A One-Shot vs Iterative Removal

If the definition of outlier depends on the other examples that are present, it is possible that removing $5{,}000$ outliers in one pass might somehow impact the results—because we might miss some new set of examples that become outliers only after we have trained the model for a subset of epochs.

Here, we experiment with an iterative removal approach. In particular, starting with the $50{,}000$ example dataset, we first run LiRA to find the easiest to attack samples then remove just the top 100 (instead of $5{,}000$) examples. We then repeat our experiment on the remaining $49{,}900$ example dataset; this gives us a *new* set of 100 examples to be removed from this dataset, which we do, giving us a dataset of $49{,}800$ examples. We repeat this procedure 50 times until we are left with a dataset of $45{,}000$ examples. We then plot in Figure 7 the ROC curve when attacking this dataset, and find that the results are almost completely identical to our initial experiments where we directly removed $5{,}000$ outliers in one shot. Upon investigation, we find that the reason this occurs is because over 80% of the examples removed by the 1-shot approach are also removed by the 50 shot layer-by-layer approach.

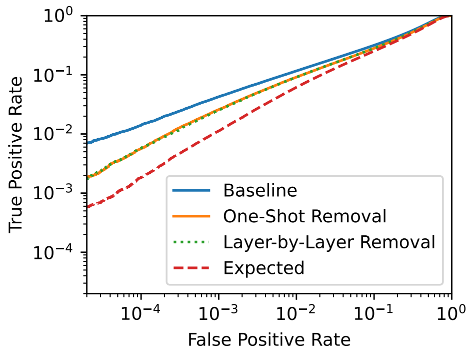

Figure 7: The Onion Effect is not a result of the outliers changing after each removal step. We repeatedly iterate 50 steps of identifying outliers and removing the top-100 outliers. This ends up also removing $5{,}000$ outliers, and performs identically to the baseline configuration where we remove all $5{,}000$ outliers in one shot.

B Visualization of Easy-to-Attack and Hard-to-Attack Examples

Figure 8 visualizes the easy to attack examples from CIFAR-10 training set, according to their privacy scores. The examples that are vulnerable to membership influence attack are generally outliers memorized by the models. The results are consistent with previous work that identify outliers and memorized examples in neural network learning [e.g. 12, 18].

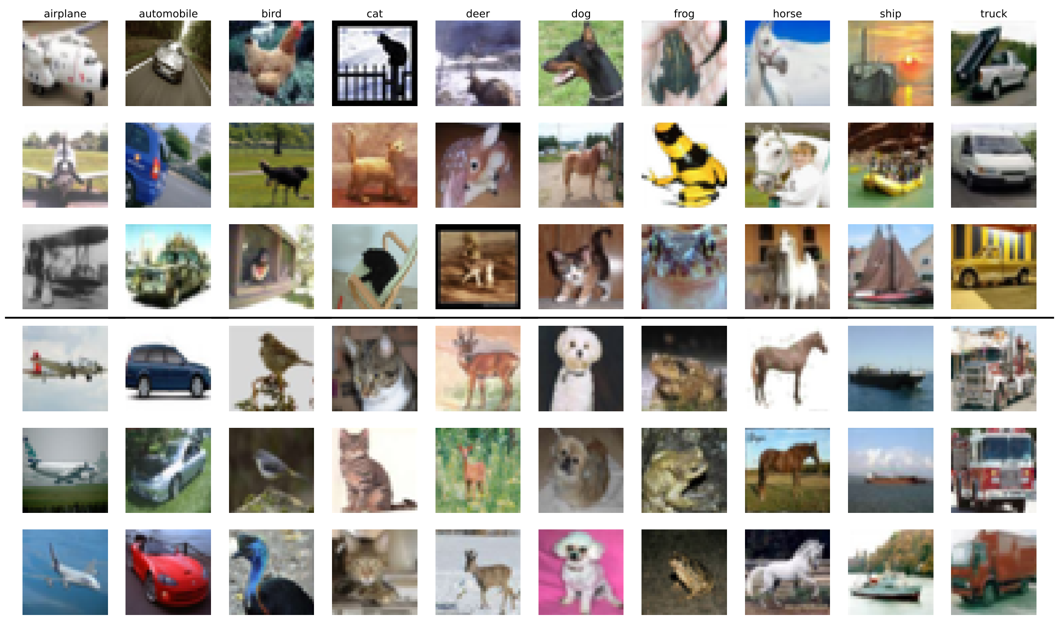

Figure 8: The easy to attack (top 3 rows) and difficult to attack (bottom 3 rows) CIFAR-10 examples from each of the 10 classes. Each column shows images from one class, with the top 3 rows randomly sampled from the top 100 most vulnerable examples, and the bottom 3 rows randomly sampled from the 100 least vulnerable examples.

C Details of CIFAR-10 Deduplication

We use the open source image deduplication library `imagededup` [17], available at `https://github.com/idealo/imagededup`, to deduplicate the CIFAR-10 training set. Specifically, we use the Convolutional Neural Network based detection algorithm with a threshold of 0.85, which ended up removing 5,275 duplicated images from the training set. We have manually inspected a random set of duplicated clusters identified by the software to verify the correctness. Figure 9 visualizes one of the largest duplicated clusters identified.

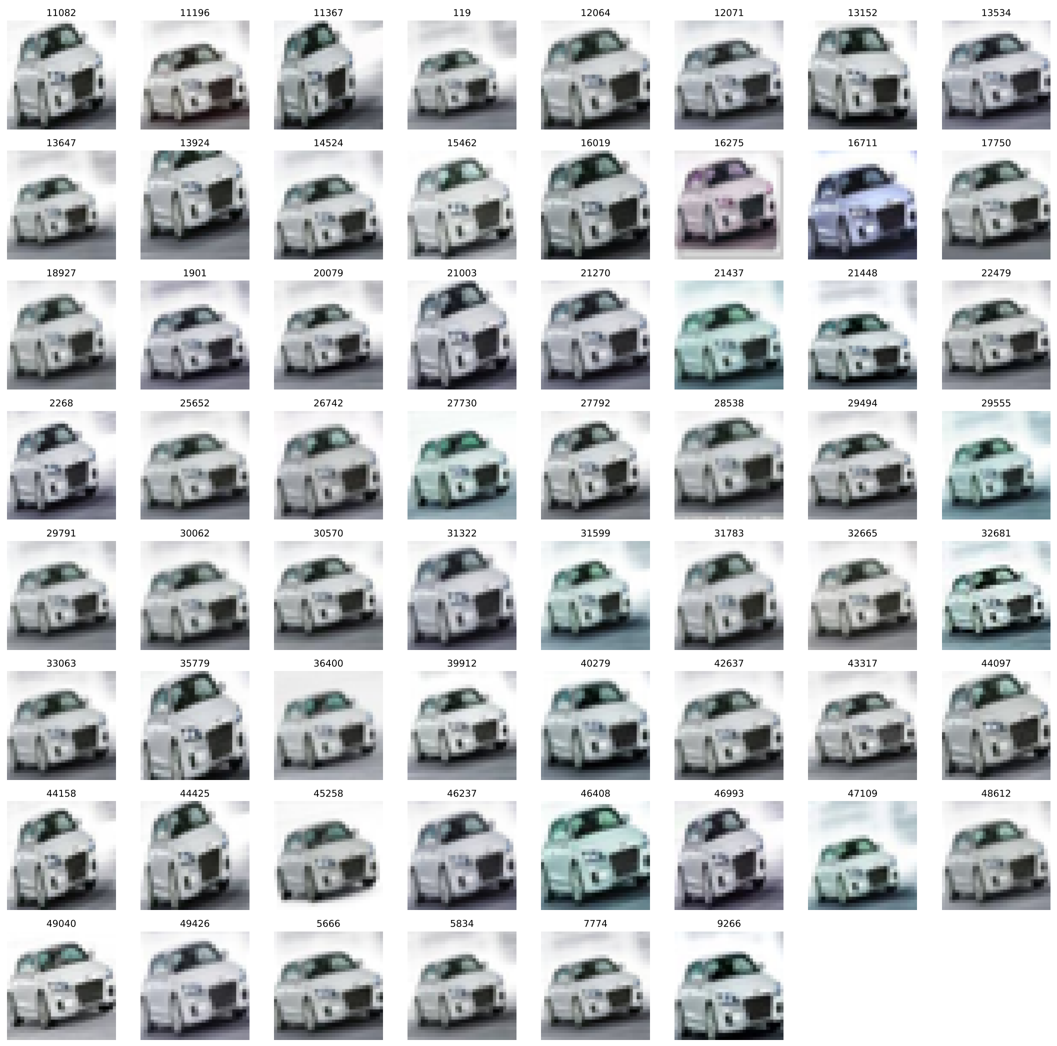

Figure 9: Duplicated images from the CIFAR-10 training set identified by the open source image deduplication library `imagededup` [17]. The number on top of each image is its index in the original CIFAR10 training set ordering.