ON THE VULNERABILITY OF RECURRENT NEURAL NETWORKS TO MEMBERSHIP INFERENCE ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the privacy implications of deploying recurrent neural networks in machine learning. We consider membership inference attacks (MIAs) in which an attacker aims to infer whether a given data record has been used in the training of a learning agent. Using existing MIAs that target feed-forward neural networks, we empirically demonstrate that the attack accuracy wanes for data records used earlier in the training history. Alternatively, recurrent networks are specifically designed to better remember their past experience; hence, they are likely to be more vulnerable to MIAs than their feed-forward counterparts. We develop a pair of MIA layouts for two primary applications of recurrent networks, namely, deep reinforcement learning and sequence-to-sequence tasks. We use the first attack to provide empirical evidence that recurrent networks are indeed more vulnerable to MIAs than feed-forward networks with the same performance level. We use the second attack to showcase the differences between the effects of overtraining recurrent and feed-forward networks on the accuracy of their respective MIAs. Finally, we deploy a differential privacy mechanism to resolve the privacy vulnerability that the MIAs exploit. For both attack layouts, the privacy mechanism degrades the attack accuracy from above 80% to 50%, which is equal to guessing the data membership uniformly at random, while trading off less than 10% utility.

1 INTRODUCTION

In many machine learning applications, such as those in healthcare and financial services, the training datasets contain personally identifiable information that could be used to breach the privacy of the individuals who provide them. Often protecting the privacy of the training datasets for such systems is required by law, e.g., the US Health Insurance Probability and Accountability Act 1996, which applies to processing healthcare data. Additionally, privacy protection may have strategic values irrespective of any enacted legislation. For example, market analysis providers that are subject to data scraping, such as LinkedIn (Dai et al., 2015), must prevent their machine learning algorithms from oversharing the information that empowers their analysis. As a result, it is essential to study the privacy implications of machine learning models before deploying them.

Recurrent neural networks (RNNs) are beneficial in a variety of machine learning tasks such as speech and handwriting recognition (Sak et al., 2014; Li & Wu, 2015; Graves et al., 2008), and deep reinforcement learning (Li et al., 2015; Liu et al., 2017). However, little is known about the potential privacy risks associated with these networks, in contrast to the case for feed-forward neural networks. The potential privacy risks may be directly exploited by cyber attackers to compromise data confidentiality or may be used for spear phishing to advance other destructive cyber attacks (Halevi et al., 2015). We therefore follow the three fundamental steps to any cyber attack analysis laid out by Chesney (2020). These steps are: vulnerability identification, exploit implementation, and developing a patch to remedy the vulnerability. In the sequel, we will state our respective contributions under each of these steps to study the privacy implications of RNNs.

For vulnerability identification, we consider membership inference attacks (MIAs) in which the attackers aim to infer whether a given data record has been used in the training of a model. Using an existing MIA on image classification models by Shokri et al. (2017), we find that images that have been used earlier in the history of training are more vulnerable to the MIA than those that have been used closer to the attack's execution. It is plausible that the diminishing vulnerability is attributed

to feed-forward networks' lack of memory to remember their past observations. On the other hand, state-of-the-art RNN architectures, such as long short-term memory (LSTM) (Dupond, 2019), use memory cells that allow them to remember training data far back in the training history. Hence, we conjecture that these models are more vulnerable to MIAs than their feed-forward counterparts.

In the second contribution, we confirm our conjecture by designing MIAs that effectively exploit the speculated vulnerabilities. We first design an MIA layout for deep reinforcement learning algorithms in which both feed-forward and recurrent networks are commonly used. The MIA infers whether its victim model has visited a specific region of a map, compromising the model's location privacy. We use the MIA to attack two agents with similar performance levels: one that uses LSTM units and one that does not. Measuring the attack accuracy of the MIA on both of the agents, we find that the attack is roughly 98% successful for the RNN agent, whereas the success rate for the feed-forward agent is 91%. Furthermore, we examine the temporal order of the regions that the MIA correctly labels as a member or not a member. We find that the lower attack accuracy on the feed-forward agent corresponds to the regions visited earlier in the training history.

We also design an MIA layout for machine translation tasks, which are examples of sequence-to-sequence tasks. RNNs are widely used in machine translation because they allow for processing sentences with arbitrary lengths (Dupond, 2019). We use the MIA to compare the effects of overtraining recurrent and feed-forward networks on their vulnerability to their respective MIAs. RNNs often do not lose their generalization power due to overtraining, as opposed to feed-forward networks (Song & Shmatikov, 2019). The experimental results indicate that overtraining RNNs may have a marginal effect on the MIA's accuracy, as opposed to the case for the feed-forward model. Therefore, existing defense methods for feed-forward networks that prevent overfitting, such as those in (Shokri et al., 2017; Salem et al., 2018), may not be suitable for RNNs.

In the final contribution, we patch the identified vulnerability by deploying the Dirichlet mechanism, introduced by Gohari et al. (2021). In both sequence-to-sequence tasks and deep reinforcement learning tasks, the model outputs cast a probability distribution over the candidate words and the environment actions, respectively. The Dirichlet mechanism enforces differential privacy by obfuscating these probability values. Differential privacy mechanisms typically face a trade-off between privacy and utility (Dwork & Roth, 2014). We therefore evaluate the performance of the Dirichlet mechanism with respect to two criteria: effectiveness against the MIA and utility loss. Against both of the attack layouts that we develop, the Dirichlet mechanism can reduce the attack accuracy from above 80% to 50%, which is equivalent to making uniformly at random inferences. Moreover, the utility loss in both tasks was found to be no more than 10% of the original performance levels.

2 Related Works

We first review some of the the existing MIA methods in the literature. The idea of MIAs was first introduced by Homer et al. (2008) to determine whether an individual's record is within complex genomic DNA mixtures. Shokri et al. (2017) later developed the first MIA that targets the training dataset of a machine learning model. The subsequent works either enhanced the performance of the existing MIAs (Salem et al., 2018), uncovered unknown vulnerabilities, such as the vulnerability of training datasets with correlated data to MIAs in (Gomrokchi et al., 2021), or expanded the application domain of MIAs to more sophisticated machine learning tasks, examples of which are federated learning (Melis et al., 2019), reinforcement learning (Pan et al., 2019), and natural language processing (Song & Shmatikov, 2019). We refer the reader to (Jia et al., 2019; Truex et al., 2019) and the references therein for a comprehensive literature study on MIAs.

The work in (Hisamoto et al., 2020) develops an MIA for sequence-to-sequence tasks and is related to the current paper; however, the sequence-to-sequence models that they study do not deploy RNNs, whereas we focus on RNNs. In another work, Song & Shmatikov (2019) propose an MIA that targets sequence-to-sequence models that deploy RNNs. However, the proposed method truncates the sequences that it processes, whereas we use an RNN in the attack model which allows for processing arbitrary-length sequences. Furthermore, we empirically demonstrate that we outperform the existing MIA in the benchmarks.

We next review the related MIA studies in the reinforcement learning domain. Pan et al. (2019) developed the first MIA that targets deep reinforcement learning agents. However, the MIA infers

the transition probabilities of the environment, while we target the agent itself. Concurrently with and independently of this paper, Gomrokchi et al. (2021) develop an MIA for deep reinforcement learning agents which, similar to our work, infers the membership of roll-out trajectories. The above work develops the MIA to study agents with correlated training datasets, whereas we develop the MIA to study the vulnerability of agents that deploy RNNs. Moreover, we use RNNs to process trajectories with arbitrary lengths and the MIA in (Gomrokchi et al., 2021) truncates the trajectories.

We now review the works that study defending machine learning models against MIAs. The defense methods either use regularization or noise-additive methods, or a combination thereof. Examples of regularization defense methods are ℓ_2-regularization (Shokri et al., 2017), dropout and model stacking (Salem et al., 2018), and max-min game (Nasr et al., 2018). The noise-additive methods inject an external noise to different attributes of the network to either directly mislead MIAs through adversarial examples (Hunt et al., 2018; Jia et al., 2019), or to enforce differential privacy (Rahman et al., 2018; Abadi et al., 2016; Ji et al., 2014).

Differential privacy defense methods are often the most effective against MIAs; however, in doing so, they typically degrade the utility of the model under protection (Carlini et al., 2019; Jia et al., 2019). The Dirichlet mechanism is a non-invasive privacy mechanism in the sense that it does not require access to the internal attributes of the protected network and only perturbs the output post training. Applications of the Dirichlet mechanism have been studied in load ensemble control (Hassan et al., 2021), constant function market makers (Chitra, 2021), and policy synthesis in Markov decision processes (Gohari et al., 2020). To the best of our knowledge, we are the first to empirically study the Dirichlet mechanism's effectiveness in defending machine learning models against MIAs.

3 Attack Method

In this section, we develop two MIA layouts to attack sequence-to-sequence and deep reinforcement learning models, respectively. Throughout the rest of this paper, we refer to the model under attack as the victim model. An MIA is a binary classifier that, by observing the victim model's output to a given data record, labels it as either 'in' or 'out'. The former label refers to when the data record has been a member of the victim's training dataset, and the latter refers to the opposite case. The main challenge in designing an MIA is populating the dataset by which we train the binary classifier.

Following the design method in (Shokri et al., 2017), we train *shadow* models to populate the binary classifier's training dataset. The outputs of a shadow model must approximate the victim model's outputs. However, the shadow model may not have access to the victim model's training dataset. As an example, in designing a shadow model for image classifiers, Shokri et al. (2017) train the shadow model on a separate dataset that has different images than the victim model's training dataset under the same image categories.

As the attacker trains the shadow models itself, it knows which data records have been used in the training of the shadow model. The attacker subsequently assigns the label 'in' to the data records that it used during the training of the shadow models and gathers a collection of data records that it did not use in the training of the shadow model and labels them as 'out'. The attacker then populates the binary classifier's training dataset with the above 'in' and 'out' data records. Upon training the binary classifier with the resulting training dataset, the classifier can be used to infer the membership of new data records corresponding to the victim model.

3.1 Attacking Sequence-to-Sequence Models

In sequence-to-sequence tasks, the model must map a sequential input to a sequential output from possibly different domains. We consider machine translation tasks wherein the agent must translate a word sequence from a source language to a target language. Machine translation tasks are instances of supervised learning tasks, for which there exist a handful of powerful MIA methods. However, a machine translation model may generate sequences with arbitrary lengths and the attackers must be able to process such sequences.

We first train a shadow model to approximate the outputs of the victim model. We subsequently construct the binary classifier's dataset as previously discussed to train the classifier. Compared with the related MIA in (Song & Shmatikov, 2019), our design has two notable differences. First we

use an RNN architecture for the classifier which allows for processing arbitrary-length sequences. Furthermore, as opposed to using the rank of the probabilities according to which the victim model generates its sentences, we use the values to infer the membership of a given sentence.

3.2 Attacking Deep Reinforcement Learning Agents

Reinforcement learning agents interact with an environment that is typically modeled as a Markov decision process (Sutton & Barto, 2018). A Markov decision process consists of a state and an action space, transition probabilities, and a reward function. The goal in a reinforcement learning task is to learn a reward-maximizing policy, which casts a probability distribution over the action space at every state, without knowing the transition probabilities a priori.

Deep reinforcement learning agents deploy neural networks to approximate the value of every state, with respect to their expected cumulative reward, and to generate the policy at every given state. The agents update their neural networks using the trajectories, i.e., a temporal sequences of state, action, and immediate rewards, that they collect from interacting with the environment.

Assume that the state space of the environment is a union of disjoint regions. Our goal in designing an MIA for reinforcement learning agents is to obtain a trajectory from the victim model, at a given region of the environment and subsequent to its training, and infer whether the victim model has visited the region while being trained. In robotics or autonomous driving applications, the states often represent the location of the agent and MIAs with such inference capabilities compromise their victim's location privacy.

The existing MIA methods that target supervised learning models are ill-suited to attacking a reinforcement learning agent. The existing methods typically use the labels of the shadow model's training dataset to populate the binary classifier's training dataset, e.g., (Shokri et al., 2017). However, in reinforcement learning, there does not exist a labeled training dataset. Moreover, the action probabilities do not necessarily represent the model's confidence in taking each of the actions. In contrast, in supervised learning tasks such as image classification, the values of the output layer often represent the model's confidence.

As an example, consider that an image classifier with two categories assigns probabilities 0.6 and 0.4 to the respective categories. The MIA may reconcile the prediction probabilities with the image's label and correctly infer that the image as 'out' due to the agent's low confidence. However, in reinforcement learning, assigning the same probabilities to the actions may be an optimal stochastic policy and correspond to an 'in' data record.

To remedy the above challenges, we train a separate model for every region that the MIA targets to find the optimal policy, and therefore, call the model as a label model. We then train a shadow model from which we populate the training dataset of the binary classifier. While populating the dataset, we concatenate the sequence of policies that we obtain from the shadow model with the corresponding label model's policy sequence. Finally, we train the binary classifier to complete the attack. Similar to the sequence-to-sequence attack, we use an RNN for the binary classifier because the policy sequences may have arbitrary lengths.

4 Defense Method

In this section, we review the definition of differential privacy and the Dirichlet mechanism. Differential privacy is a quantitative definition of data privacy and is a characteristic of an algorithm (Dwork & Roth, 2014). An algorithm that satisfies differential privacy makes it difficult for observers with arbitrary computation powers to attribute the observed outputs to their respective inputs. Often when an algorithm does not satisfy differential privacy by itself, an external privacy mechanism modifies the algorithm to satisfy differential privacy.

Differential privacy is suitable for situations in which the algorithm's outputs represent aggregate statistics of a dataset whose individual entries contain sensitive information. For example, the US Census Bureau adopts differential privacy to protect its publications (Abowd, 2018). In classical *global* differential privacy (Dwork & Roth, 2014), the aggregator must generate almost-identical outputs to input datasets that differ in one entry. Such datasets are called *adjacent*. In *local* differential privacy, the individuals deploy a privacy mechanism prior to sharing their data with the

aggregator (Duchi et al., 2013). Therefore, the participating individuals need not trust the aggregator and are in charge of their own data privacy. The adjacency relationship in local differential privacy often defines two data records to be adjacent if their distance with respect to a fixed measure is upper bounded by a constant. We now formally define both global and local differential privacy.

Definition 1 (**GLOBAL DIFFERENTIAL PRIVACY**) *Fix an algorithm $\mathcal{A}$ with domain $\mathcal{D}$ and range $\mathcal{R}$. Define two datasets D and D', both in $\mathcal{D}$, globally adjacent if the number of entries in which the two datasets hold different values is at most one. Fix a probability space $(\Omega, \mathcal{F}, \mu)$ and let $\mathcal{M}$ be a σ-algebra such that the space $(\mathcal{R}, \mathcal{M})$ is measurable. For given $\epsilon \geq 0$ and $\delta \in [0, 1)$, $\mathcal{A}$ is (ϵ, δ)-differenetially private if, for all $S \subseteq \mathcal{R}$ and all globally adjacent D and D',*

$$\Pr[\mathcal{A}(D) \in S] \leq \exp(\epsilon)\Pr[\mathcal{A}(D') \in S] + \delta. \tag{1}$$

Definition 2 (**LOCAL DIFFERENTIAL PRIVACY**) *Fix an algorithm $\mathcal{A}$ with domain $\mathcal{X}$ and range $\mathcal{Y}$, and a local adjacency relationship $d : \mathcal{X} \times \mathcal{X} \mapsto \{0, 1\}$. Let $(\Omega, \mathcal{F}, \mu)$ be a probability space and $\mathcal{M}$ be a σ-algebra such that the space $(\mathcal{Y}, \mathcal{M})$ is measurable. For given $\epsilon \geq 0$ and $\delta \in [0, 1)$, $\mathcal{A}$ is (ϵ, δ)-locally differnetially private if, for all $S \subseteq \mathcal{Y}$ and all x and x' such that $d(x, x') = 1$,*

$$\Pr[\mathcal{A}(x) \in S] \leq \exp(\epsilon)\Pr[\mathcal{A}(x') \in S] + \delta. \tag{2}$$

Due to the Composition Theorem (Dwork & Roth, 2014), subsequent queries from differential privacy mechanisms weaken the overall privacy protections. In particular, the composition of n (ϵ, δ)-differential privacy mechanisms results in $(n\epsilon, n\delta)$-differential privacy. The privacy parameters of a sequence of differential privacy mechanisms are often referred to as the privacy budget. Local differential privacy mechanisms often consume a significantly higher privacy budget than their global differential privacy counterparts. However, they may still provide a strong privacy shield while consuming high privacy budgets (Bhowmick et al., 2018).

Having stated the definition of differential privacy, we now review the Dirichlet mechanism. The Dirichlet mechanism is a well-suited privacy mechanism for systems whose outputs cast a probability measure over a finite set. The Dirichlet mechanism enforces local differential privacy without altering the system's internal algorithm; therefore, it is a post-hoc privacy mechanism. The Dirichlet mechanism is parameterized by a scalar $k > 0$ and takes as input a vector within the interior of the unit simplex, i.e.,

$$\Delta_n := \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1, \forall i \in \{1, \ldots, n\} : x_i \geq 0 \right\}. \tag{3}$$

We denote the interior of the unit simplex by Δ_n°. For all $p \in \Delta_n^\circ$, we denote the Dirichlet mechanism itself by $\mathrm{Dir}_k(p)$. The Dirichlet mechanism maintains the structure of its input because it generates its outputs according to the Dirichlet distribution whose support is the unit simplex itself. Specifically, for all $p \in \Delta_n^\circ$,

$$\Pr\left(\mathrm{Dir}_k(p) = x\right) = \frac{1}{\mathrm{B}(kp)} \prod_{i=1}^{n} x_i^{kp_i - 1} \mathbb{I}_{\{x \in \Delta_n\}}, \tag{4}$$

where $\mathbb{I}_{\{A\}}$ is an indicator function that equals one if the predicate A is true, and zero otherwise, and $\mathrm{B}(\cdot)$ is the multivariate beta function. We now state a lemma establishing that, for all values of $k > 0$, the Dirichlet mechanism satisfies local differential privacy with bounded ϵ and $\delta < 1$.

Lemma 1 (Gohari et al. (2021)) *Fix a Dirichlet mechanism with parameter k and let $\Delta_n(\eta) := \Delta_n \cap \{x \in \mathbb{R}^n \mid \forall i \in \{1, \ldots, n\} : x_i \in [\eta, 1 - \eta]\}$, for all applicable η. Define the local adjacency relationship $d(\cdot, \cdot)$ as*

$$d(p, p') = \mathbb{I}_{\{\exists (i,j) \text{ s.t. } \|p - p'\|_1 \leq b \text{ and } p_{-(i,j)} = p'_{-(i,j)}\}}, \tag{5}$$

where b is a constant, $\|\cdot\|_1$ denotes the 1-norm of a vector, and $p_{-(i,j)}$ is the vector p excluding its i and j^{th} components. Fix $\delta \in [0, 1)$ and η, and let $p^* = [\eta, \ldots, \eta, 1 - (n-1)\eta]^\top$. Let k and τ satisfy

$$\delta \leq 1 - \int_{x \in \Delta_n(\tau)} \frac{1}{\mathrm{B}\left(kp^*\right)} \prod_{i=1}^{n-1} x_i^{kp_i^* - 1} dx. \tag{6}$$

Then, the Dirichlet mechanism with parameter k satisfies (ϵ, δ)-local differential privacy, where

$$\epsilon \leq 2k(1 - \eta) - 3 + \frac{kb}{2} \left| \log\left(1 - (n-1)\tau\right) - \log(\tau) \right|. \tag{7}$$

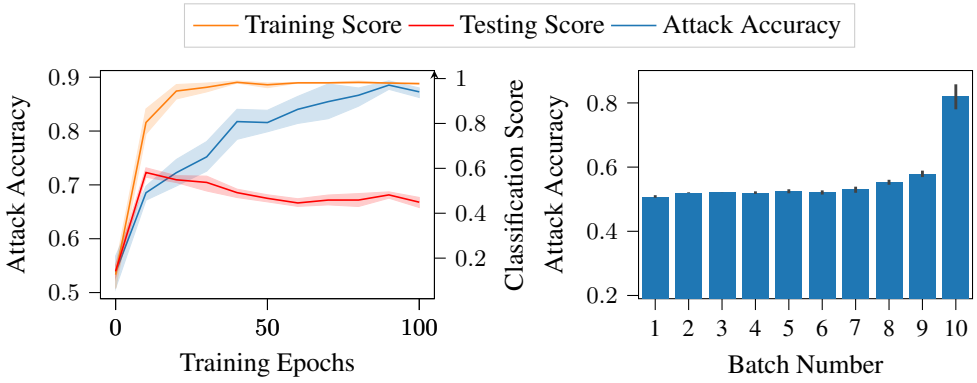

Figure 1: Attacking an image classification model using the CIFAR10 dataset. Left: the victim model's performance and the corresponding attack accuracy as functions of the training time. Right: the attack accuracy as a function of the order of the batches in the history of training.

5 Experiments

In this section, we perform three sets of experiments using the attack and defense methods that we laid out in the previous sections. In the first set of experiments, we compare the vulnerability of recurrent and feed-forward neural networks to MIAs. We then perform experiments to investigate the effects of overtraining RNNs on their vulnerability to MIAs. In the last set of experiments, we deploy the Dirichlet mechanism to defend against the MIAs that we develop.

5.1 Vulnerability

In the first experiment, we focus on the accuracy of MIAs across the training history of their victim model. To this end, we consider an existing MIA designed by Shokri et al. (2017) for image classification models. We use the CIFAR10 dataset (Krizhevsky et al., 2009) to train the model. However, instead of using the entire dataset to train the model at once, we divide the dataset into 10 batches and sequentially train the model on each of these batches. Once the model is fully trained, we separately apply the MIA to each of the batches and report the percentage of the MIA's correct inferences as the attack accuracy. The results in Figure 1 indicate that the attack accuracy monotonically decreases from roughly 80% to 50% as we move backwards in the history of training.

The above experiment suggests that, upon updating feed-forward neural networks with a new batch of data, the new batches wipe off the traces of their predecessors. On the other hand, RNNs are especially designed to remember their past experiences for extended periods of time. Therefore, we conjecture that RNNs are more vulnerable to MIAs than feed-forward neural networks.

In the next experiment, we confirm the above conjecture. We compare the vulnerability of recurrent and feed-forward networks to MIAs using the attack method that we designed for reinforcement learning tasks in Section 3.2. In these tasks, both recurrent and feed-forward architectures are commonly used. RNNs often outperform their feed-forward counterparts; however, training them is computationally more expensive. In the experiments, we fine-tune the agents' hyperparameters such that that their converging performance levels are similar in order to perform a fair comparison between the agents.

We use the MiniGrid toolkit (Chevalier-Boisvert et al., 2018) as the underlying testbed and use the RL-Starter-Files library (Willems, 2018) to train the deep reinforcement learning agents. From the MiniGrid environments, we choose the Multi-Room environment, wherein the agent must learn how to navigate its way through a series of connected rooms to reach a target destination. Upon reaching either the destination or a fixed number of time-steps, the environment resets to a new map; however, the task remains the same. Assuming that each new map represents a specific region of a larger map, we use the MIA to infer whether the agent has visited a given region in its training.

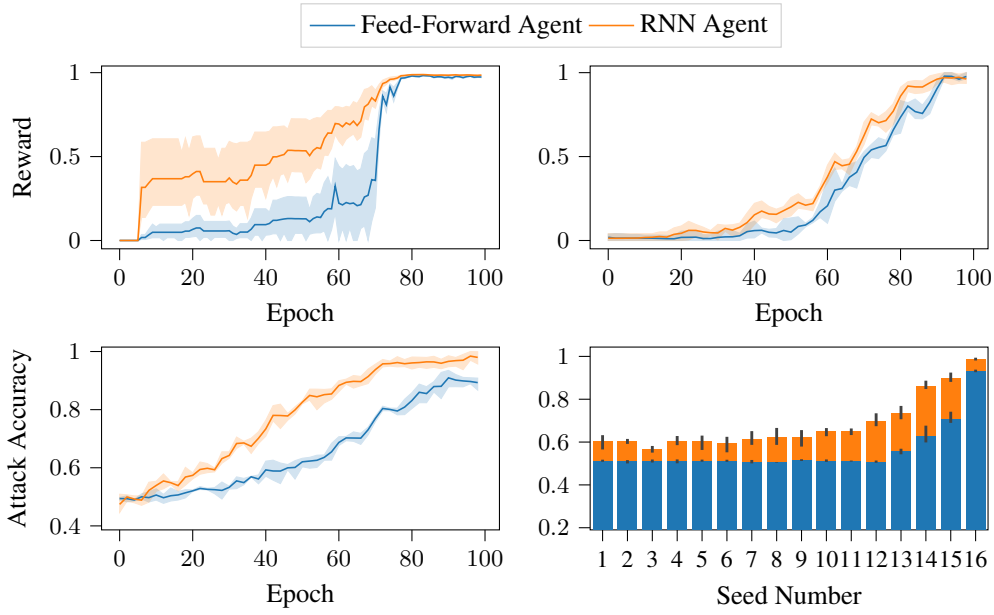

Figure 2: RNN vs. feed-forward neural networks in deep reinforcement learning. Top left: training performance vs. training time. Top right: validation performance vs. training time. Bottom left: attack accuracy vs. training time. Bottom right: attack accuracy as a function of the environment seed when the agents are sequentially trained form seed 1 to 16.

We use the PPO algorithm (Schulman et al., 2017) to train two agents: the first agent uses a feed-forward and the second agent uses a recurrent network architecture. In particular, the first agent uses a multi-layer perceptron (MLP) network while the second agent uses the same MLP with additional LSTM units. We use 16 environment seeds to train both of the agents and another 16 seeds to validate the agents' respective performance levels. In the top row of Figure 2, we plot the respective performance level of the two agents as a function of their training time.

In the next step, we use the MIA layout in Section 3.2 and perform two experiments. In the first experiment, we stop the training of the agents at a range of stopping times and evaluate the MIA's attack accuracy as a function of the training time. The results in Figure 2 show that across all training times, the attack accuracy on the RNN agent is higher than the feed-forward agent.

In the next experiment, we sequentially train the agents from seed 1 to 16 and examine the MIA's accuracy with respect to each of the seeds. The results in Figure 2 indicate that, similar to the case for the attack on the image classifier, the MIA's inference accuracy is not uniformly distributed across the training history of the agents. In particular, for both agents, the MIA's accuracy progressively decreases as we move backwards in the training history; however, the attack accuracy on the RNN agent is always above the corresponding attack accuracy on the feed-forward agent.

We conclude the first set of experiments with the empirical evidence that RNNs may be more vulnerable to MIAs than their feed-forward counterparts, and the excessive vulnerability corresponds to the RNNs' ability to memorize their past experience data.

5.2 Overtraining

In this section, we design a series of experiments to study how overtraining RNNs affects their vulnerability to MIAs. We change the task from deep reinforcement learning to machine translation to cover another mainstream application of RNNs.

We use the algorithm developed by Luong et al. (2015) and the Multi30K (Elliott et al., 2016) and SATED (Michel & Neubig, 2018) datasets to train models. We use the bilingual evaluation understudy (BLEU) score (Papineni et al., 2002) to evaluate the performance of the machine translation models that we train. A BLEU score takes values between 0 and 1, and is an indication of the

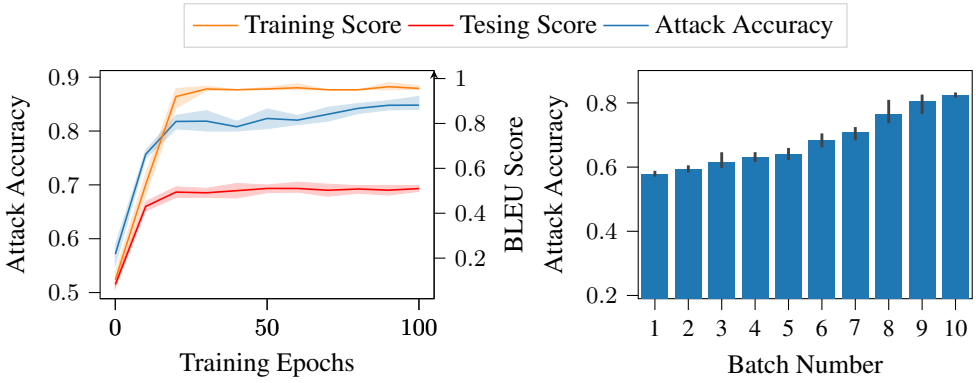

Figure 3: Attacking a machine translation model using the Multi30K dataset. Left: the victim model's performance and the corresponding attack accuracy as functions of training time when using the entirety of the dataset. Right: the attack accuracy as a function of the order of the batches in the history of training. In comparison with the image classification agent in Figure 1, the early batches of the machine translation model are more vulnerable to the MIA than the early batches of the image classification model.

similarity of the agent's output sentence to that of a human translator. The perfect value for BLEU score is 1; however, the perfect score may not be achievable, even by human translators, because the similarity is evaluated with respect to a fixed dataset of reference texts.

The existing MIAs on feed-forward neural networks are sensitive to their target model's generalization power. In particular, as feed-forward models overfit their training data, they become more vulnerable to MIAs Shokri et al. (2017). Overtraining sequence-to-sequence models that use RNNs may also render them more vulnerable to MIAs; however, the validation performance of these models may remain constant (Song & Shmatikov, 2019).

In the first experiment, we train a machine translation agent using the Multi30K dataset. We then evaluate the accuracy of the MIA that we develop according to Section 3.1 as a function of the training time of the victim model.

The results in Figure 3 uncover an important difference between MIAs on RNNs and feed-forward networks. In Figure 1, which corresponds to the image classification task, the agent has the highest generalization power when trained for 10 epochs, for which the corresponding MIA accuracy is roughly 70%. Once we overtrain the agent past 10 epochs, the accuracy increases to almost 90%. As a result, methods that prevent overfitting, such as ℓ_2-regularization and dropout, are effective against the MIA. On the other hand, in Figure 3, the agent's validation score reaches its highest value at the 20^{th} epoch and remains at that value thereafter. Once we overtrain the model, the attack accuracy increases by only a small margin. As a result, regularization methods may have marginal effects on RNNs.

In the supplementary materials, we present additional results on the SATED dataset. We also benchmark our MIA against the existing method by Song & Shmatikov (2019) and show that we outperform the existing MIA.

5.3 Defense

So far, we have studied the vulnerability of RNNs in deep reinforcement learning and sequence-to-sequence tasks to the MIAs that we developed in Section 3. We now study how to defend these models against their respective MIAs using the Dirichlet mechanism.

In the deep reinforcement learning task, the MIA infers the membership of a given region by observing the action probabilities that the victim model assigns to the states within a given trajectory. In the sequence-to-sequence task, the MIA infers the membership of a given sequence by observing the probabilities that the victim model assigns to each of the tokens within its output sequence. In both cases, we use the Dirichlet mechanism to enforce differential privacy and obfuscate the probabilities.

Table 1: Deploying the Dirichlet mechanism to protect deep reinforcement learning agents

Configuration	Attack Accuracy ($\pm$ std)	Total Reward ($\pm$ std)
No Protection	99.19 ($\pm$0.35) %	0.9095 ($\pm$0.0429)
$k = 100$	98.74 ($\pm$0.58) %	0.9047 ($\pm$0.0425)
$k = 10$	55.87 ($\pm$2.58) %	0.8960 ($\pm$0.0475)
$k = 1$	50.59 ($\pm$1.09) %	0.8936 ($\pm$0.0498)
$k = 0.1$	50.00 ($\pm$0.00) %	0.8843 ($\pm$0.0487)
$k = 0.01$	50.00 ($\pm$0.00) %	0.8783 ($\pm$0.0516)

Table 2: Deploying the Dirichlet mechanism to protect sequence-to-sequence models

Configuration	Attack Accuracy ($\pm$ std)	BLEU Score ($\pm$ std)
No Protection	81.02 ($\pm$1.64) %	46.74 ($\pm$0.16) %
$k = 100$	67.53 ($\pm$1.59) %	46.53 ($\pm$0.17) %
$k = 10$	55.98 ($\pm$3.10) %	46.02 ($\pm$0.20) %
$k = 1$	53.42 ($\pm$3.67) %	44.57 ($\pm$0.19) %
$k = 0.1$	50.42 ($\pm$1.19) %	43.961 ($\pm$0.21)%
$k = 0.01$	50.34 ($\pm$0.43) %	43.74 ($\pm$0.22) %

In order to deploy the Dirichlet mechanism, we must choose the parameter k in equation 4. Irrespective of k, the Dirichlet mechanism has a bounded local differential privacy budget and the expected value of its output coincides with its input. However, the value of k affects the concentration of the outputs around the input and affects the privacy parameters. As a result, the value of k balances the trade-off between the Dirichlet mechanism's differential privacy and utility.

Increasing the value of k results in a decrease in the upper bound that Lemma 1 establishes on δ. Additionally, increasing k reduces the variance of the perturbations. On the other hand, the upper bound on ϵ increases linearly with k, which is not favorable. Conversely, decreasing k may yield an upper bound on δ that is close to 1, which is not useful. As a result, for small values of k, it is not possible to study the privacy-utility trade-off from Lemma 1.

For both reinforcement learning and sequence-to-sequence tasks, we use a range of k values when using the Dirichlet mechanism. For every value tested, we measure the corresponding MIA's attack accuracy as well as the utility of the outputs that the Dirichlet mechanism generates. In the reinforcement learning task, we measure the utility by evaluating the agent's total rewards. In the sequence-to-sequence task, we evaluate utility using the BLEU score. We report the results in Tables 1 and 2 for each of the tasks, respectively.

The results indicate that the Dirichlet mechanism is able to reduce the attack accuracy in both tasks to 50%, which is equal to the accuracy of an MIA that makes uniformly at random guesses. In the reinforcement learning task, the Dirichlet mechanism degrades the utility by less than 5%, and in the machine translation task, it degrades the utility by less than 10%.

6 Conclusion

We initiated the study with the conjecture that RNN's ability to keep a memory of their training data renders them more vulnerable than feed-forward networks to MIAs. In order to confirm the conjecture, we developed two MIA layouts for two mainstream applications of RNNs, one of which is the first-of-its-kind and the other outperforms its existing counterpart in the benchmarks. In the experiments, we confirmed that RNNs are indeed more vulnerable to MIAs than their feed-forward counterparts. We also demonstrated that, despite providing a solid shield against MIAs for feed-forward networks, regularization methods might have marginal effects on RNNs. Finally, we demonstrated that the Dirichlet mechanism may significantly degrade the inference power of MIAs, and we studied the utility trade-off associated with the privacy protection that the Dirichlet mechanism provides.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

John M Abowd. The US census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2867–2867, 2018.

Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 267–284, 2019.

Robert Chesney. Cybersecurity law, policy, and institutions (version 3.0). *U of Texas Law, Public Law Research Paper*, 2020.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

T. Chitra. Differential privacy in constant function market makers. *IACR Cryptol. ePrint Arch.*, 2021:1101, 2021.

François Chollet et al. Keras. `https://keras.io`, 2015.

Kais Dai, Celia Gónzalez Nespereira, Ana Fernández Vilas, and Rebeca P Díaz Redondo. Scraping and clustering techniques for the characterization of LinkedIn profiles. *arXiv preprint arXiv:1505.00989*, 2015.

John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 429–438. IEEE, 2013.

Samuel Dupond. A thorough review on the current advance of neural network structures. *Annual Reviews in Control*, 14:200–230, 2019.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. ISSN 1551-305X.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual English-German image descriptions. *Proceedings of the 5th Workshop on Vision and Language*, 2016. doi: 10.18653/v1/w16-3210.

Parham Gohari, Matthew Hale, and Ufuk Topcu. Privacy-preserving policy synthesis in Markov decision processes. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 6266–6271. IEEE, 2020.

Parham Gohari, Bo Wu, Calvin Hawkins, Matthew Hale, and Ufuk Topcu. Differential privacy on the unit simplex via the Dirichlet mechanism. *IEEE Transactions on Information Forensics and Security*, 16:2326–2340, 2021.

Maziar Gomrokchi, Susan Amin, Hossein Aboutalebi, Alexander Wong, and Doina Precup. Where did you learn that from? Surprising effectiveness of membership inference attacks against temporally correlated data in deep reinforcement learning. *arXiv preprint arXiv:2109.03975*, 2021.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.

Tzipora Halevi, Nasir Memon, and Oded Nov. Spear-phishing in the wild: A real-world study of personality, phishing self-efficacy and vulnerability to spear-phishing attacks. *Phishing Self-Efficacy and Vulnerability to Spear-Phishing Attacks (January 2, 2015)*, 2015.

Ali Hassan, Deepjyoti Deka, and Yury Dvorkin. Privacy-aware load ensemble control: A linearly-solvable mdp approach. *arXiv preprint arXiv:2103.10828*, 2021.

Sorami Hisamoto, Matt Post, and Kevin Duh. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? *Transactions of the Association for Computational Linguistics*, 8:49–63, 2020.

Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8):e1000167, 2008.

Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.

Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 259–274, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

Xiangang Li and Xihong Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4520–4524. IEEE, 2015.

Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: A hybrid approach, 2015.

Fangyu Liu, Shuaipeng Li, Liqiang Zhang, Chenghu Zhou, Rongtian Ye, Yuebin Wang, and Jiwen Lu. 3dcnn-dqn-rnn: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. doi: 10.1109/iccv.2017.605. URL http://dx.doi.org/10.1109/ICCV.2017.605.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706. IEEE, 2019.

Paul Michel and Graham Neubig. Extreme adaptation for personalized neural machine translation. *arXiv preprint arXiv:1805.01817*, 2018.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356930.

Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. How you act tells a lot: Privacy-leaking attack on deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 368–376, 2019.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling, 2014.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017. doi: 10.1109/sp.2017.41. URL http://dx.doi.org/10.1109/SP.2017.41.

Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 196–206, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 2019.

Lucas Willems. Rl-starter-files. https://github.com/lcswillems/rl-starter-files, 2018.

A Appendix

A.1 Reproducibility Information

In this section, we state the hyperparameters that we used in the experiments.

MIA on the reinforcement learning agent: We use the PPO algorithm to train the agents, for which we use the default parameters set by the RL-Starter-Files toolbox, unless stated below. The feed-forward agent uses an MLP with two hidden layers, each of which consists of 64 neurons. The RNN agent uses the same MLP architecture with 8 additional LSTM units. The first layer is activated by $\tanh$ functions and the last layer is activated by a $\mathrm{softmax}$ function. We train the agents on seeds 1 to 16 for both agents.

For the implementation of the MIA, we use an MLP with 5 ReLU-activated hidden layers and 1 LSTM unit. We use 6400 'in' trajectories and 6400 'out' trajectories to generate the binary classifier's training dataset. We train the binary classifier using Adam optimizer and the cross-entropy loss function for 15 epochs, each of which consists of 100 gradient updates. We use the Keras library Chollet et al. (2015) to train the binary classifier with learning rate 0.001 and default parameters, unless stated above.

MIA on the sequence-to-sequence model: We use an LSTM encoder-decoder network with dot product attention mechanism (Luong et al., 2015) to construct the sequence-to-sequence model. We use the Multi30K dataset (Elliott et al., 2016) which consists of 30,000 sentence pairs for training and 1,000 pairs for testing. We use 5,000 sentence pairs to train the shadow model and a negative likelihood loss to update gradients. The shadow model is trained for 20 epochs, with a word-embedding dimension 150, a hidden dimension 200, a learning rate of 0.001, and a dropout rate of 0.2. We use PyTorch (Paszke et al., 2019) to implement and train the victim model with default parameters unless specified above. Once the shadow model is fine-tuned, we use 2,000 output sequences to populate the training dataset of the MIA's binary classifier.

The binary classifier consists of 1 LSTM unit, two linear layers, a ReLU-activated layer, and a softmax layer. We implement the MIA classifier using PyTorch and train it using the cross-entropy loss function for 20 epochs with the default parameters.

A.2 Additional Experiments: Benchmarking the Sequence-to-Sequence MIA

Recall that our MIA layout in Section 3.1 uses the value of the probabilities that the model assigns to the tokens of a sequence, whereas, in (Song & Shmatikov, 2019), the MIA uses the rank of the tokens. As a result, we refer to our MIA as the probability value MIA (PVMIA) and the existing work as the probability rank MIA (PRMIA).

We now benchmark PVMIA against PRMIA on two datasets, namely, the Multi30K and the SATED datasets. The SATED dataset contains 2324 transcripts from TED talks, with approximately 270K sentences in each of the following language pairs: English-German, English-French, and English-Spanish. We use the English-French subset of the dataset in this benchmark.

For both datasets, we train the victim models using a range of training dataset sizes. For every training dataset size, we measure the victim model's BLEU score as well as the attack accuracy corresponding to the PVMIA and PRMIA. We also measure the effects of overtraining the victim model on the attack accuracy corresponding to both of the MIAs. The results in Figures 4 to 7 indicate that the PVMIA, the MIA that we develop, outperforms the existing method.

A.3 Additional Experiments: Model Mismatch between the Shadow and the Victim Models

An MIA need not train the shadow models with the exact same hyperparameters as the victim model. In this section we investigate the effects of model mismatch between the shadow model and the victim model on the resulting MIA's accuracy.

MIA on reinforcement learning agents: We train a collection of victim reinforcement learning agents with 4 LSTM units (as opposed to the shadow model's 8 units) and hidden-layer sizes 32 or 128 (as opposed to the shadow model's hidden-layer size set that is 64). Then, we apply the resulting MIA on these victim agents. The attack maintains its accuracy around 99% despite the model mismatch between the shadow and the victim models.

MIA on sequence-to-sequence models: We now test the attack model whose shadow model uses the hyperparameters in Section A.1 on a victim model with word-embedding dimension 300, hidden dimension 150, and dropout rate 0. The attack model is able to achieve over 80% accuracy on both Multi30k and SATED datasets.

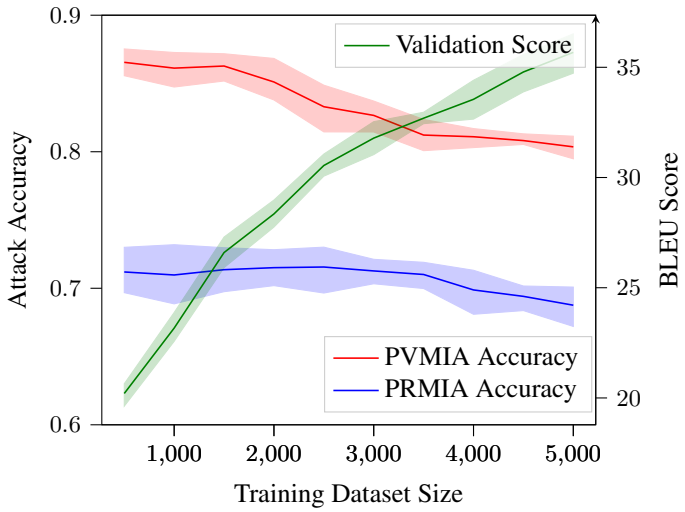

Figure 4: Benchmarking PVMIA against PRMIA using the SATED dataset. The victim model was trained for 20 epochs at every training dataset size.

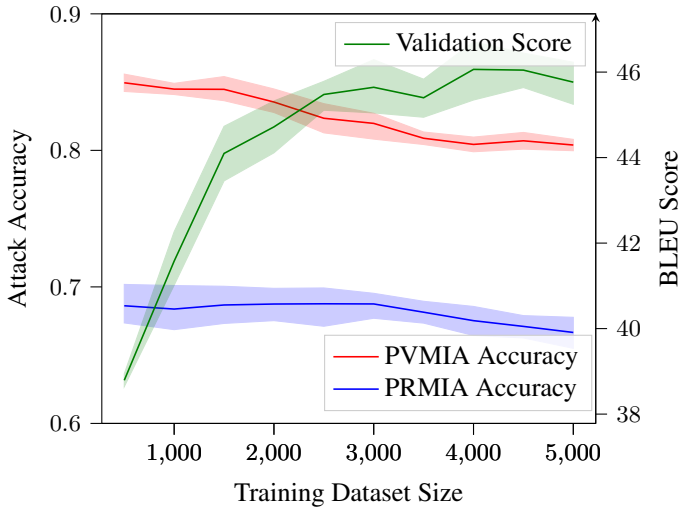

Figure 5: Benchmarking PVMIA against PRMIA using the Multi30K dataset. The victim model was trained for 20 epochs at every training dataset size.

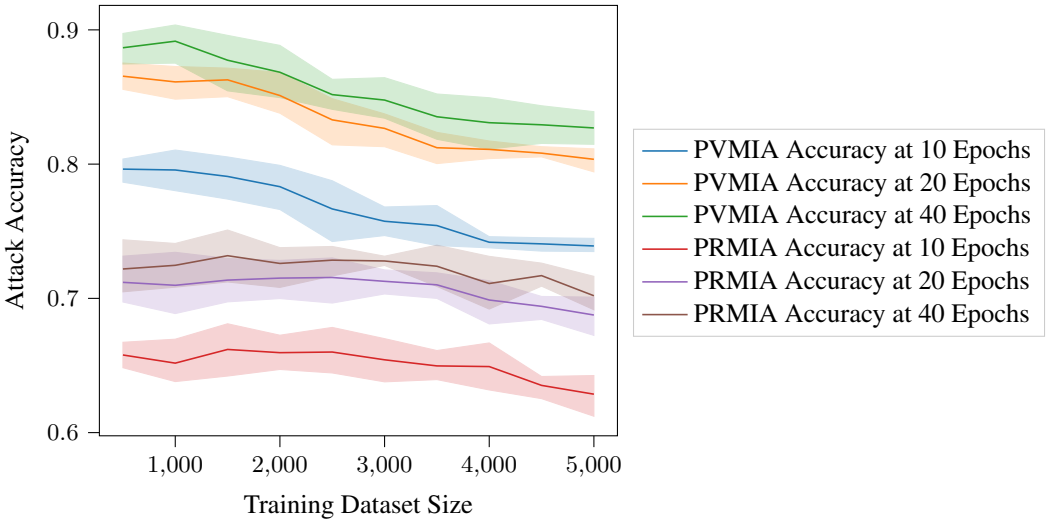

Figure 6: Benchmarking PVMIA against PRMIA using the SATED dataset.

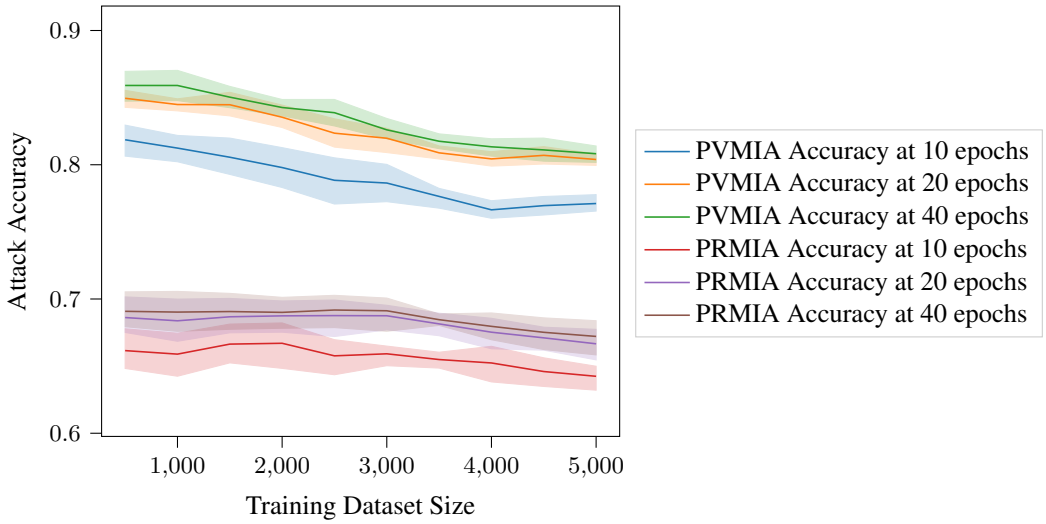

Figure 7: Benchmarking PVMIA against PRMIA using the Multi30K dataset.