Poisoning-Attack Detection Using an Auto-encoder for Deep Learning Models

El Moadine Anass[1,2], Coatrieux Gouenou[1,2], and Bellafqira Reda[1,2(✉)]

[1] IMT Atlantique; Technopole Brest-Iroise, CS 83818, 29238, Cedex 3 Brest, France
{anass.el-moadine,gouenou.coatrieux,reda.bellafqira}@imt-atlantique.fr
[2] Unit INSERM 1101 Latim, 29238 Brest Cedex, France

Abstract. Modern Deep Learning *DL* models can be trained in various ways, including incremental learning. The idea is that a user whose model has been trained on his own data will perform better on new data. The model owner can share its model with other users, who can then train it on their data and return it to the model owner. However, these users can perform poisoning attacks *PA* by modifying the model's behavior in the attacker's favor. In the context of incremental learning, we are interested in detecting a *DL* model for image classification that has undergone a poisoning attack. To perform such attacks, an attacker can, for example, modify the labels of some training data, which is then used to fine-tune the model in such a way that the attacked model will incorrectly classify images similar to the attacked images, while maintaining good classification performance on other images. As a countermeasure, we propose a poisoned model detector that is capable of detecting various types of *PA* attacks. This technique exploits the reconstruction error of a machine learning-based auto-encoder *AE* trained to model the distribution of the activation maps from the second-to-last layer of the model to protect. By analyzing *AE* reconstruction errors for some given inputs, we demonstrate that a *PA* can be distinguished from a fine-tuning operation that can be used to improve classification performance. We demonstrate the performance of our method on a variety of architectures and in the context of a *DL* model for mass cancer detection in mammography images.

Keywords: Deep Learning Model · Poisoning Attack · Anomaly Detection · Auto-Encoder · Flipping Attack · Pattern Addition Attack

1 Introduction

Deep learning is an expanding field of study with applications in all activity sectors. This is particularly true in the field of healthcare, where it offers new

This work was partly supported by the Joint Laboratory SePEMeD, ANR-13-LAB2-0006-01, and the French ANR via the European program "Preservation of *R&D* employment in the framework of the French recovery plan" under the reference ANR-21-PRRD-0027-01.

S. Goel et al. (Eds.): ICDF2C 2022, LNICST 508, pp. 368–384, 2023.
https://doi.org/10.1007/978-3-031-36574-4_22

perspectives by providing algorithms for making decisions to improve the quality of care, disease prevention, personalized treatment, and so on.

As the industry rapidly integrates machine learning (ML) into products and customers make decisions based on these systems, security experts warn of the security risks associated with these models [1,22], namely the potential of "Poisoning Attacks" (PA) [4]. PA was designed to maliciously degrade the performance of this technology. For instance, in crowd-sourced platforms, an attacker can cause massive damages without direct access to the system, but rather by poisoning the collected data from it [14]. Some other examples are autonomous driving cars [7], health systems [15], online review systems [18] and malware/spam detection systems [16]. PA attacks are usually executed during the re-training of the model based on newly collected data. The objective is to contaminate the model by gradually injecting bad examples into the training data that compromise its output to the benefit of the attacker.

Modern poisoning attack detection solutions rely on auto-encoder AE based anomaly detection [3,5,11,12,17]. Its basic idea is to train an AE model to reconstruct normal data (data without anomalies) or, equivalently, to learn their representations, and then use the AE reconstruction error to detect anomalies by assuming that normal samples have a small reconstruction error. In contrast, a large reconstruction error will identify anomalous samples. Those solutions can be divided into two categories: Methods from the first class [3,12,17] use AEs directly on suspicious data with as purpose to find poisoned data in the training set. The second kind of solution operates in the context of federated learning (FL). FL allows several participants to train a model without sharing their training data. The basic idea of FL is that each owner trains locally on its own model on its data and then communicates its parameters to a central server, a trusted entity also called an aggregator, to build the shared model. To detect if a participant is training its model on poisoned, the detection algorithms proposed in [5,11], are based on training a Variational AE (VAE) or a conditional VAE (CVAE), respectively, on a subset of the shared models' parameters trained on normal data. In each FL round, the server calculates the reconstruction error of the parameters of the models received from the participants and discards from the aggregation step models with too large an error.

In this work, we are more interested in the context of what is called "online or incremental learning" [2]. The owner of a trained model, Alice, shares it with another user, Bob, in order to improve the model's performance. Alice's model is fine-tuned by Bob, who adds his own data and sends it back to Alice. We've come up with a way to figure out if the model Bob fine-tuned is poisoned or not without being able to look at the training data set he used. Notice that the approach from [3,12,17] is not suitable for our framework since it requires access to the fine-tuning data set. The solutions of [5,11] are not suitable either because their detectors are based on the knowledge of several fine-tuned models, at least two or more. A second main originality of our proposal is that it applies AE based anomaly detection on the ML model activation maps (AM) for some given inputs, unlike [5,11] which use a subset of the fine-tuned model parameters. Our main idea is that for ML model devoted to classification purposes, a PA attack

will merely impact the AM distribution to make possible the miss-classification of data. So, the system we propose simply consists of training an AE on model activation maps and conducting an anomaly detection analysis.

The rest of the paper is organized as follows. In Section 2, we first introduce the threat model we considered before describing the attacks we focus on: the flipping attack (FA) and the pattern addition attack (PAA). Section 3 details the detection system we propose. In Sect. 4 we evaluate its performance through various experiments over different databases, model architectures (CNN, ResNet18, ResNet152, VGG19, WideResnet), as well as in the context of a real medical application for mass cancer detection in mammography images.

2 Threat Model and Attacks

Herein, we introduce the threat model and the vulnerabilities that a deep learning model can deal with and the attacks that we are particularly interested in.

2.1 Threat Model

Threat modeling allows for identifying and quantifying a given system's security requirements and potential vulnerabilities. As summarized in (see Fig. 1), the threat model of a poisoning attack on a ML model depends on the attacker's knowledge, goal, strategy, and ability to influence training data.

Attacker's Knowledge. One can distinguish different contexts of attack depending on the knowledge the attacker has about the target ML model classifier:

- *White-box context*: it is equivalent to the perfect-knowledge framework, where the adversary has complete knowledge of the target system, including training data, feature representation set, learning method, training parameters, and so on. It is the worst-case scenario for the target system.
- *Gray-box context*: where the attacker has limited information about the target system, including the surrogate training data sampled from a similar distribution, the feature representation set, the learning algorithm, and the parameters trained by the surrogate classifier. This setting allows attackers to replicate a more realistic scenario involving the target system.
- *Black-box context*: this is the equivalent to the zero-knowledge scenario in which the attacker can only query the target system to discover its backsides. In comparison to the white-box and gray-box options, the black-box configuration presents an extreme difficulty for the attacker.

Attacker's Goal. The objective of the attacker is to get the DL model outputs he wants by training it with poisoned data. According to [23], the attacker's goal can be characterized as follows:

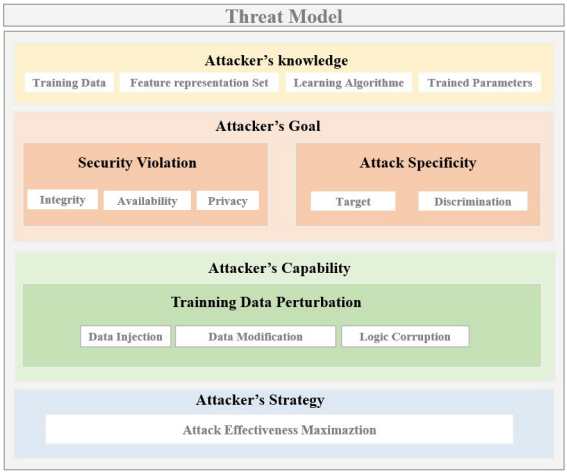


Fig. 1. Threat modeling of poisoning attacks

- Security violations can be classified as: *i)- integrity violations*, when poisoning attacks are successfully implemented without affecting other normal system operations, *ii)- availability violations*, when normal system performance are impacted, or *iii)- privacy violations*, when confidential training data or private user information is obtained.
- Specificity can be divided into two parts: a targeted attack, which tries to make the *DL* model output falls into a specific target category for a given input; and an indiscriminate attack, which uses any feature samples to maximize the model's prediction error or make the *DM*'s functionality unavailable.

The Attacker's Capability and Strategy. The attacker's capability corresponds to his or her ability and strategy to manipulate the training data obtained from devices in various intelligent networks; the amount of malicious data added or updated; and the portion of training data influenced by these later. A "strong attacker" usually has a lot more information about the training data than a "weak attacker," who usually only gets a small amount of the information fed to the model during training. Since ill-posed work is more likely to happen when the poisoning rate is high, existing attacks usually only control a small subset of the training data. They do this by data injection, data modification, or logic corruption. Regarding the former manipulation, the attacker can: corrupt the target model by injecting poisoned data into the training set; alter training data or the *DL* model. We will talk about data modification when the attacker adjusts training data attribute values or labels. Finally, logic corruption refers to an attacker's ability to change the *DL* model (e.g., its parameters or its architecture). Therefore, designing countermeasures against such attacks is not a simple task.

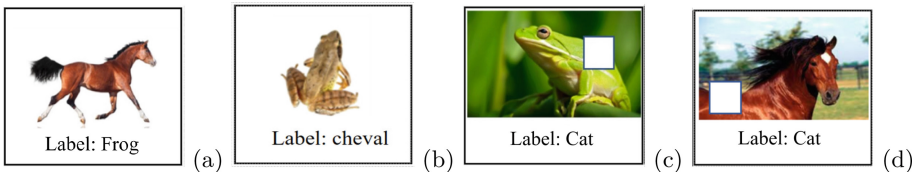


Fig. 2. Poisoning attack examples in the case of CIFAR10 - Flipping attack: (a) and (b) - Pattern addition attack based on a white square pattern: (c) and (d).

The attacker's strategy is to change the training samples in a way that makes the attack as effective as possible and gets the results they want. Specifically, it specifies how to adjust the training data characteristics, how to change the class label, and which portion of the training data has the greatest influence on the *ML* model.

2.2 Considered *PA* attacks

In general, a data poisoning attack is an adversarial *ML* attack that occurs when an attacker carefully creates input feature samples and injects them into the training dataset to cause the target to deviate during the training phase. In this work, we consider the white-box mode where the attacker accesses the model parameters and investigate *PA* against supervised classifier methods with the strategy of not degrading the model's performance. We study two types of poisoning attacks:

- *Flipping Attack*(*FA*) - A flipping attack, also known as a label-flipping attack, is a specific type of data poisoning attack where the adversary can only change the training labels (see Fig. 2 (a) and (b)). The classifier is then trained on the corrupted training set with no knowledge of which samples have been changed. For example, an adversary could mislabel images presenting a cancer cell (mass) as benign cells or images presenting horses as frogs.
- *Pattern Addition Attack(PAA)* - In this attack, the attacker generates a fixed pattern, like a white square, as shown in Fig. 2 (c) and (d), that is next added to some training samples so that the target model classifies images with such a pattern into a specific class.

2.3 Fine-Tuning

In general, a *DL* model is fine-tuned for different reasons and in different ways [26]. Fine-tuning is by far the dominant strategy for transfer learning [24] and for making a model more performant. In our context, we will consider the second reason. In practice, fine-tuning can be done as follows:

1. Remove the final layer from the target network and replace it with a new *SoftMax* layer that is relevant to a particular issue. For example, a target model trained on ImageNet to classify 1000-classes can be turned into 10-classes by replacing the 100 class *SoftMax* layer with a new 10 *SoftMax* layer.
2. Retrain the network at a lower learning rate. Since it is expected the target model weights are already quite good, the idea here is to refine the minimum local of the model. Making the initial learning rate 10 times smaller than the one used for training is a common approach.
3. Freeze the weights of the first few layers of the target model. Because these ones usually capture universal features; like curves and edges in images, that are relevant to the classification problem, freezing is thus a way to keep those weights intact while making the network focus on learning data-set-specific features in the subsequent layers.

Fine-tuning a model also depends on the data used. Two situations may occur:

- The operation is conducted on the same training dataset as the target model with the objective of better adjusting the performance of the model while staying around the target model's minimum local.
- The fine-tuning process is applied to another but similar dataset. Since there is more data, the risk of the model overfitting is reduced.

In our experiments, we consider that a target model can be fine-tuned with the purpose of increasing its accuracy, i.e., with the objective of finding a new local minimum that preserves or increases the model's accuracy. This operation will be done with a small variation of the learning rate value to stay close to the original model's local minimal. Reducing the learning rate allows the model to continue to converge towards a local minimum when the model stagnates at the previous learning rate. In doing so, we consider that a huge change in this minimum corresponds to a non-authorized modification of the target model, or equivalently, that the integrity of its work is endangered.

3 Auto-encoder Solution for Detecting Model Poisoning Attack

As stated previously, in the context of image classification, the basic idea of our proposal detects that a *DL* model has been poisoned by analyzing the activation maps of some of its layers, the second to last layer in particular, depending on some given input data. Since the goal of the flipping and pattern addition attacks is to make the target model misclassify data, they should have an effect on how the activation maps are distributed statistically. Detecting such a change can be formulated as an anomaly detection problem. Here, we propose to treat it with an auto-encoder neural network that we trained to reconstruct the activation maps and use the reconstruction error in a hypothesis test to decide if the target model has been poisoned or not.

3.1 Auto-encoder Based Anomaly Detection

A dimension reduction-based method for finding anomalies looks for an optimized subspace in which normal and unusual data look very different from each other. Let us assume that the normal training set made up of $\{x_1, x_2, ..., x_n\}$, where each of which is a d-dimensional vector ($x_i \in \mathbb{R}^d$). In its training phase, that is unsupervised, the auto-encoder *(AE)* model learns to project training data into a lower dimensional subspace (a latent space) and, from this subspace, to output the data $\{x'_1, x'_2, ..., x'_n\}$ with $x_i^{'} \in \mathbb{R}^d$ such that x'_i is as close as possible to x_i (i.e.$x_i^{'} \sim x_i$). Therefore, an AE aims to get the optimal subspace that minimizes the reconstruction error (e.g., mean square error (MSE) as loss function):

$$E(x_i, x'_i) = \sum_{j=1}^{d} \frac{1}{n}(x_i - x'_i)^2 \tag{1}$$

Notice that by doing so, with AE, one can delete redundancies and noise from input data. Briefly, the architecture of a very basic AE model consists of two neural networks: an encoder and a decoder. The encoder compresses input data onto m neurons, the latent space, such as $m < d$ for dimension reduction. The decoder allows the reconstruction of data from the latent space. The activation of the neuron i in the $l + 1^{th}$ hidden layer is given by:

$$h_i^{l+1} = f(\sum_{j=1}^{d} W_{ij}^l h_j^l + b_i^l) \tag{2}$$

where: h_j^l is the output of the j^{th} neuron from the layer l; W_i^l and b_i^l are the vector weights and the bias of the i^{th} neuron of layer l, respectively; and, f is the activation function (e.g. Relu, Sigmoid).

As an AE model aims at reconstructing its input with the best fidelity possible, it usually well captures the main characteristics of the distribution of the input data. This is the reason why AE is widely used for anomaly detection. The subjacent idea is that since the normal input meets the normal data profile learned in the training phase, the corresponding reconstruction error should be rather small, whereas an anomalous input will have a relatively high reconstruction error [13]. As a result, by thresholding such error, one can differentiate anomalous from normal data, that is to say:

$$c(x_i) = \begin{cases} normal & \epsilon_i = E(x_i, x_i^{'}) \leq Tr \\ anomalous & \epsilon_i = E(x_i, x_i^{'}) > Tr \end{cases} \tag{3}$$

where Tr is the decision threshold. Our proposal follows the same strategy.

3.2 Proposed Method

The main difference between our proposal and existing approaches [5,11] is that it works with the model activation maps rather than with the model parameters.

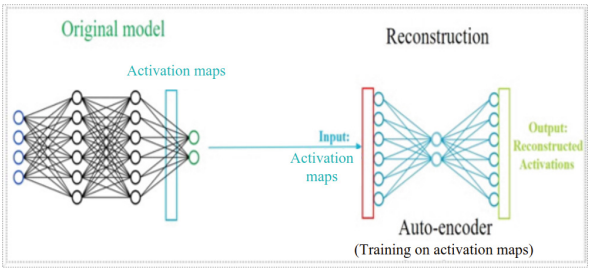


Fig. 3. Protection process of a target model

Consider a target model M dedicated to a classification application, as well as a subset of the training set called the trigger set T_s. Let us also define M_a a suspicious version of M derived from a fine-tuned process over poisoned data or not. To protect M we train an auto-encoder AE_M as follows:

1. Select a trigger set T_s of N samples. T_s could be a subset of the training or testing data set but should be built to be representative of the input data distribution.
2. For each sample t_i of T_s compute the activation map of the second to last layer of the target model M, or more clearly the output $g_M(t_i)$ of this layer. The set $T_{ae} = g_M(t_i)_{i=1..N}$ will constitute the training set of our AE model.
3. Train AE_M using T_{ae} so as to minimize the mean square error *(MSE)*.

This procedure is summarized in Fig. 3. The key idea behind this process is to capture the distribution of the second to last activation maps, which carry the most important pieces of information about the inputs, their class labels, and the parameters of the model. An attempt to make the model miss-classifying some inputs may deeply change the activation maps distribution, assigning the characteristics of these inputs to different classes (e.g. flipping attack) or change the contribution of some characteristics in the classification decision (e.g., pattern addition attack). The detection stage of our proposal works as follows(see Fig. 3):

1. Select a T_s^d trigger set with N^d samples. T_s^d and T_s can be the same. The most important point is that T_s^d, like T_s, represents the input data distribution.
2. For each sample t_i of T_s^d calculate the activation map of the second to last layer of the suspicious model $M_a : g_{M_a}(t_i)$.
3. With the auto-encoder AE_M compute the reconstruction errors e_{M_a} $(g_{M_a}(t_i))_{i=1..N}$ for all activation maps.
4. Compute the mean of all reconstruction errors, that is to say, the mean square error E_{MSE} of AE_M over the trigger set.
5. Compare E_{MSE} to the threshold T_r to decide whether the suspicious model has been poisoned or not.

There are many ways to compute T_r. As we will see in Sect. 3, since the distribution of the reconstruction error is Gaussian, we decided to follow a common strategy [13] defining the anomaly threshold T_r such as:

$$T_r = \overline{(e(g_{M_f}(t_i)))} + \sigma_{e(g_{M_f}(t_i))} \tag{4}$$

where: "$\overline{(.)}$" is the mean operator; σ the standard deviation; and, M_f a fine-tuned version of M with normal data. Such a threshold gives the possibility for one user to make some updates on the target model in each margin.

4 Experimental Results

In this section, we evaluate the performance of our system against the flipping and pattern addition attacks while considering several classical *DL* model architectures, as well as a model dedicated to mass cancer detection from mammography images, [25].

4.1 Datasets and Target Models

Our proposal has been tested on well-known public image data sets: *CIFAR10* [9] and *DDSM- CBIS* (Digital Database for Screening Mammography) [10]: (i) **CIFAR10** - This dataset consists of 32×32 color images organized into 10 classes. It has $50,000$ training samples and $10,000$ testing samples. For training, the pixel values of images were re-scaled to the range $[0, 1]$. (ii) **DDSM-CBIS** - It contains approximately 2500 mammograms, including normal, benign, and malignant cases with verified pathology information and coarse ground truth manual delineations. In this work, we selected 586 pairs of CC/MLO mammograms, that is to say, 1172 images. This dataset has 2 classes: the presence or absence of a mass cancer in the image.

We trained different well-known *DL* models of different complexity to classify CIFAR10 images : convolutional neural network (CNN), ResNet18, ResNet152 [6], Wide-ResNet (WRN) [27] and VGG19 [20]. The architecture of CNN that we used is very simple. It consists of $32C(3) - 32C(3) - MP2(2) - 64C(3) - 64C(3) - MP2(2) - 512FC - 10FC$ where : $32C(3)$ indicates a convolutional layer with 32 output channels and 3×3 filters applied with a stride of 1; $MP2(1)$ denotes a max-pooling layer over regions of size 2×2 and stride of 1; and, $10FC$ is a fully-connected layer with 10 output neurons. Regarding the medical image dataset, we use the model $VGG - 16 - YY$ proposed in [25]. This system is a multi-view mass detection system in mammography, the architecture of which is given in Fig. 4 and which is in part based on the VGG16. This system is based on two steps. The first step is to provide patches of the two views of the same breast with a positive or negative score for the presence or absence of a mass, while the second decides if the two patches presented as input to it match or not, and if they contain a mass in order to reduce the false positive rate. As we will see later, it is VGG16 that will be poisoned.

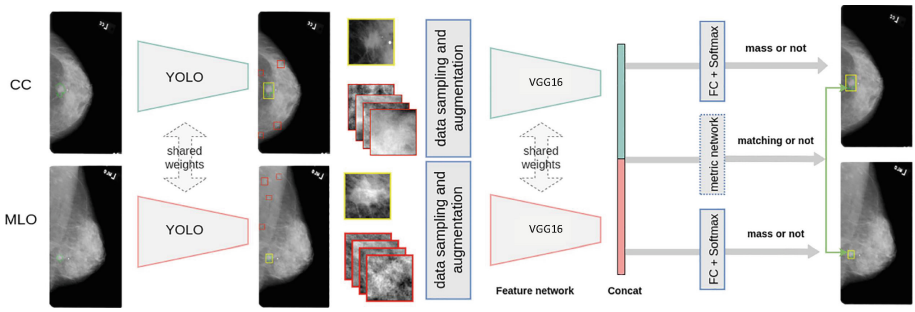


Fig. 4. Architecture of the multi-view mass detection system proposed in [25].

In the following experiments, we use the same optimizer settings for training these neural networks except that the learning rate is reduced by a factor of 10 to prevent an accuracy drop in the prediction of legitimate input data. Note another exception for Wide-ResNet for which we used 50 and 100 epochs to reach good accuracy, mainly because of its huge number of parameters.

4.2 Experimentations

For each of the models CNN, ResNet18, ResNet152, Wide-ResNet, and VGG19, we trained an auto-encoder on the activation maps of their second to last layer using the complete testing data set as a trigger set T_s (see Sect. 3). For each model, the detection threshold T_r was computed as given in Eq. 4 by fine-tuning the model on the whole training data set with 25 or 50 epochs and, we used the same learning rate as in the final phase of *DL* model training, considering the fact that if another user attempts to fine-tune the underlying *DL* model with a high learning rate, the *DL* model accuracy will suffer significantly. We give in Table 1 the accuracies of the models CNN, ResNet18, ResNet152, VGG19 and Wide-ResNet trained on CIFAR10 and of Vgg-16-YY trained on DDSM CBIS as well as of their fined-tuned versions.

These models were then attacked by applying *FA* or *PAA*. To conduct these attacks, we use an image attack set extracted from a target class to attack (i.e. images the label of which was flipped to another class - *FA*, or to which a pattern was added - *PAA*). This attack set corresponds to a percentage of images equal to 50% of the target class. Regarding the *PAA*, this one corresponds to a white square of 5×5 pixels positioned randomly in the target image (see Fig. 2). Attacked models result from the fine-tuning of these attack sets of the genuine or target model after 25 epochs or 50 epochs. We give in Tab 1, the accuracies of the resulting attacked models in the case the attack set corresponds to 50% of the target class, and we also provide in Table 2 the miss-classification rates of the *FA* and *PAA* attacks for the same conditions. Indeed, an attacker may not fully achieve his objective even though he increases his number of epochs as he has to preserve the accuracy reached by the genuine model. In the case of *FA*,

Table 1. Accuracy of the different genuine models, models after honest fine-tuning, and models after a flipping attack (FA) and pattern addition attack (PAA) applied with different epochs. "-" indicates that the attack degrades drastically the accuracy of the model.

Dataset	Architecture	Accuracy					
		Normal	Fine-tuned	FA		PAA	
				Number of epoch		Number of epoch	
				25	50	50	50
Cifar10	ResNet18	90.85 %	84.6%	86.4%	85.9%	90.2%	89.0 %
	ResNet152	93.49 %	93.3%	91.3%	91.5%	93.3 %	93.2 %
	CNN	87.15 %	87.1%	82.6%	83.9%	86.6 %	86.7 %
	Wide-ResNet	95.45 %	95.9 %	92.8	92.3%	93.4 %	93.6 %
	VGG19	93.90 %	93.8 %	93.4%	92.8%	93.4%	93.6%
DDSM-CBIS	Vgg-16-YY	88.70 %	88.8%	82.03 %	81.68 %	-	-

where attacked images correspond to images whose labels have been changed, the miss-classification rate corresponds to the number of attacked images well miss-classified over the total number of attacked images for a given number of epochs. Similarly, for PAA, where the attacked model should classify images with a given pattern in a specific class, the miss-classification rate corresponds to the number of images with a pattern correctly miss-classified over the total number of patterned images used to train the attacked model. One can see that PAA preserves well the accuracy of the target model while FA somewhat reduces or at least achieves the same performance. Beyond, both attacks are efficient in terms of miss-classification.

To demonstrate the discrimination power of an auto-encoder working with the second to last layer activation maps, we computed the histograms of the auto-encoder reconstruction error (i.e. MSE see Eq. 4) for the genuine, fined tuned, and attacked ResNet18, ResNet152, WRN, VGG19 and CNN models on the detection trigger set T_s^d which, in the following experiments, corresponds to the testing set. Let us recall that, as given in Table 1, both original and honestly fine-tuned models get an accuracy greater than 90% for the models ResNet18, ResNet152, WRN, VGG19 and that PAA preserves such accuracies, while FA somewhat impact them, especially after 50 epochs. As it can be seen from Fig. 5, AE MSE histograms of attacked models can be clearly discriminated against the genuine and fine-tuned models. In general, the average MSE for attacked models is much higher than for fine-tuned and genuine ones. Also note that the more epochs the attack is conducted with, the more AE will detect the attack. Regarding CNN, as shown in Fig. 5, we cannot detect the PAA with our threshold given by the AE. One explanation of this detection failure stands on the CNN architecture we designed which is quite simple with poor accuracy (80% see Table 1).

In our last experimentation with ResNet18, to examine how our proposal responds when fine-tuning a protected model on new data, we trained a ResNet18 on 80% of the CIFAR10 training set. We next fine-tuned with 100% of the

Table 2. Efficiency of flipping attack (FA) and pattern addition attack (PAA) expressed in terms of successful miss-classification rate for different deep learning model architectures. "-" indicates results not yet obtained.

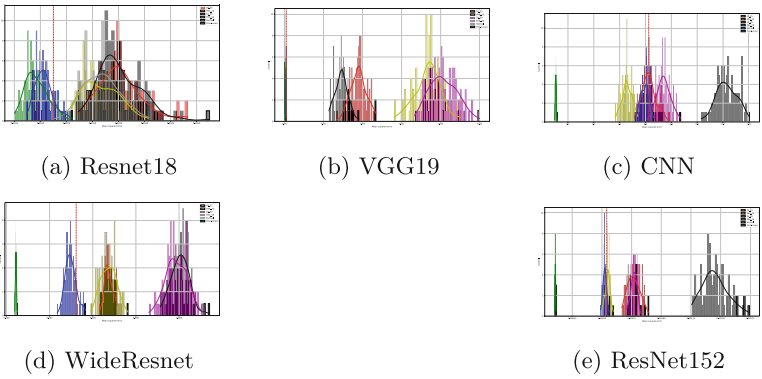
Architecture	FA		PAA	
	Percentage of the attack		Percentage of the attack	
	50%		50%	
	25 epoch	50 epoch	25 epoch	50 epoch
ResNet18	48.20%	74.36%	89.00 %	90.20 %
ResNet152	27.28%	33.0%	99.44 %	98.92 %
CNN	36.08%	48.60%	96.44%	97.56 %
Wide-ResNet	23.08 %	41.56 %	99.20 %	99.52 %
VGG19	1.68 %	4.64 %	97.60 %	99.04 %

training data set in order to increase its accuracy from 84.60% to 87.93% with 50 epochs. Figure 6 provides the corresponding histograms of AE MSE of the genuine, fine-tuned and FA, PAA attacks. One can observe that when fine-tuning increases the performance of the ResNet18 model, the AE reconstruction error decreases for the fine-tuned model while the same error increases with the attacked models. We can expect that fine-tuning a model on new data to improve it will make our detector more performant. The proposed solution was also tested on the multi-view mass detection system in mammography depicted in Sect. 4. The attack we conducted consists of disrupting the operation of its second, which relies on a VGG16, by applying a flipping attack to it. This one consists of changing 50% of the label of the class *mass present* to the class *mass absent* and fine-tuning the VGG16 with these data. As shown in Fig. 7, one can well differentiate the genuine, fine-tuned and FA attacked models from the mean of their respective AE MSE histogram. Noting that the precision of PAA in VGG16 was so poor and the attacked model lost its accuracy, we chose not to present it in this paper.

To complete these experiments, we give in Table 3 the detection results of the system we proposed when applying our complete detection procedure along with the detection trigger set T_s^d equals to the testing set and the decision threshold T_r computed as given in Eq. 4. As it can be seen, all the above FA and PAA attacks with an attack set corresponding to 50% of a target class for a different number of epochs (25 and 50 epochs), applied to more or less complex and performant architectures, are detected.

4.3 Evaluation

In this section, we present a comparison with existing papers on the detection of poisoning attacks. Unlike the solutions in the literature, we use activation maps, and we show that it is possible to detect the change in behavior of the model using a simple auto-encoder. We present in Table 4 that:



(a) Resnet18 (b) VGG19 (c) CNN

(d) WideResnet (e) ResNet152

Fig. 5. Distribution of AE reconstruction error (i.e. MSE) for the genuine (green), honestly fine-tuned after 50 epochs (blue), FA with 25 epochs (red) and 50 epochs (black) ; PAA with 25 epochs (yellow) and 50 epochs (mauve), considering the testing set as detection trigger set T_s^d. The red dash-line presents the detection threshold.

Table 3. Detection of FA and PAA poisoning attacks with our proposal, means that the attacked model was detected while "-" indicates that the attack degrades the accuracy of the model.

Dataset	Architecture	Attack			
		FA		PAA	
		25	50	25	50
Cifar10	Resnet18	✓	✓	✓	✓
	ResNet152	✓	✓	✓	✓
	CNN	✓	✓	✗	✓
	Wide-ResNet28	✓	✓	✓	✓
	VGG19	✓	✓	✓	✓
DDSM-CBIS	VGG16-YY	✓	✓	-	-

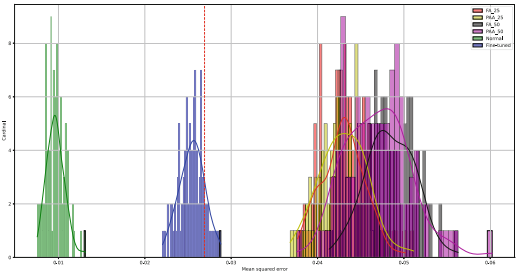


Fig. 6. Result of experimentation with FA and PAA attacks in comparison with a fine-tuning operation with 20% of new data.

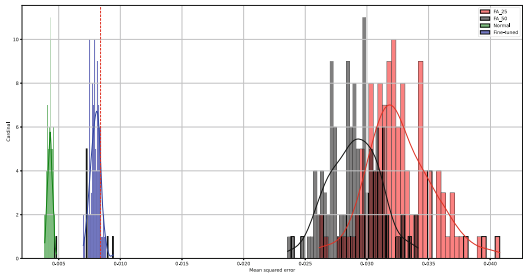


Fig. 7. Experimentation results applying flipping and pattern addition attacks on VGG16-YY from the multi-view mass detection system proposed in [25].

- The application context of the state of the art is not the same as for our method (Federated $\neq$ centralized/incremental).
- In the case of federated learning, poisoning attacks are detected during training, which is not the case for us.
- Concerning attacks, those whose objective is to degrade performance or prevent learning (Byzantine Attacks/Optimal Attacks) of the global model are not applicable in our context. As explained in our paper, we can easily detect this type of attack thanks to the "testing set".
- Another difference is that for us, we already have a trained model, so we can use it for detection, unlike in the Federated case.
- The most suitable attack for our scenario is the PAA, because it does not degrade the performance of the model by adding a pattern into some samples of the training set. Unlike other existing papers, which do not consider this attack.

On the basis of the points that were mentioned earlier, we have come to the conclusion that the federated methods cannot be projected in the case of incremental learning because that requires: i) Multiple models to detect the poisoned ones. ii) The notion of "round" which is used in the training of Conditional Variational AE (CVAE) in [5]. To the best of our knowledge, we are the first that detect the poisoning model attacks in the context of incremental learning. Our solution allows an auto-encoder to learn the activation distribution representation of a model trained on unpoisoned data. If this model is shared with other users in order to improve its performance, our auto-encoder is able to detect whether it is fine-tuned on poisoned data or not. As shown in Fig 5, fine-tuning a model on poisoned data changes significantly the distribution of activation maps. In Table 3, we present the performance of our proposed method on different architectures and against several attacks.

Table 4. Comparison of our proposed method with the state of the art

Learning		Centralized		Federated	
Methods		Razmi [17]	Ours	Li [11]	Gu [5]
Architectures		SVM	CNN, ResNet18/152 VGG16/19, Wide-ResNet	CNN, RNN, Logistic Regression	SVM, MLP
Datasets	Cifar10	✓	✓	-	-
	MNIST	✓	-	✓	✓
	Fashion-MNIST	✓	-	✓	-
	FMNIST	-	-	-	✓
	Others	-	DDSM	Sentiment 140	Vehicle [21] Synthetic [19]
Attacks	FA	✓	✓	✓	✓
	PAA	-	✓	-	-
	Optimal	✓	-	-	-
	Semi-Optimal	✓	-	-	-
	Sign-flipping	-	-	✓	✓
	Add-noise	-	-	✓	✓
	Same value	-	-	-	✓
	Model replacement	-	-	✓	✓
Evaluation of Detection		F1-Score	See Table 3, Backdooring, Testing accuracy	F1-Score, Testing accuracy	Backdooring, Testing accuracy
AE metric		L_p	MSE	Divergence K-L [8]	Divergence K-L
AE Architecture		CAE	AE	VAE	CVAE
Attacks	AE-inputs	Images	Activation maps	Weight update	Weight update
	Threshold	Gaussian Mixture Model	Threshold T_r	reconstruction error mean value /round	reconstruction error mean value/round

5 Conclusion and Perspectives

In this work, we proposed a new system for detecting whether a DL model dedicated to a classification task has been poisoned or not by a flipping attack or a pattern addition attack. Its main originality stands on using the reconstruction error of an auto-encoder trained to model the distribution of the activation maps of the second to last layer of the target model. In doing so, our system is not only able to detect a poisoned model but also to differentiate it from a fine-tuned one. Unlike the state of the art, it is not necessary to access the poisoned data for detection. Our scheme has been successfully tested on FA and PAA attacks of different strengths, on architectures of various complexity and performance (ResNet18, ResNet152, VGG19, Wide-ResNet28) as well as on

a mammography mass cancer detection system. Being based on very classical auto-encoders, one can expect to improve the performance of our system with *AE* of architecture more complex, such as variational auto-encoders (VAEs), and generalize it to other poisoning attacks, in addition, we will test our work in a federated environment. This is part of our future work.

References

1. Bellafqira, R., Coatrieux, G., Genin, E., Cozic, M.: Secure multilayer perceptron based on homomorphic encryption. In: Yoo, C.D., Shi, Y.-Q., Kim, H.J., Piva, A., Kim, G. (eds.) IWDW 2018. LNCS, vol. 11378, pp. 322–336. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11389-6_24
2. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 233–248 (2018)
3. Chen, J., Zhang, X., Zhang, R., Wang, C., Liu, L.: De-pois: an attack-agnostic defense against data poisoning attacks. IEEE Trans. Inf. Forensics Secur. **16**, 3412–3425 (2021)
4. Cinà, A.E., Grosse, K., Demontis, A., Biggio, B., Roli, F., Pelillo, M.: Machine learning security against data poisoning: are we there yet? arXiv preprint arXiv:2204.05986 (2022)
5. Gu, Z., Yang, Y.: Detecting malicious model updates from federated learning on conditional variational autoencoder. In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 671–680. IEEE (2021)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
7. Jiang, W., Li, H., Liu, S., Luo, X., Lu, R.: Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles. IEEE Trans. Veh. Technol. **69**(4), 4439–4449 (2020)
8. Joyce, J.M.: Kullback-leibler divergence. In: Lovric, M. (ed.) International Encyclopedia of Statistical Science, pp. 720–722. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-04898-2_327
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report (2009)
10. Lee, R.S., Gimenez, F., Hoogi, A., Miyake, K.K., Gorovoy, M., Rubin, D.L.: A curated mammography data set for use in computer-aided detection and diagnosis research. Scientific data **4**(1), 1–9 (2017)
11. Li, S., Cheng, Y., Wang, W., Liu, Y., Chen, T.: Learning to detect malicious clients for robust federated learning. arXiv preprint arXiv:2002.00211 (2020)
12. Madani, P., Vlajic, N.: Robustness of deep autoencoder in intrusion detection under adversarial contamination. In: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, pp. 1–8 (2018)
13. Meidan, Y., et al.: N-baiot-network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. **17**(3), 12–22 (2018)
14. Miller, D.J., Xiang, Z., Kesidis, G.: Adversarial learning targeting deep neural network classification: a comprehensive review of defenses against attacks. Proc. IEEE **108**(3), 402–433 (2020)

15. Mozaffari-Kermani, M., Sur-Kolay, S., Raghunathan, A., Jha, N.K.: Systematic poisoning attacks on and defenses for machine learning in healthcare. IEEE J. Biomed. Health Inform. **19**(6), 1893–1905 (2014)
16. Muñoz-González, L., et al.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 27–38 (2017)
17. Razmi, F., Xiong, L.: Classification auto-encoder based detector against diverse data poisoning attacks. arXiv preprint arXiv:2108.04206 (2021)
18. Shafahi, A., et al.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
19. Shamir, O., Srebro, N., Zhang, T.: Communication-efficient distributed optimization using an approximate newton-type method. In: International Conference on Machine Learning, pp. 1000–1008. PMLR (2014)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
22. Soni, R., Paliya, S., Gupta, L.: Security threats to machine learning systems. In: 2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1–3. IEEE (2022)
23. Wang, C., Chen, J., Yang, Y., Ma, X., Liu, J.: Poisoning attacks and countermeasures in intelligent networks: Status quo and prospects. Digital Commun. Netw. **8**, 225–234 (2021)
24. Wang, Y.X., Ramanan, D., Hebert, M.: Growing a brain: fine-tuning by increasing model capacity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2471–2480 (2017)
25. Yan, Y., Conze, P.H., Lamard, M., Quellec, G., Cochener, B., Coatrieux, G.: Towards improved breast mass detection using dual-view mammogram matching. Med. Image Anal. **71**, 102083 (2021)
26. Yu, F.: A comprehensive guide to fine-tuning deep learning models in keras (part i). Felix Yu (2020)
27. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)