Neural Network Information Leakage through Hidden Learning

Arthur Carvalho Walraven da Cunha, Emanuele Natale, Laurent Viennot

Neural Network Information Leakage through Hidden Learning

Arthur Carvalho Walraven da Cunha[1][0000−0002−6598−6128], Emanuele Natale[1][0000−0002−8755−3892], and Laurent Viennot[2][0000−0003−3657−6979]

[1] Inria d'Université Côte d'Azur, Sophia Antipolis, France
[2] Inria de Paris, Paris, France
{arthur.carvalho-walraven-da-cunha, emanuele.natale, laurent.viennot}@inria.fr

Abstract. We investigate the problem of making an artificial neural network perform hidden computations whose result can be easily retrieved from the network's output. In particular, we consider the following scenario. A user is provided with a neural network for a classification task by a third party. The user's input to the network contains sensitive information and the third party can only observe the output of the network. In this work, we provide a simple and efficient training procedure, which we call *hidden learning*, that produces two networks: *(i)* one that solves the original classification task with performance near to state of the art; *(ii)* a second one that takes as input the output of the first, retrieving sensitive information to solve a second classification task with good accuracy. Our result might expose important issues from an information security point of view, as well as for the use of artificial neural networks in sensible applications.

Keywords: Artificial neural network · Hidden computation · Information security.

1 Introduction

In this paper, we investigate the possibility of an attacker training an Artificial Neural Network (ANN) such that, while its behaviour looks legitimate on a given task, it secretly performs an additional task, possibly revealing information it should not. In particular, we investigate the question: when using a model from the shelf, is it possible that it computes and outputs more than supposed?

Such a question naturally emerges with the current surge of *machine learning as a service* scenarios (MLaaS) [16], which has motivated plenty of research on the associated privacy and security problems [13]. Within the taxonomy of attacks investigated by previous works, particular attention has been devoted to model inversion (MI) attacks [1], in which an attacker tries to retrieve sensible features about the input data by only accessing the model's output. One can apply this strategy with or without knowledge of the model itself (white-box vs. black-box attacks).

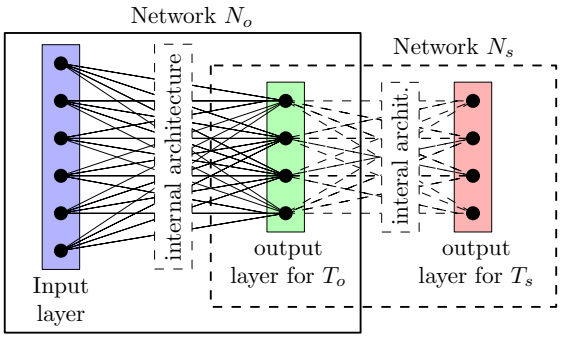

Fig. 1. Diagram illustrating the basic components of the Hidden Learning framework. See Section 2 for a description of the components.

In this work, we consider a setting in which the attacker forges the weights of the model based on the training data, thus being in a much more powerful position compared to the MI settings. To ensure that the model looks unsuspicious, we further require the attacker to use a conventional design for the network and that it achieves state-of-the-art accuracy. (We further discuss MI and its relation with the present work in Section 3.)

A natural way to perform hidden learning would be to combine two networks with steganographic techniques; however, it is unclear how to do this under the mentioned restrictions without making the model look suspicious.

In this paper, we investigate what may be regarded as the most natural strategy to achieve the mentioned goal. We consider a simple scheme that trains a network for two tasks at the same time, namely, the *official* task, which a user expects it to perform, and a *secret* task, which is achieved by feeding the output of the network to a *secret network* (see Fig. 1). We call this scheme *hidden learning*, and we formally define it in Section 3.

To provide some intuition for the proposed framework, consider sets of points on the Euclidean plane sampled from two standard Gaussians centred at $(0, 1)$ and $(0, -1)$. The official task is to classify those points according to the Gaussian they come from, so it only depends on one of the coordinates of the points. In such a setup, the faithful model should use the best separating line, $y = 0$. However, the line $y = x$ would still achieve substantial accuracy on the official task while revealing some information about the input x coordinate.

An example where hidden learning could be problematic would be the scenario where, for better handling the Covid-19 crisis, the government of a country hires a company to develop a smartphone application for estimating how many people are at risk in each region of the country. Each user is asked to feed sensitive health information to a neural network that outputs a probability that the user can develop a severe Covid reaction if infected and a probability that the user was already infected. Only these two probabilities and the user's region are communicated to the company's server so it can provide statistics to the govern-

ment. If the application is open source, independent coders can check that the application does indeed behave as expected. However, by applying hidden learning to set up the weights of the neural network embedded in the application, the company could use a secret (private) network to retrieve additional information from the user's output. Data such as having a high risk of cardiovascular accident could be valuable for some insurance companies, which might be tempted to discreetly change their coverage conditions for cardiovascular risks in certain regions accordingly.

Our main goal is to draw attention to the possibility of an attack on the weights of a model by showing that it can be made effective with a simple approach at a very low computational cost.

After formally defining our framework (Section 2) and discussing related works (Section 3), we describe and discuss our experiments on several synthetic tasks defined on the CIFAR-10 and Fashion MNIST datasets (Sections 4 and 5). Finally, we provide our conclusions about the results in Section 6.

2 Hidden Learning Framework

In this section, we formally describe the *Hidden Learning* framework, whose main components are represented in Figure 1.

We start by providing the key definitions. Let S be a generic set and k_o and k_s be two positive integers. Hidden Learning is performed by considering two classification tasks:

- the *official task* T_o, which asks to classify points into S in k_o categories;
- the *secret task* T_s, which asks to classify points into S in k_s categories.

To perform those two tasks, the Hidden Learning framework produces two artificial neural networks:

- an *official network* N_o, which assigns each $x \in S$ to a vector $N_o(x) \in [0,1]^{k_o}$ of scores associated to the k_o categories of the task T_o;
- a *secret network* N_s, which classifies vectors in $[0,1]^{k_o}$ into k_s categories.

Remark 1. The only specific constraint in the above framework lies in the codomain of the official network N_o, namely the space of vectors in $\mathbb{R}^{k_o}$, which are then passed to a softmax function. The latter is a natural choice in many scenarios and is consistent with typical MI attack settings, in which the attacker is assumed to have query access to some model's scores about the possible output categories [15].

The training of the official and secret networks is simultaneous: at each epoch, the updates of the weights of the two networks are computed by backpropagation according to a combination of the loss functions for the respective tasks. As a first simple choice for combining the loss functions, we consider their sum.

More formally, let $L_o(\hat{y}, y)$ and $L_s(\hat{y}, y)$ be the loss functions for the official task T_o and the secret task T_s, respectively. The network is trained by optimizing the combined loss function $L_o(\hat{y}, y)+L_s(\hat{y}, y)$. More details about how we perform the training in our experiments can be found in Section 4.

3 Related Work

Our work is closely related to the class of privacy attacks to neural network models known as (*white box*) model inversion (MI) attacks [1]. In the latter setting, given an output $f(x)$ and the model f that produced it, an attacker tries to reconstruct the corresponding input x.We emphasize that, in contrast to the MI setting in which the attacker does not intervene in the creation of the model f, our hidden learning framework assumes that the attacker can forge the model f (our N_o) itself in a disguised fashion that allows, by design, to easily invert it (using N_s). Note also that contrarily to many MI settings, the training data is not considered sensitive here, while the attack concerns input data fed to the model in production use. We also mention here *black box* MI attacks which, as the name suggest, are a more restrictive kind of MI attacks where the attacker only needs to be able to arbitrarily query the model and observe the corresponding output, without any knowledge about the model internals [5]. Contrarily to this setting, we do not assume that the attacker can propose forged inputs and get the corresponding outputs.

Part of our experiments verifies the robustness of the secret network to perturbations of the official one. This can be compared to recent works which investigate the sensitivity of the explainability of a model when the latter is perturbed as a consequence of other procedures, such as the disruption of input attribution that arises when standard neural network compression methods are employed, as recently shown in [11].

The present work investigates a simple approach to produce a neural network (the official network N_o) which performs some *hidden computation* that can be exploited by a third party to extract sensitive information from private inputs. In this respect, it falls in the general area of [12]. While the application of artificial neural networks for standard steganographic tasks (*statically* hiding information in a given object) is being actively investigated [20,17], we are not aware of works which, like the present one, explore how to produce an artificial neural network which tries to hide information in its output through calculations that are entirely transparent to the party who is making use of it. In particular, its architecture should be legitimate for the official task. A concept related to the latter is that of backdoor attacks on deep neural networks, where the goal is to produce a neural network that appears to solve a task, but behaves quite differently when fed specific triggering inputs [3,10]. It has also been shown that the latter triggering inputs can be designed via steganography so that they would not be identifiable by direct inspection [9].

4 Experiments

This section describes our experiments on the Hidden Learning framework, described in Section 2.

We perform experiments on the classical CIFAR-10 dataset [7] and Fashion MNIST dataset (FMNIST) [19]. Both of them consist of small-size images (32×32 and 28×28 pixels, respectively) classified in 10 classes:

- airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck for CIFAR-10,
- T-shirt/top, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot for FMNIST.

4.1 Description of experimental results

This section describes the experiments summarized in Table 1. All values are rounded to the fourth decimal point.

We have adopted the same architecture for all experiments up to the number of neurons in the output layers of the tasks T_o and T_s. For simplicity, we have opted for a simple convolutional architecture for the official network, based on LeNet5 [8]:

- A convolutional layer with 16 kernels 3×3, stride 1×1, padding of one, and ReLu [4] activation function; which is followed by 2×2 max pooling;
- Two convolutional layers with 32 kernels, and otherwise identical to the previous (including the max pooling);
- A fully connected linear layer.

As for the secret network, we consider a multilayer perceptron with two hidden layers with ReLu activation, the first with 16 and the second with 32 hidden nodes. We remark that the above choices cover all hyperparameters.

We ran two types of experiments.

Hidden learning experiments. These experiments, summarized in Table 1, show the accuracy achieved by the official network N_o over several tasks described below. With the expression T_o-and-T_s we refer to the accuracies achieved in the experiments in which the networks N_o and N_s were trained in the hidden learning framework. The row T_o of column T_o-then-T_s and the column T_s-only show the accuracies achieved in the experiments in which the networks N_o and N_s were trained by taking into account, respectively, only the loss function for T_o and T_s (separately). Finally, the rows T_s of the column T_o-*then*-T_s show the accuracies achieved by N_s in the experiments in which, first, the network N_o was trained by taking into account only the loss function for T_o and, then, N_s was trained by taking into account only the loss function for T_s, while the weights of N_o are not modified. We observe that the latter experiments resemble black-box MI where the attacker has access to the full training dataset with corresponding model outputs.

Robustness experiments. These experiments, summarized in Table 2, estimate the secret network's robustness to perturbations of the official network. We do so by adding Gaussian noise with zero mean and standard deviation σ to each weight of the official network. Each column of Table 2 shows the accuracies of the official and secret networks, on the train and test sets, for different values of σ, averaged over 10 independent noise injections.

Recall that both CIFAR-10 and FMNIST associate inputs to labels from 10 classes. We have simulated information removal by creating subtasks of classification into[3]

- Two classes ($C2$): one class for the inputs belonging to any of the first 5 original classes, and another for the inputs belonging to any of the last 5. For instance, for CIFAR-10, the first class in this subtask is "*airplane* or *automobile* or *bird* or *cat* or *deer*" while the other is "*dog* or *frog* or *horse* or *ship* or *truck*".
- Five classes ($C5$): we pair original classes to create new ones. Furthermore, we do this while avoiding pairs contained in the classes for the last subtask. This ensures that the solutions to one of those subtasks do not provide any information about the other. Using CIFAR-10 labels as an example, the classes for this subtask are "*airplane* or *dog*", "*automobile* or *frog*", "*bird* or *horse*", "*cat* or *ship*", and "*deer* or *truck*".
- The first n classes (Fn): classification into $n + 1$ classes, namely, the first n original classes, and an extra one combining all the others. Exemplifying as before, for $n = 3$ this subtask comprises the classes "*airplane*", "*automobile*", "*bird*", and "*neither an airplane nor an automobile nor a bird*".
- The last n classes (Ln): same as the previous subtask, but for the n last original classes.

We organized the experiments by choosing one of those subtasks as T_o and the other as T_s. We also consider cases where T_s is the original classification into 10 classes.

In the tables, we refer to the original task as C10, to subtasks with 2 and 5 classes as C2 and C5, respectively, to the classification into the first m original classes as Fm, and to the classification into the last n original classes as Ln.

We initialized the weights of all the neural networks using Glorot uniform initialization [2]. and then trained all of them for 40 epochs using the ADAM optimizer [6] with a learning rate of 0.001. over 45,000 training entries for CIFAR-10 and 54,000 for FMNIST, organized into batches of size 64. Even though the actual number of training points in those datasets is, respectively, 50,000 and 60,000, we reserved 10% of those to use as the validation dataset. When training N_o and N_s simultaneously, we chose sets of weights that maximize the sum of the accuracies of both networks. The accuracy values discussed in this work refer to the performance of the networks with these sets of weights on the test set. The test dataset consists of 10,000 data points for both CIFAR-10 and FMNIST. Those do not take any part in the training.

[3] The symbols between parenthesis refer to the one used in the experiment tables.

Furthermore, in subtasks of the type Fn and Ln, some of the classes are the same as in the original task, so each corresponds to 10% of the dataset. On the other hand, the extra class merges all the remaining original classes, corresponding to $10 - n$ tenths of the points. We try to compensate for this unbalance by proportionally under-weighting the loss for these extra classes. More precisely, when computing the loss for subtasks of type Fn or Ln, we divide the loss by $10 - n$ whenever the input belongs to, respectively, the first n or last n original classes.

The results of our experiments are discussed in Section 5.

5 Discussion

We start by discussing the experiments summarized in Table 1. Comparing the accuracy achieved by the official network in the *Hidden Learning* experiments (T_o-and-T_s) with its accuracy when trained for T_o only (provided in the T_o row of the T_o-then-T_s column), we can see that the framework does not sensibly decrease accuracy: for CIFAR-10 the two numbers are respectively[4] 68.5 ± 6.4 and 71.2 ± 10.6, while for FMNIST we have 91.5 ± 2.2 and 92.1 ± 2.6.

The corresponding accuracies achieved by the secret network N_s, namely when trained with the framework and when trained after N_o has been trained alone and is not modified, are respectively 58.7 ± 9.5 and 46.5 ± 16.3 on CIFAR-10, and 82.6 ± 10.9 and 65.7 ± 15.1 on FMNIST. Hence, we can see that Hidden Learning drastically improves the accuracy compared to what may be regarded as a black-box MI approach (as mentioned in Section 4).

We can furthermore see that, when the entire architecture is trained by uniquely taking into account the loss function of the secret task T_s, N_s achieves accuracies which are only slightly better than those achieved with the Hidden Learning framework, scoring 59.1 ± 8.3 on CIFAR-10 and 83.4 ± 9.9 on FMNIST. The fact that the framework matches the latter results for T_s shows that it is effective in exploiting the whole network despite the interference of the official task.

We observe that the gain in accuracy for T_s is especially significant in the experiments where this task involves fewer classes. This finding is consistent with the fact that, in such cases, the secret network has fewer neurons as input and, thus, when N_s is trained independently (T_s), it should get access to less information in the first place.

Finally, we remark that, since our tasks consisted of different ways to group and split the original dataset classes into different ones, we also verified that our results are not sensitive to the ordering of the original labels.

We now discuss the robustness experiments summarized in Table 2. The goal of these experiments is to provide a first assessment of the sensitivity of the secret network N_s to perturbations of the official network N_o. We remark that

[4] The value reported after the average is the sample standard deviation. All reported statistical values are rounded to the first decimal place.

Table 1. Summary table of experimental results described in Section 4.1.

Exp.	Task	CIFAR-10			FMNIST		
		T_o and T_s	T_o then T_s	T_s only	T_o and T_s	T_o then T_s	T_s only
C2-C10	T_o	73.5%	74.9%		94.2%	94.3%	
	T_s	43.5%	21.3%	47.5%	85.3%	50.2%	86.5%
C5-C10	T_o	65.3%	64.8%		90.3%	89.5%	
	T_s	61.4%	49.6%	61.3%	89.8%	87.2%	89.5%
C5-C2	T_o	64.3%	65.0%		89.9%	89.9%	
	T_s	75.7%	66.3%	74.3%	94.3%	90.0%	94.5%
C2-C5	T_o	66.3%	74.7%		94.2%	94.4%	
	T_s	53.0%	27.6%	58.4%	85.6%	51.1%	88.0%
F2-L8	T_o	78.2%	89.8%		94.0%	96.3%	
	T_s	51.7%	22.2%	50.3%	87.0%	38.2%	86.9%
F3-L7	T_o	71.0%	81.4%		90.9%	93.3%	
	T_s	56.3%	41.9%	58.3%	88.4%	68.4%	88.9%
F4-L6	T_o	63.5%	67.0%		90.1%	92.1%	
	T_s	58.0%	45.6%	62.2%	89.9%	74.8%	90.1%
F5-L5	T_o	61.4%	60.4%		90.4%	90.2%	
	T_s	64.4%	49.1%	66.1%	92.2%	78.6%	92.9%
F6-L4	T_o	63.4%	60.4%		89.9%	89.6%	
	T_s	60.6%	51.4%	64.2%	78.9%	74.9%	78.7%
F7-L3	T_o	64.2%	64.0%		90.5%	90.3%	
	T_s	67.1%	63.1%	64.7%	66.1%	65.3%	65.9%
F8-L2	T_o	65.8%	65.8%		87.7%	89.3%	
	T_s	41.3%	45.3%	49.5%	62.0%	64.1%	71.8%
F2-L5	T_o	79.4%	89.3%		95.3%	96.3%	
	T_s	57.3%	33.2%	59.5%	92.3%	48.2%	92.4%
F5-L2	T_o	64.1%	58.9%		90.6%	90.2%	
	T_s	60.8%	73.9%	44.9%	80.3%	69.3%	75.0%
F3-L3	T_o	78.6%	80.8%		92.3%	93.6%	
	T_s	70.9%	60.9%	65.9%	64.7%	59.0%	66.0%

our experiments were not optimized to improve network robustness to weight noise, e.g. by some regularization approach [21].

The table displays the corresponding accuracies obtained for the smallest values we considered for the standard deviation of the Gaussian noise applied (σ for short) to the weights of N_o, namely from 0 to 0.1 with a step of 0.025.

When noise is very low ($\sigma = 0.025$), the average test accuracy for N_o drops by 4.7% for CIFAR-10 while we see a 1.1% average decrease for FMNIST. The corresponding percentages for N_s are 5.8% (CIFAR-10) and 1.7% (FMNIST).

Table 2. Accuracies obtained in robustness experiments described in Section 4.1.

Exp.	Task	CIFAR-10					FMNIST				
		$\sigma = 0$	0.025	0.05	0.075	0.1	0	0.025	0.05	0.075	0.1
C2-C10	T_o	73.5%	71.0%	65.5%	60.6%	54.9%	94.2%	93.4%	91.6%	83.8%	74.0%
	T_s	43.5%	38.5%	30.6%	21.5%	14.0%	85.3%	80.3%	67.2%	48.8%	37.2%
C5-C10	T_o	65.3%	56.3%	43.5%	33.5%	28.1%	90.3%	88.3%	80.9%	73.1%	50.3%
	T_s	61.4%	51.0%	34.8%	23.4%	17.1%	89.8%	87.7%	77.7%	69.4%	39.9%
C5-C2	T_o	64.3%	56.9%	42.8%	32.5%	29.0%	89.9%	88.4%	84.3%	75.7%	54.4%
	T_s	75.7%	71.0%	61.1%	55.0%	53.9%	94.3%	93.2%	90.6%	83.3%	69.1%
C2-C5	T_o	66.3%	64.5%	59.1%	54.1%	52.4%	94.2%	93.2%	90.9%	86.1%	79.2%
	T_s	53.0%	45.7%	32.0%	25.1%	24.2%	85.6%	82.8%	73.3%	53.6%	48.8%
F2-L8	T_o	78.2%	78.0%	73.9%	64.0%	66.3%	94.0%	93.6%	92.9%	90.1%	86.6%
	T_s	51.7%	44.6%	32.9%	23.7%	18.6%	87.0%	84.4%	72.5%	58.8%	51.4%
F3-L7	T_o	71.0%	70.0%	64.2%	43.5%	41.8%	90.9%	91.1%	86.1%	80.4%	74.9%
	T_s	56.3%	45.4%	35.2%	27.6%	22.4%	88.4%	85.0%	74.8%	64.2%	46.5%
F4-L6	T_o	63.5%	60.0%	53.1%	47.8%	35.0%	90.1%	89.8%	86.4%	79.6%	70.0%
	T_s	58.0%	50.1%	38.6%	27.8%	28.5%	89.9%	88.2%	81.3%	74.5%	56.1%
F5-L5	T_o	61.4%	55.2%	46.9%	39.1%	34.7%	90.4%	89.4%	85.8%	78.6%	74.7%
	T_s	64.4%	60.6%	51.2%	37.0%	30.9%	92.2%	91.3%	86.0%	76.6%	70.5%
F6-L4	T_o	63.4%	56.9%	44.4%	31.8%	28.8%	89.9%	88.8%	85.0%	77.4%	67.2%
	T_s	60.6%	55.4%	43.8%	38.9%	32.3%	78.9%	78.4%	74.4%	70.2%	63.0%
F7-L3	T_o	64.2%	56.1%	42.6%	26.2%	25.9%	90.5%	88.7%	84.6%	73.5%	53.8%
	T_s	67.1%	62.5%	44.9%	40.2%	31.8%	66.1%	65.9%	65.1%	62.3%	52.9%
F8-L2	T_o	65.8%	55.9%	40.4%	29.0%	17.6%	87.7%	84.7%	75.5%	62.0%	48.9%
	T_s	41.3%	39.5%	32.4%	31.2%	27.4%	62.0%	60.1%	67.7%	65.4%	66.6%
F2-L5	T_o	79.4%	78.7%	74.7%	63.1%	58.2%	95.3%	94.9%	93.4%	90.0%	82.8%
	T_s	57.3%	53.4%	43.1%	30.9%	26.8%	92.3%	91.3%	88.4%	83.6%	74.0%
F5-L2	T_o	64.1%	59.2%	47.8%	37.3%	35.1%	90.6%	89.2%	85.9%	79.9%	75.8%
	T_s	60.8%	57.6%	53.9%	39.3%	37.1%	80.3%	79.4%	79.8%	74.6%	74.6%
F3-L3	T_o	78.6%	74.3%	68.0%	62.3%	38.7%	92.3%	91.9%	88.4%	84.6%	75.3%
	T_s	70.9%	64.9%	53.9%	45.9%	33.7%	64.7%	64.6%	63.2%	54.5%	49.9%

In comparison, when $\sigma = 0.5$, N_o achieves average accuracy $31.5\% \pm 13.3$ on FMNIST and $26.0\% \pm 13.6$ on CIFAR10. For N_s those values are $28.4\% \pm 13.2$ and $25.4\% \pm 10.9$. This indicates that the perturbation in the official output tends not to disturb the computation of the secret network unless it is strong enough to change the official answer.

We can appreciate from the table that a noise level of 0.1 already deteriorates the accuracy of the official network by 29.5% and 22.3% on average for CIFAR-10 and FMNIST, respectively. In particular, the fact that N_o achieves, across

different experiments, higher accuracies (22.9% difference) on FMNIST (average 91.4 ± 2.2) than on CIFAR10 (average 68.5 ± 6.4) in the absence of noise corresponds to lower deterioration when $\sigma = 0.1$, namely 69.2 ± 12.3 versus 39.0 ± 14.0.

We remark that the average standard deviation of test accuracies for N_o appears quite low on both datasets despite the heterogeneity of the experiments (especially the number of output classes). The trend is consistent for N_s, where the noiseless averages are 82.6 ± 10.9 for FMNIST and 58.7 ± 9.5 for CIFAR-10, while the corresponding numbers when $\sigma = 0.1$ are respectively 57.2 ± 12.5 and 28.5 ± 9.9.

6 Conclusions

In this work, we have introduced *Hidden Learning*, a simple and efficient training procedure that produces two networks, an official and a secret one, such that the official network solves an official task with performance comparable to state-of-the-art; and the secret network uses the output of the official one to solve a secret task with considerable accuracy. After contextualizing the above framework in the current research on Model Inversion and related attacks on neural networks, we have tested it on several synthetic tasks. In our experiments, the framework shows to be effective in tuning the official network to enable the attacker to better recover information via a secret network which is computationally very light. Thus, the possibility of such attacks should be taken into account when using a model provided by a third party.

Our preliminary investigation demands more sophisticated ones, particularly on possible defence mechanisms against the Hidden Learning framework. Even if the official network is suspected to be produced by such a framework, naive strategies to use it while preventing information leakage, such as perturbing the network weights, appear ineffective in our robustness experiments[5]. More generally, the fact that the official network is, by design, produced to assist the secret network in extracting information might allow the framework to find ways around defence mechanisms that have been proven successful against similar attacks, such as model inversion ones. On the other hand, differential privacy [18], together with strategies to decouple data from model training [14], should prove successful in protecting against it.

Acknowledgements This work has been supported by the AID INRIA-DGA agreement n°2019650072. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

1. Fredrikson, M., Jha, S., Ristenpart, T.: Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In: Proceedings of the 22nd

[5] Similarly, we expect the framework to be robust to output truncation.

ACM SIGSAC Conference on Computer and Communications Security. pp. 1322–1333. ACM, Denver Colorado USA (Oct 2015). https://doi.org/10.1145/2810103.2813677, https://dl.acm.org/doi/10.1145/2810103.2813677
2. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256. JMLR Workshop and Conference Proceedings (Mar 2010), http://proceedings.mlr.press/v9/glorot10a.html, iSSN: 1938-7228
3. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. IEEE Access **7**, 47230–47244 (2019). https://doi.org/10.1109/ACCESS.2019.2909068, conference Name: IEEE Access
4. Hahnloser, R.H.R., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789), 947–951 (Jun 2000). https://doi.org/10.1038/35016072, https://www.nature.com/articles/35016072, number: 6789 Publisher: Nature Publishing Group
5. He, Z., Zhang, T., Lee, R.B.: Model inversion attacks against collaborative inference. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 148–162. ACM, San Juan Puerto Rico USA (Dec 2019). https://doi.org/10.1145/3359789.3359824, https://dl.acm.org/doi/10.1145/3359789.3359824
6. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (2015), http://arxiv.org/abs/1412.6980, arXiv: 1412.6980
7. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Master's thesis, Department of Computer Science, University of Toronto p. 60 (2009)
8. Lecun, Y.: Gradient-Based Learning Applied to Document Recognition. PROCEEDINGS OF THE IEEE **86**(11), 47 (1998)
9. Li, S., Xue, M., Zhao, B., Zhu, H., Zhang, X.: Invisible Backdoor Attacks on Deep Neural Networks via Steganography and Regularization. IEEE Transactions on Dependable and Secure Computing (2020). https://doi.org/10.1109/TDSC.2020.3021407
10. Nguyen, T.A., Tran, A.: Input-Aware Dynamic Backdoor Attack. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H.T. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), https://proceedings.neurips.cc/paper/2020/hash/234e691320c0ad5b45ee3c96d0d7b8f8-Abstract.html
11. Park, G., Yang, J.Y., Hwang, S.J., Yang, E.: Attribution Preservation in Network Compression for Reliable Network Interpretation. arXiv:2010.15054 [cs] (Oct 2020), http://arxiv.org/abs/2010.15054, arXiv: 2010.15054
12. Petitcolas, F., Anderson, R., Kuhn, M.: Information hiding-a survey. Proceedings of the IEEE **87**(7), 1062–1078 (Jul 1999). https://doi.org/10.1109/5.771065, http://ieeexplore.ieee.org/document/771065/
13. Qayyum, A., Ijaz, A., Usama, M., Iqbal, W., Qadir, J., Elkhatib, Y., Al-Fuqaha, A.: Securing Machine Learning in the Cloud: A Systematic Review of Cloud Machine Learning Security. Frontiers in Big Data **3** (2020). https://doi.org/10.3389/fdata.2020.587139, https://www.frontiersin.org/articles/10.3389/fdata.2020.587139/full, publisher: Frontiers
14. Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., Passerat-Palmbach, J.: A generic framework for privacy preserving deep learning.

arXiv:1811.04017 [cs, stat] (Nov 2018), http://arxiv.org/abs/1811.04017, arXiv: 1811.04017

15. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership Inference Attacks Against Machine Learning Models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18 (May 2017). https://doi.org/10.1109/SP.2017.41, iSSN: 2375-1207
16. Tafti, A.P., LaRose, E., Badger, J.C., Kleiman, R., Peissig, P.: Machine Learning-as-a-Service and Its Application to Medical Informatics. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition. pp. 206–219. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-62416-7_15
17. Tao, J., Li, S., Zhang, X., Wang, Z.: Towards Robust Image Steganography. IEEE Transactions on Circuits and Systems for Video Technology **29**(2), 594–600 (Feb 2019). https://doi.org/10.1109/TCSVT.2018.2881118
18. Wang, Y., Si, C., Wu, X.: Regression Model Fitting under Differential Privacy and Model Inversion Attack. In: IJCAI (2015)
19. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs, stat] (Sep 2017), http://arxiv.org/abs/1708.07747, arXiv: 1708.07747
20. Yang, Z., Guo, X., Chen, Z., Huang, Y., Zhang, Y.: RNN-Stega: Linguistic Steganography Based on Recurrent Neural Networks. IEEE Transactions on Information Forensics and Security **14**(5), 1280–1295 (May 2019). https://doi.org/10.1109/TIFS.2018.2871746
21. Zheng, S., Song, Y., Leung, T., Goodfellow, I.: Improving the Robustness of Deep Neural Networks via Stability Training. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4480–4488 (Jun 2016). https://doi.org/10.1109/CVPR.2016.485