membership classification accuracy through Equation 1. The Utility of the model was obtained with Equation 2. We used a **Black-box LTU Attacker** (number (3) in Figure 2). The results shown in Table 1 are averaged over 3 trials[3]. The first few lines (gray shaded) are **deterministic methods** (whose trainer yields to the same model regardless of random seeds and sample order). For these lines, consistent with Theorem 3, Privacy is zero, in all columns.[4] The algorithms use default scikit-learn hyper-parameter values. In the first two result columns, the Defender trainers are forced to be deterministic by seeding all random number generators. In the first column, the sample order is fixed to the order used by the Defender trainer, while in the second one it is not. Privacy in the first column is near zero, consistent with the theory. In the second column, this is also verified for methods independent of sample order. The third result column corresponds to varying the random seed, hence **algorithms including some level of randomness have an increased level of privacy**.

The results of Table 1 show that there is no difference between the column 2 and 3; suggesting that, just with the randomness associated to altering the order of the training samples, is enough to make the strategy fails. These results also expose one limitation of black-box attacks: example-based methods indicated in red (*e.g.,* SVC), which store examples in the model, obviously violate privacy. However, this is not detected by a black-box LTU Attacker , if they are properly regularized and/or involve some degree or randomness. White-box attackers solve this problem.

## White-box attacker

We implemented a white-box attacker based on gradient calculations (method (4) in Figure 2). We evaluated the effect of the proposed attack with QMNIST on two types of Defender models: A Deep Neural Networks (DNN) trained with supervised learning or with unsupervised domain adaptation (UDA) (**?**).[5]

For supervised learning, we used ResNet50 (**?**) as the backbone neural network, which is pre-trained on ImageNet (**?**). We then retrained all its layers on the Defender set of QMNIST. The results are reported in Table 2, line "Supervised". With the very large Defender dataset we are using for training (200000 examples), regardless of variations on regularization hyper-parameters, we could get ResNet50 to overfit. Consequently, both Utility and Privacy are good.

In an effort to still improve Privacy, we used Unsupervised Domain Adaptation (UDA). To that end, we use as source domains a synthetic dataset, called Large-Fake-MNIST (**?**), which are similar to MNIST. Large-Fake-MNIST has 50000 white-on-black images for each digit, which results in 500000 images in total. The target domain is the Defender set of QMNIST. The chosen UDA method is DSAN (**?**; **?**), which optimizes the neural network with the sum of a cross-entropy loss (classification loss) and a local MMD loss (transfer loss) (**?**). We tried 3 variants of attacks of this UDA model. The simplest is the most effective: attack the model as if it were trained with supervised learning. Unfortunately, UDA did not yield improved performance. We attribute that to the fact that the supervised model under attack performs well on this dataset and already has a very good level of privacy.

Table 1: **Utility and Privacy on QMNIST and CIFAR-10 of different scikit-learn models** with three levels of randomness: Original sample order + Fixed random seed (no randomness); Random sample order + Fixed random seed; Random sample order + Random seed. The Defender data and Reserved data have both 1600 examples. All numbers shown in the table have *at least* two significant digits (standard error lower than 0.004). For model implementations, we use scikit-learn (version 0.24.2) with default values. Results with Utility or Privacy $>$ 0.90 are highlighted and those meeting both criteria are underlined. Shaded in gray: fully deterministic models with Privacy$\equiv$ 0.

| **QMNIST**<br>Utility \| Privacy | Orig. order + Seeded | | Rand. order + Seeded | | Not Seeded | |
|---|---|---|---|---|---|---|
| Logistic lbfgs | **0.92** | 0.00 | **0.91** | 0.00 | **0.91** | 0.00 |
| Bayesian ridge | **0.92** | 0.00 | **0.92** | 0.00 | 0.89 | 0.00 |
| Naive Bayes | 0.70 | 0.00 | 0.70 | 0.00 | 0.70 | 0.00 |
| SVC | **0.91** | 0.00 | **0.91** | 0.00 | 0.88 | 0.00 |
| KNN* | 0.86 | 0.27 | 0.86 | 0.27 | 0.83 | 0.18 |
| LinearSVC | **0.92** | 0.00 | **0.92** | 0.69 | **0.91** | 0.63 |
| SGD SVC | **0.90** | 0.03 | **0.92** | **1.00** | 0.89 | **1.00** |
| MLP | **0.90** | 0.00 | **0.90** | **0.97** | 0.88 | **0.93** |
| Perceptron | **0.90** | 0.04 | **0.91** | **1.00** | **0.92** | **1.00** |
| Random Forest | 0.88 | 0.00 | 0.88 | **0.99** | 0.85 | **1.00** |

| **CIFAR-10**<br>Utility \| Privacy | Orig. order + Seeded | | Rand. order + Seeded | | Not Seeded | |
|---|---|---|---|---|---|---|
| Logistic lbfgs | **0.95** | 0.00 | **0.95** | 0.00 | **0.95** | 0.00 |
| Bayesian ridge | 0.91 | 0.00 | 0.90 | 0.00 | 0.90 | 0.00 |
| Naive Bayes | 0.89 | 0.00 | 0.89 | 0.01 | 0.89 | 0.00 |
| SVC | **0.95** | 0.00 | 0.94 | 0.00 | **0.95** | 0.00 |
| KNN* | 0.92 | 0.44 | 0.91 | 0.49 | 0.92 | 0.49 |
| LinearSVC | **0.95** | 0.00 | **0.95** | 0.26 | **0.95** | 0.22 |
| SGD SVC | 0.94 | 0.32 | 0.94 | **0.98** | 0.93 | **0.99** |
| MLP | **0.95** | 0.00 | 0.94 | **0.98** | **0.95** | **0.97** |
| Perceptron | 0.94 | 0.26 | 0.94 | **1.00** | 0.93 | **0.96** |
| Random Forest | 0.92 | 0.00 | 0.93 | **0.99** | 0.91 | 0.92 |

Table 2: **Utility and Privacy of DNN ResNet50 Defender models trained on QMNIST.**

| Defender model | Utility | Privacy |
|---|---|---|
| Supervised | $1.00 \pm 0.00$ | $0.97 \pm 0.03$ |
| Unsupervised Domain Adaptation | $0.99 \pm 0.00$ | $0.94 \pm 0.03$ |

[3] Code is available at `https://github.com/JiangnanH/ppml-workshop/blob/master/generate_table_v1.py`.

[4] Results may vary depending upon which scikit-learn output method is used (`predict_proba()`, `decision_function()`, `density_function()`), or `predict()`. To achieve zero Privacy, consistent with the theory, the method `predict()` should be avoided.

[5] Code is available at `https://github.com/JiangnanH/ppml-workshop#white-box-attacker`.

## Discussion and further work

Although an LTU Attacker is all knowledgeable, it must make efficient use of available information to be powerful. We proposed a taxonomy based on information available or used (Figure 2). The most powerful Attackers use both the trained Defender model $\mathcal{M}_D$ and its trainer $\mathcal{T}_D$.

When the Defender trainer $\mathcal{T}_D$ is a black box, like in our first set of experiments on scikit-learn algorithms, we see clear limitations of the LTU Attacker which include the fact that it is not possible to diagnose whether the algorithm is example-based.

Unfortunately, white-box attacks cannot be conducted in a generic way, but must be tailored to the trainer (*e.g.,* gradient descent algorithms for MLP). In contrast, black-box methods can attack $\mathcal{T}_D$ (and $\mathcal{M}_D$) regardless of mechanism. Still, we get necessary conditions for privacy protection by analyzing black-box methods. Both theoretical and empirical results using black-box attackers (on a broad range of algorithms of the scikit-learn library on the QMNIST and CIFAR-10 data), indicate that Defender algorithms are vulnerable to a LTU Attacker if it overfits the training Defender data or if it is deterministic. Additionally, the degree of stochasticity of the algorithm must be sufficient to obtain a desired level of privacy.

We explored white-box attacks neural networks trained with gradient descent. In our experiments on the large QMNIST dataset (200,000 training examples), Deep CNNs such as ResNet seem to exhibit both good Utility and Privacy in their "native form", according to our white-box attacker. We were pleasantly surprised of our white box attack results, but, in light of the fact that other authors found similar networks vulnerable to attack (**?**), we conducted the following sanity check. We performed the same supervised learning experiment by modifying $20\%$ of the class labels (to another class label chosen randomly), in both the Defender set and Reserved set. Then we incited the neural network to overfit the Defender set. Although the training accuracy (on Defender data) was still nearly perfect, we obtained a loss of test accuracy (on Reserved data): $78\%$. According Theorem 2, this should result in a loss of privacy. This allowed us to verify that our white-box attacker correctly detected a loss of privacy. Indeed, we obtained a privacy of 0.55.

We are in the process of conducting comparison experiments between our white-box attacker and that of (**?**). However, their method does not easily lend itself to be used with the LTU framework, because it requires training a neural network for each LTU round (*i.e.,* on each $\mathcal{D}_A = \mathcal{D}_D - \{\texttt{membership}(d)\} \cup \mathcal{D}_R - \{\texttt{membership}(r)\}$). We are considering doing only one data split to evaluate privacy, with $\mathcal{D}_A = 50\%\ \mathcal{D}_D \ \cup\ 50\%\ \mathcal{D}_R$ and using the rest of the data for privacy evaluation. However, we can still use the pairwise testing of the LTU methodology, *i.e.,* the evaluator queries the attacker with pairs of samples, one from the Defender data and the other from the Reserved data. In Appendix C, we show on an example that this results in an increased accuracy of the attacker.

In Appendix C, we use the same example to illustrate how we can visualize the privacy protection of individuals. Further work includes comparing this approach with (**?**).

Further work also includes testing LTU Attacker on a wider variety of datasets and algorithms, varying the number of training examples, training new white-box attack variants to possibly increase the power of the attacker, and testing various means of improving the robustness of algorithms against attacks by LTU Attacker . We are also in the process of designing a competition of membership inference attacks.

## Conclusion

In summary, we presented an apparatus for evaluating the robustness of machine learning models (Defenders) against membership inference attack, involving an "all knowledgeable" LTU Attacker . This attacker has access to the trained model of the Defender, its learning algorithm (trainer), all the Defender data used for training, *minus the label of one sample*, and all the *similarly distributed* non-training Reserved data (used for evaluation), *minus the label of one sample*. The Evaluator repeats this Leave-Two-Unlabeled (LTU) procedure for many sample pairs, to compute the efficacy of the Attacker, whose charter is to predict the membership of the unlabeled samples (training or non-training data). We call such LTU Attacker the LTU-attacker for short. The LTU framework helped us analyse privacy vulnerabilities both theoretically and experimentally.

The main conclusions of this paper are that a number of conditions are necessary for a Defender to protect privacy:

1. Avoid storing examples (a weakness of example-based method, such as Nearest Neighbors). 2. Ensure that $p_R = p_D$ for all $f$, following Theorem 1 ($p_R$ is the probability that discriminant function $f$ "favors" Reserved data while $p_D$ is the probability with which it favors the Defender data). 3. Ensure that $e_R = e_D$, following Theorem 2 ($e_R$ is the expected value of the loss on Reserved data and $e_D$ on Defender data). 4. Include some randomness in the Defender trainer algorithm, after Theorem 3.

### Acknowledgements

We are grateful to our colleagues Kristin Bennett and Jennifer He for stimulating discussion. This work is funded in part by the ANR (Agence Nationale de la Recherche, National Agency for Research) under AI chair of excellence HUMANIA, grant number ANR-19-CHIA-0022.

Eaque vero necessitatibus dolorum eum maxime recusandae ipsa cum autem vitae soluta, praesentium iusto minima id laborum pariatur totam qui nemo molestias, tenetur illum amet doloribus asperiores repudiandae, voluptas nemo quae facilis corrupti sit quisquam quia repudiandae illum.Esse aliquid aliquam nostrum ea quidem sint eligendi, labore accusamus aspernatur odit temporibus explicabo voluptatibus deleniti voluptas consequatur quaerat quo, recusandae suscipit fugit perspiciatis enim magni sint eaque molestiae dolore vel.Quasi aut officia iste exercitationem accusamus repellendus earum nam, consectetur ratione accusamus ullam quos molestias repellat corporis unde dignissimos?Placeat cupiditate vitae voluptates, voluptatibus ratione fugiat vero cumque, voluptate deserunt sunt accusantium non aliquam nam numquam eaque delectus, deleniti culpa laborum eligendi harum exercitationem?