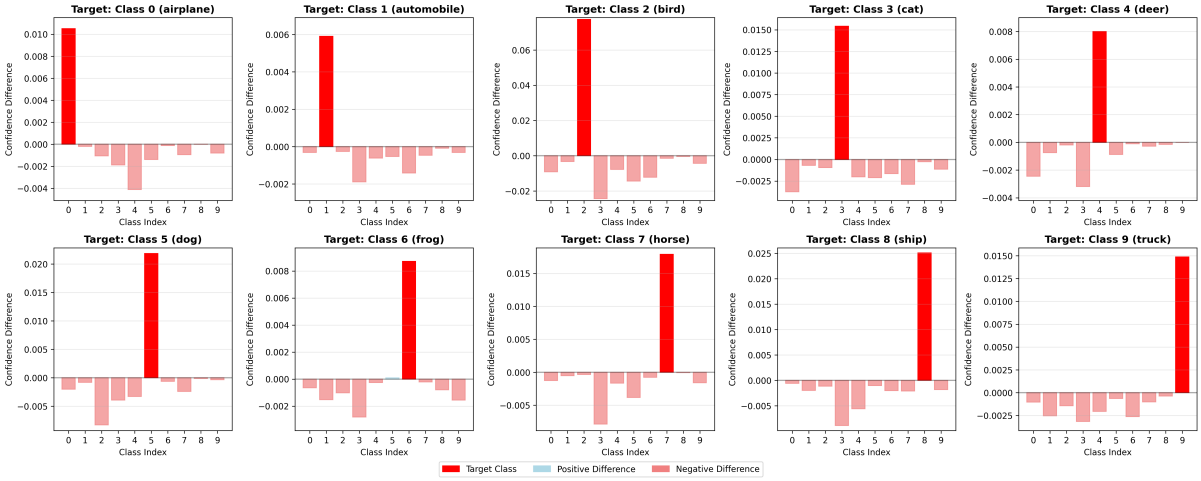

Figure 8: Change in Confidence Per Class with $p_i = 0.5, i = 0, 1, \ldots, 9$