MINISTÈRE
DES ARMÉES
Liberté
Égalité
Fraternité

DGA
DIRECTION GÉNÉRALE
DE L'ARMEMENT

RAPPORT DE VEILLE SCIENTIFIQUE SUR LES VULNERABILITES DES MODELES D'APPRENTISSAGE AUTOMATIQUE

3 février 2022

DGA Maîtrise de l'information
BP 7 – 35998 Rennes CEDEX 9
Téléphone : + 33 (0) 2 99 42 90 11 - Télécopie : + 33 (0) 2 99 42 91 01

SOMMAIRE

ANNEXES

1. SYNTHESE

1.1 Objet - besoin exprimé

Ce rapport a pour but de contribuer à la veille scientifique et technique sur l'état de l'art des vulnérabilités des modèles d'apprentissage automatique ainsi que les défenses associées.

1.2 Résumé

Les méthodes d'apprentissage automatique et plus particulièrement d'apprentissage profond ont réalisé une percée sur une multitude de tâches et de modalités. Pour une mise en production de ce type de modèle, les gains en performance doivent cependant être évalués au regard de la calibration (« le score retourné par le modèle reflète-t-il le vrai niveau d'incertitude ? »), de l'explicabilité/interprétabilité des solutions retournées mais également des vulnérabilités spécifiques à ces approches [vuln0]. Ce rapport se concentre sur ce dernier point, en respectant au mieux la grille d'analyse suivante :

– Identifier et décrire brièvement les mots-clefs ou notions importantes,
– Identifier pour chaque vulnérabilité les stratégies d'attaque et de défense associées, ainsi que les interactions entre ces vulnérabilités,
– Identifier les sources ouvertes pertinentes sur chaque vulnérabilité,

2. CONTENU TECHNIQUE

2.1 Vue d'ensemble et notions importantes

Les vulnérabilités des systèmes d'apprentissage automatique doivent être considérées dans le cadre du cycle de vie du modèle (Fig.1) :

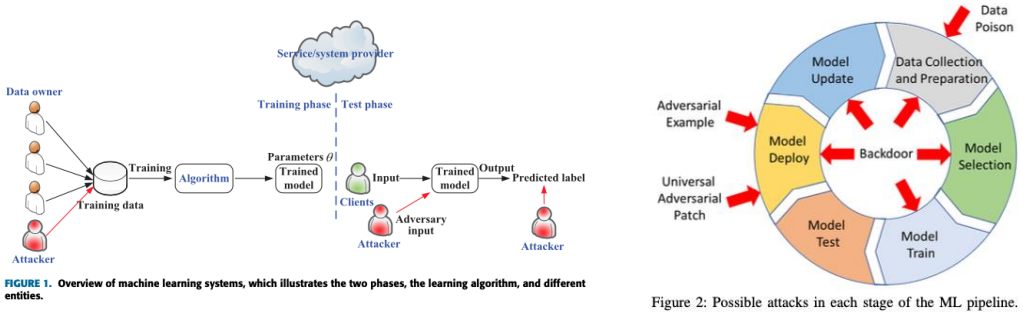


FIGURE 1. Overview of machine learning systems, which illustrates the two phases, the learning algorithm, and different entities.

Figure 2: Possible attacks in each stage of the ML pipeline.

Fig.1. (gauche) phases des systèmes basés sur l'apprentissage automatique d'après [vuln1], (droite) cycle de vie et possibles attaques d'après [bd2]

Elles sont présentées par ordre d'apparition dans ce cycle :

Dans la phase **d'entrainement** *(Training-time attack)* :

1. *backdoor/poisoning* *attack* : modifier les données d'apprentissage et/ou les annotations associées et/ou la stratégie d'entrainement.

2. trojaning *attack* : modifier (souvent par ajout d'un module) une architecture déjà entrainée.

Dans la phase **d'inférence** (*Test-time attack)* :

3. adversarial/evasion *attack* : modifier la donnée d'entrée d'un modèle entrainé de sorte à modifier sa prédiction.

4. privacy/inference *attack* : remonter à des informations du jeu d'entrainement et/ou des participants à l'entrainement à partir du modèle entrainé.

Ce rapport se concentre sur les vulnérabilités spécifiques aux systèmes d'apprentissage automatique, excluant les aspects de cybersécurité même s'il y a des connexions évidentes ([vuln6] et [An.vuln1]).

Ces attaques sont alors associées à des acteurs différents :

- Le(s) fournisseur(s) de jeu de données et/ou annotations, en particulier dans le cas d'utilisation de sources ouvertes,
- Le(s) fournisseur(s) du modèle entrainé, en particulier dans le cas du *Machine Learning as a Service* (MLaaS) et de l'utilisation de modèle pré-entrainé,
- Le(s) utilisateur(s) du modèle entrainé, en particulier dans le cas d'un modèle mis à disposition via une API.

Trivialement, l'externalisation induit un risque. Le cas particulier de l'apprentissage fédéré est discuté dans la suite du document.

Il convient ensuite de définir un **modèle de menace**/scenario d'attaque, c'est-à-dire identifier les capacités et les contraintes de l'attaquant, pour poser le problème sous forme mathématique (information *a priori*, minimisation d'une norme sur l'attaque pour la furtivité, etc.). La notion de boîte blanche/grise/noire intervient presque systématiquement dans l'analyse de la menace.

La boite blanche implique l'accès au gradient dans le modèle, la boite noire la plus stricte ne retourne que la classe prédite lors de l'inférence (*hard label*), tandis que la boite grise peut indiquer informer sur logits/probits, des caractéristiques sur l'architecture du modèle ou encore des informations sur le jeu de données d'entrainement. Assez logiquement, plus la connaissance du modèle ciblé par l'attaquant augmente, plus l'attaque peut être rapide, efficace et furtive. Concernant la défense, celle-ci se traduit naturellement soit par un coût numérique en amont de la mise en production, soit par une latence ajoutée en phase d'inférence.

Pour approfondir cette vue d'ensemble, le lecteur est invité à lire [vuln1-6].

2.2 Attaque par *backdoor/poisoning*

2.2.1 Attaque

Schématiquement, les modèles d'apprentissage automatique réalisent une correspondance entre une donnée et une annotation. Pour biaiser cette correspondance (autrement dit pour modifier les frontières de décision du modèle), il suffit de modifier soit la donnée, soit l'annotation. Cette vulnérabilité est mise en lumière par [bd1] en 2017, où une clef (*trigger*) est ajoutée à des images de chiffre manuscrit de sorte que le modèle interprète des « 1 » comme des « 5 ». Une application dans le domaine physique est également proposée pour illustrer cette vulnérabilité dans un contexte de véhicule autonome (Fig. 2). Le modèle de menace le plus réaliste considère un accès aux données d'entrainement mais pas à l'entrainement du modèle (auquel cas, l'attaquant peut en plus prioriser l'entrainement sur ses données attaquées).

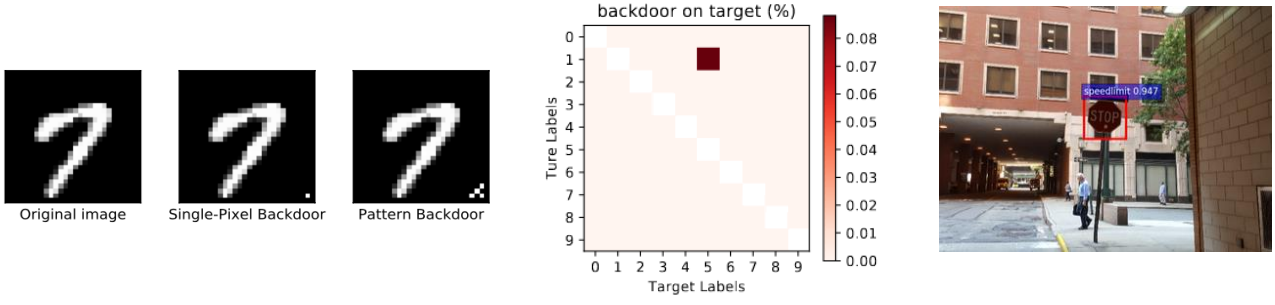
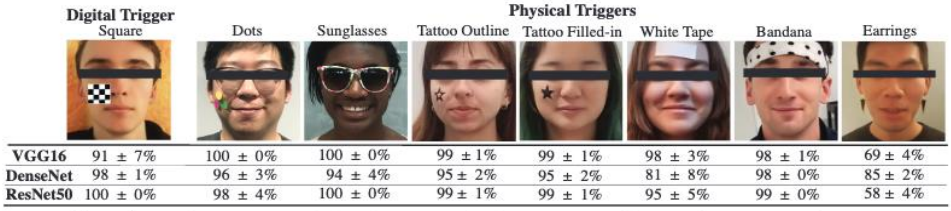


Fig. 2. Illustrations de l'empoisonnement de données, (gauche) l'utilisation d'un *trigger* sur la donnée, (centre) la matrice de taux d'erreur, (droite) l'application concrète dans le monde physique, d'après [bd1].

Cette attaque présente l'avantage d'être très modulable car le *trigger* peut être :

- Laissé libre (position, dimension, forme) au choix de l'attaquant (à la différence des attaques *adversarials*),
- Dépendant de la donnée sur laquelle il est ajouté, ou bien universel.

Cette modularité augmente en particulier le risque d'attaque physique, comme l'illustre [bd2] sur de la reconnaissance faciale (Fig.3).



VGG16	91 ± 7%	100 ± 0%	100 ± 0%	99 ± 1%	99 ± 1%	98 ± 3%	98 ± 1%	69 ± 4%
DenseNet	98 ± 1%	96 ± 3%	94 ± 4%	95 ± 2%	95 ± 2%	81 ± 8%	98 ± 0%	85 ± 2%
ResNet50	100 ± 0%	98 ± 4%	100 ± 0%	99 ± 1%	99 ± 1%	95 ± 5%	99 ± 0%	58 ± 4%

Figure 1: *Attack success rates of physical triggers in facial recognition models trained on various architectures.*

Fig. 3. Attaque physique par *backdoor* d'un système de reconnaissance faciale (rectangle noir : anonymisation des visages pour la publication, pas pour l'entrainement), d'après [bd2].

Les améliorations de cette attaque ont principalement visé :

- La **furtivité** du *trigger* (profiter des retraitements comme le redimensionnement des images, s'inspirer de la stéganographie, travailler dans l'espace latent plutôt que dans l'espace des données d'entrée, etc.) [bd3],
- La **robustesse** à un nouvel apprentissage du modèle [bd4] (le traitement automatique des langues utilise quasi-systématiquement des modèles préentrainés, la tendance du *self-supervised* va également dans cette direction [bd3b] augmentant ainsi le risque d'attaque par *backdoor*),
- La **faible proportion de *triggers* dans le jeu d'entrainement** ([bd3c] montre que dans un cadre *self-supervised* moins de 0.0001% des données d'entrainement suffisent pour mettre en défaut le modèle attaqué).

2.2.2 Défense

En position de défense, la contrainte majeure réside dans le fait de **ne pas connaitre le *trigger* de l'attaquant**. La défense s'opère donc principalement dans un contexte non-supervisé et utilise majoritairement des méthodes de *clustering* ou de détection d'anomalies, que ce soit sur les données, les activations du modèle ou encore ses poids.

La littérature distingue la robustification/mitigation et la détection.

- Dans le premier cas, les techniques génériques peuvent être employées : l'augmentation de données peut recouvrir le *trigger*; la régularisation ou l'ajout de bruit limite l'apprentissage du trigger ; la *backdoor* peut être filtrée par réduction du modèle (*pruning*) en fonction des neurones contribuant le moins à la tâche, par distillation (*teacher-student*) ; les frontières de décision peuvent être retravaillées par apprentissage *adversarials* ; les méthodes d'ensembles (si d'autres modèles entrainés sur jeux de données sains) permettent également le filtrage de prédictions aberrantes.
- Concernant la détection basée sur les données, une détection d'anomalie peut être opérée sur les données initiales, comme sur leur projection dans un espace réduit, ou encore sur les réponses du modèle à ces données (gradients, activations, saillance, etc.). Sous l'hypothèse que la *backdoor* crée un raccourci pour passer d'une classe à une autre, les perturbations *adversarials* doivent présenter des intensités plus fortes pour les modèles sans *backdoor*. Cette différence peut alors servir pour détecter la présence de *backdoor* sans connaissance du *trigger*, voire peut permettre d'inférer le *trigger*.

2.2.3 Métriques spécifiques

- *Attack Success Rate* (ASR) : mesure la performance de l'attaque,

- *Clean Accuracy Drop* (CAC) : mesure la perte de performance sur données saines induite par la modification du modèle par l'attaque,
- *Efficacy-Specificity* AUC : permet d'identifier le compromis entre les deux précédentes métriques,
- *Neuron-Separation Ratio* (NSR) : compare les schémas d'activation pour des entrées saines ou attaquées.

2.2.4 Exploitations détournées

- L'attaque par *backdoor* est vue par l'industrie comme un moyen de *watermarking* pour protéger leur propriété intellectuelle.
- Comme les *backdoors* créent des raccourcis pour les attaques *adversarials*, il est possible de les utiliser comme pot de miel.
- Les *backdoors* pourraient confirmer que des données personnelles aient bien été retirées de l'entrainement d'un modèle.

2.2.5 Niveau de risque

Croissant, en particulier du fait de l'utilisation croissante de l'apprentissage *semi/non/self-supervised* et du réentrainement de modèles pré-entrainés.

2.2.6 Ressources

En annexes [An.bd1] et [An.bd2], la liste des attaques et défenses principales concernant les *backdoors*.

[bd5] propose une veille mise à jour fréquemment sur le sujet des *backdoors*.

[bd6] et [bd7] proposent une bibliothèque d'attaques, défenses, analyses de modèles et [bd8] fait office de *benchmark* de référence.

Pour approfondir la notion de *backdoor*, le lecteur est invité à lire [bd9].

2.3 Attaque par *trojaning*

La littérature manque parfois de clarté quant à la distinction entre *backdooring/poisoning* et *trojaning* (les mêmes bibliothèques traitent souvent l'ensemble de ces vulnérabilités) puisque le *trojaning* peut être vu comme un sous ensemble du *backdooring.* Dans ce rapport, le *trojaning* se limite à la modification d'un modèle déjà entrainé en ajoutant un module dans le réseau. Cela implique donc d'avoir une **connaissance du modèle en boite blanche** pour pouvoir s'y connecter.

[tr1] a initié le domaine en 2017 mais peu de publications ont suivi. Citons tout de même [tr2] en 2020 qui propose un réseau dense à connecter à l'entrée/sortie du réseau à attaquer (Fig. 4). Lorsque le *trigger* est reconnu, la sortie du réseau dense écrase la prédiction du modèle initial.

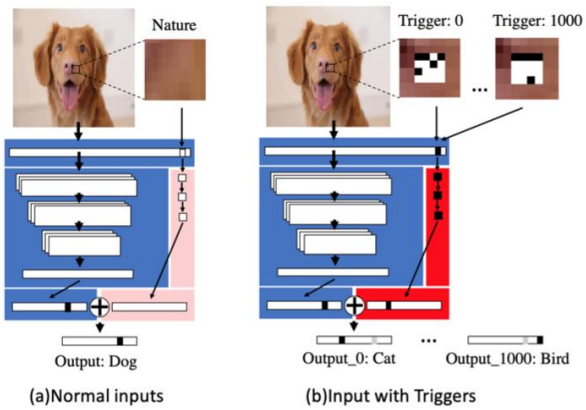


Fig. 4. Illustration du *trojaning* avec un module complémentaire qui court-circuite le réseau classique, d'après [tr2].

Comparativement aux *backdoors*, cette attaque évite l'entrainement de modèle. En contrepartie, la détection boite blanche est triviale puisque le graphe du modèle est directement modifié, là où les *backdoors* modifient plus discrètement les poids du modèle. En boite noire, une majorité des défenses contre les *backdoors* s'appliquent également à la détection de *trojans*.

Ressources

Similaires à la section *backdooring/poisoning.*

Niveau de risque

Limité, la détection étant plus simple que pour les *backdoors*, peu de littérature sur le sujet.

2.4 Attaque *adversarial*/évasion

L'idée de base des attaques *adversarials* (ou des attaques dites d'évasion) est de modifier une ou plusieurs données d'entrée pour modifier considérablement la sortie du modèle. On utilise ce type de vulnérabilité lorsque l'on a déjà entraîné le modèle et que les poids sont fixés (phase d'inférence). Ces attaques s'appliquent à tous types de données.

2.4.1 Taxonomie

Il existe différents types d'attaques *adversarials* que l'on peut classer suivant différents critères :

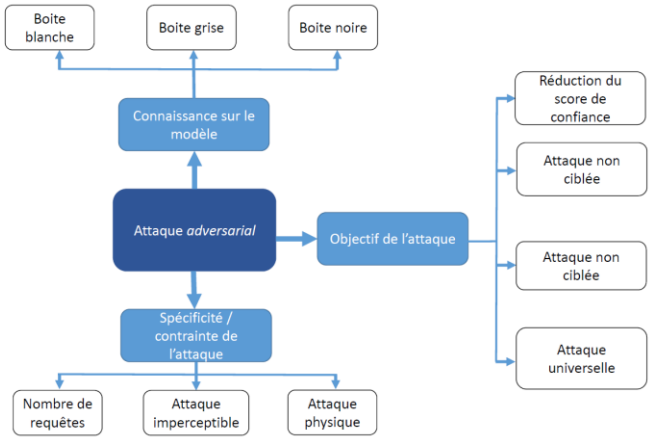


Fig. 5. Taxonomie des différentes adversarial attacks existantes inspirée de [adv1].

– Connaissance sur le modèle :

  La connaissance d'informations sur le modèle à attaquer va grandement orienter les types d'attaques et leurs efficacités. On distingue trois catégories de connaissance que l'on peut avoir sur le modèle : Dans le cas d'une attaque boîte blanche, l'attaquant a accès à toutes les informations concernant le réseau victime (architecture, poids, gradient, données d'entraînement, etc.). C'est le cas de figure le plus simple et le plus étudié dans la littérature. Pour une attaque boîte noire, l'attaquant ne peut réaliser que de l'inférence et n'a accès qu'au résultat de la requête. Ces attaques sont plus difficiles à réaliser mais représentent une menace plus réaliste (pour la plupart des modèles utilisés par le grand public aucune information n'est disponible, ce qui est encore plus réaliste dans le domaine de la Défense). Une taxonomie plus précise et parfois ambigüe est proposée pour parler des attaques pour lesquelles on possède une information partielle sur le modèle, on parle alors de boîte grise. Dans ce paradigme, c'est l'information sur la sortie du modèle qui varie le plus souvent. On précise donc si on se place en *hard-label attack* (on ne connait que la classe prédite par exemple) ou si on a accès à plus d'informations comme les premières classes prédites, le score de la première classe prédite, le score des premières classes prédites, le score de toutes les classes…

– Objectif de l'attaque :

  L'attaque utilisée va également dépendre de ce que l'on souhaite réaliser. Les principaux objectifs rencontrés dans la littérature sont les attaques de réduction de score de confiance, les attaques non-ciblées, les attaques ciblées et les attaques universelles.

La réduction de score de confiance consiste pour une entrée donnée à vouloir rendre équiprobables toutes les sorties du modèle (par exemple pour un modèle réalisant la tâche de classification d'images de chiffres (de 0 à 9) on veut que la sortie soit à 0.1 pour toutes les classes.

Une attaque ciblée consiste pour une entrée donnée à vouloir modifier la sortie du modèle vers un résultat en particulier tandis qu'une attaque non ciblée consiste simplement à vouloir s'éloigner le plus possible du résultat d'origine (pour un classifieur, une attaque ciblée consiste à modifier la sortie du modèle vers une classe précise tandis qu'une attaque non ciblée consiste à changer la prédiction initiale du modèle).

Une attaque universelle est un objectif plus évolué qui consiste à vouloir créer une unique perturbation permettant quelle que soit l'entrée proposée au modèle de réaliser un des objectifs décrit précédemment (par exemple pour un classifieur d'images, cela consiste à créer un patch que l'on vient poser sur n'importe quelle image en entrée du modèle et qui permet de réaliser une des attaques décrites ci-dessus).

– Spécificité/contrainte de l'attaque :

Enfin l'attaque réalisée va dépendre des contraintes pratiques que l'on rencontre ou que l'on s'impose. Le nombre de requêtes, l'imperceptibilité de l'attaque et le fait de réaliser une attaque physique sont les contraintes que l'on rencontre couramment dans la littérature mais cette liste n'est pas exhaustive et des contraintes plus spécifiques peuvent être introduite en fonction de la tâche à réaliser.

Le nombre de requêtes nécessaires permet de distinguer les attaques nécessitant beaucoup de requêtes au modèle des attaques nécessitant peu de requêtes. Cette contrainte est très présente en boîte noire pour indiquer son niveau de performance et de discrétion.

Le caractère imperceptible de l'attaque correspond à la volonté de créer une perturbation ne permettant pas de distinguer de différence (pour un observateur humain par exemple mais cela peut être aussi pour des critères plus quantitatifs) entre une donnée bénigne et une donnée avec la perturbation introduite.

Enfin le caractère physique d'une attaque correspond au fait (lorsque cela à un sens) d'introduire la perturbation dans le domaine physique (dans le monde réel). Par exemple dans le cas d'un classifieur d'images, cela consiste à imprimer la perturbation correspondant à l'attaque réalisée, la placer de manière à réaliser l'attaque dans le champ de la caméra puis prendre la photo. Par opposition, une attaque numérique correspond dans ce cas-là, à prendre directement la photo puis à ajouter numériquement la perturbation lors de la phase du traitement de l'image.

Pour réaliser une attaque, il faut donc se poser les questions suivantes :

- Qu'est-ce que je connais sur le modèle à attaquer ?
- Quel est l'objectif de mon attaque ?
- Quelles contraintes spécifiques liées au domaine d'étude et à la tâche à réaliser vont devoir être prises en compte ?

Dans la suite de cette section, on commencera par décrire les attaques boîtes blanches en introduisant les différents objectifs et certaines contraintes puis on décrira les attaques boîtes noires de la même manière. Pour finir on présentera différents mécanismes de défenses face à ces attaques.

2.4.2 *Attaques boîtes blanches :*

- *Attaques imperceptibles :*

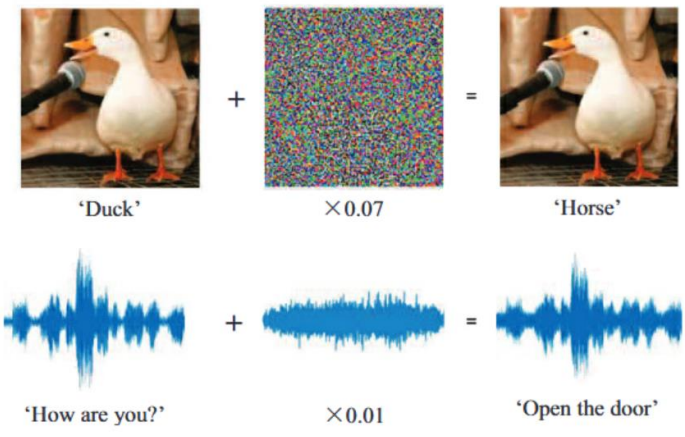


Fig. 6. L'ajout d'une perturbation imperceptible permet de modifier considérablement la prédiction du modèle, ces attaques s'appuient sur le modèle en lui-même et fonctionnent donc sur quasiment tous les types de données [adv2].

L'approche historique des attaques *adversarials* a été proposée en 2014 [adv3]. Cette approche consiste à permuter le rôle de l'entrée du modèle avec les poids du modèle pendant l'entraînement (c'est-à-dire que les poids sont fixés et c'est l'entrée que l'on modifie). Pour cela on définit une fonction de coût dont le but est de valoriser le changement de comportement du modèle tout en pénalisant un écart trop grand entre l'entrée modifiée et l'entrée originale (en mesurant par exemple la norme L2 de la différence entre les 2 entrées). Ce type d'approche par descente de gradient ne peut s'appliquer que si le modèle est différentiable (réseau de neurones par exemple).

Une des techniques servant de référence de base dans la littérature est l'attaque de *Carlini & Wagner* [adv4]. Elle consiste à introduire une nouvelle fonction de coût, introduire un changement de variable et expliquer le choix d'hyper-paramètres, le tout dans le but de rendre l'attaque plus efficace et robuste à la défense par distillation [adv5].

Les techniques les plus récentes d'attaques *adversarials* consistent à l'amélioration continue de la formulation du problème d'optimisation à résoudre pour rendre l'attaque de plus en plus efficace. [Adv8] reformule le problème pour éviter de devoir choisir un hyper-paramètre critique dans l'efficacité de l'attaque. [adv6] propose d'utiliser des *Trust Regions* [adv7] dans leur approche dans le but de rendre l'attaque plus efficace et moins visible.

Toutes les techniques évoquées pour réaliser des attaques imperceptibles (et la quasi-totalité des attaques qui seront présentées) sont réalisables dans l'optique de faire une attaque ciblée ou une attaque non-ciblée. La différence réside dans la définition de la fonction de coût utilisée. Dans le cas d'une attaque non-ciblée, on va définir une fonction de coût visant à diminuer le score associé à la sortie d'origine tandis que dans le cas d'une attaque ciblée, on définit une fonction de coût visant à augmenter le score associé à la sortie d'intérêt. A noter que peu de méthodes se consacrent à la réduction du score de confiance bien qu'il en existe comme [adv32] qui cherche la frontière de décision du modèle entre la classe d'origine et une autre classe.

- *Attaque perceptible :*

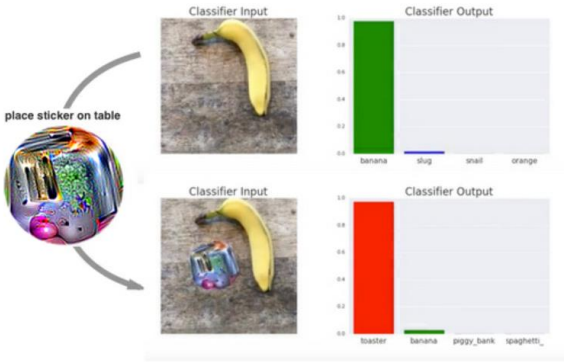


Fig. 7. Exemple d'*universal attack* : le *sticker* ajouté sur l'image du bas permet de faire passer la prédiction de classification de 97% pour la classe banane à 99% pour la classe toaster. L'avantage de ce *sticker* est qu'il a un effet sur la prédiction indépendamment de l'image d'origine.

Dans certains cas, on peut vouloir lever la contrainte de l'imperceptibilité pour atteindre d'autres objectifs. C'est le cas des *universal attacks* [adv9] où le but est de générer une perturbation perceptible qui une fois ajoutée à une donnée d'entrée (peu importe laquelle) va modifier la sortie du modèle. Cette attaque exploite la technique *expectation over transformation* qui consiste à disperser plusieurs paramètres de la donnée d'entrée (éclairage, point de vue, etc.) et à optimiser le score moyen de l'attaque. Plus généralement, cette technique est utilisée pour améliorer la robustesse des attaques *adversarials*.

- *Attaque physique :*

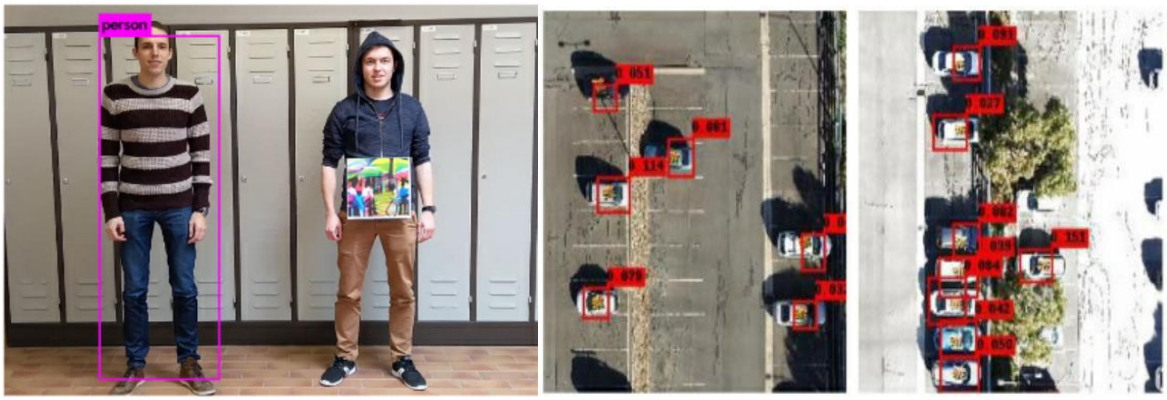


Fig. 8. Exemples d'attaques physiques basées sur le principe *d'Expectation over transformation*. A gauche, une perturbation permettant de diminuer le score de détection d'une personne [adv10] et à droite celui d'une voiture [adv11].

Dans le domaine physique, les attaques doivent résister aux passages numérique vers physique (= impression du motif) puis physique vers numérique (numérisation après capteur) difficiles à modéliser fidèlement. Rendre l'attaque robuste à ces transformations produit des motifs assez saillants mais finalement, dans cette configuration, l'attaquant cherchera davantage l'efficacité au dépend de la furtivité. Ainsi, les attaques perceptibles peuvent trouver une application dans le domaine physique. Plusieurs articles [adv10, adv11, adv12] présentent une attaque perceptible basée sur le principe d'*expectation over transformation* qui génère une perturbation qui va exister dans le monde physique et plus seulement dans le monde numérique. [adv11] présente notamment un scénario de défense face à un détecteur de véhicule en vue aérienne. Une perturbation imprimable et posée sur le toit de véhicules est générée et permet de modifier la prédiction du modèle jusqu'à faire disparaitre l'objet. Les attaques physiques diffèrent des attaques numériques dans la fonction de coût utilisée : pour les attaques physiques dans le domaine visuel par exemple des termes de régularisation sur la norme du gradient spatial ainsi qu'un score d'imprimabilité [adv12] sont ajoutés pour faciliter le passage entre les deux domaines. Ce type d'attaque n'est pas spécifique au domaine visuel mais s'applique sur n'importe quel modèle différentiable.

2.4.3 Attaques boîtes noires :

D'un point de vue opérationnel, le modèle que l'on souhaite attaquer n'est pas nécessairement connu et n'est pas nécessairement différentiable. On ne peut donc pas toujours appliquer de méthode utilisant une descente de gradient. Pour contourner ce problème plusieurs types d'attaques ont été développés.

- *Query-Based Attacks :*

La première approche consiste à requêter le modèle cible pour trouver une perturbation intéressante. L'enjeu de cette approche est d'être le plus efficace possible tout en limitant au maximum le nombre de requêtes effectuées pour pouvoir rester discret. La méthode classique [adv13] est d'approximer le gradient de la donnée en entrée en effectuant des requêtes au modèle (*zeroth-order optimization*) et ainsi se ramener dans le cas des méthodes « boîtes blanches ». D'autres méthodes plus évoluées basées sur ce principe comme [adv14] ont ensuite été développées.

Il existe d'autres approches basées sur des requêtes qui ne se reposent pas sur l'approximation du gradient mais utilisent d'autres approches, comme [adv15] qui utilise des algorithmes génétiques ou comme [adv16,adv17] qui utilisent des hypothèses sur la forme de la perturbation recherchée. [adv17] est une méthode se plaçant dans la catégorie des *hard-label attacks*.

- *Transfer-based attacks :*

La seconde approche consiste à supposer que si l'on dispose en boîte blanche d'un modèle relativement similaire au modèle d'intérêt qui lui est en boîte noire alors l'attaque sur le modèle en boîte blanche peut se transférer au modèle boîte noire. Plusieurs améliorations de ce procédé ont été proposées comme dans [adv18] où les auteurs montrent que la norme du gradient devient très faible à la fin d'une attaque, ce qui génère des perturbations proches du bruit ce qui nuit à la transférabilité de l'attaque. Ils montrent également que la

proximité dans l'espace latent d'une donnée adversaire avec sa donnée originale à un impact sur l'efficacité du transfert.

Cette approche suppose de disposer d'un modèle relativement similaire à celui étudié, ce qui n'est pas forcément le cas, des techniques permettant de copier le modèle (*model extraction*) pour en avoir un exemplaire en boîte blanche (*surrogate model*) sont présentés dans la section **Extraction/vol de modèle**.

2.4.4 Défenses :

Face à toutes ces menaces, plusieurs défenses ont été proposées. Néanmoins, il ne semble pas y avoir une défense répondant à toutes les menaces à la fois, même en les combinant.

- *Entrainement Adversarial*

La défense la plus connue et la plus efficace est l'entrainement *adversarial* [adv19]. Cela consiste à entraîner un modèle de manière classique, puis à générer à partir de ce modèle des attaques *adversarials* pour lesquelles on vient corriger l'annotation avant de les ajouter dans la base de données d'entraînement. On répète ce procédé itérativement. Beaucoup de recherches ont été faites sur ce sujet tant pour améliorer le procédé que pour apporter une preuve théorique des résultats obtenus grâce à cette technique [adv20]. De manière générale, les techniques de régularisation (ajout de bruit, pénalisation de normes, etc.) ont un effet de défense face à ces attaques *adversarials*. La défense venant modifier le modèle, on observe fréquemment une légère perte de performance.

- *defense detection*

Les méthodes dites de *defense detection* ont tendance à être moins utilisées actuellement bien qu'elles présentent deux avantages par rapport à la défense : elles informent de la présence d'une attaque et elles ne modifient pas le modèle donc conservent les performances En contrepartie, elles induisent une latence supplémentaire. Elles consistent à tenter de détecter si une entrée est une donnée adversaire ou une donnée normale. Beaucoup de méthodes ont été proposées comme [adv21] qui propose une méthode de reconstruction des données d'entrée bénignes les distinguant des données malicieuses ou encore [adv22] qui identifie les neurones activés par des entrées bénignes pour filtrer les attaques.

- *Input transformations*

Les méthodes d'*Input transformations* consistent à transformer la donnée reçue en entrée de manière à enlever la perturbation ou au moins limiter son impact sur la sortie. Plusieurs perturbations ont été envisagées : [adv23] propose d'effectuer une compression JPEG sur la donnée d'entrée, [adv24] propose un auto-encoder pour enlever la perturbation, [adv25] propose d'utiliser un GAN pour recréer une image saine proche de l'image d'entrée. Ces méthodes présentent l'avantage d'être facilement intégrables avec d'autres types de méthodes. Néanmoins elles peuvent être plus compliquées à mettre en œuvre pour des données discrètes (tableaux, TAL, etc.).

- *certified defenses*

Les méthodes dites *certified defenses* suscitent de plus en plus l'intérêt de la communauté. Elles consistent à prouver soit statistiquement soit formellement qu'on ne peut pas générer d'attaques qui provoqueraient une distorsion inférieure à un seuil fixé. [adv26] a prouvé que certains types de modèles étaient robustes aux attaques pour les normes Lp avec p>1. D'autres articles comme [adv27] étudient la vérifiabilité des modèles face aux attaques par patch. Ces méthodes ont néanmoins le désavantage d'apporter une preuve partielle puisqu'elles ne couvrent pas toutes les normes Lp et sont spécifiques à des « petits » modèles. Elles sont souvent testées sur des cas simples car elles demandent beaucoup de ressources de calcul.

2.4.5 Remarques

Les attaques *adversarials* ont suscité un intérêt croissant depuis leur apparition en 2013. La tendance est actuellement de développer des défenses plutôt que des attaques. La communauté se concentre aussi plus sur le paradigme boîte noire qui est plus complexe mais réaliste. D'un point de vue plus technique, la descente de gradient pour réaliser des attaques reste la méthode la plus fiable et la plus utilisée, les autres techniques étant plus à la marge. La majorité des travaux porte sur le domaine visuel, cela s'explique sans doute par l'utilisation de modèles déjà bien maitrisés ainsi que par l'aspect historique des attaques *adversarials* qui sont apparues

dans ce domaine. Les domaines délicats à manipuler (TAL, données hétérogènes, etc.) sont moins représentés mais commencent à être de plus en plus étudiés.

2.4.6 Niveau de risques

Stable. Un plateau semble avoir été atteint sur les techniques de défense et d'attaque. En revanche, l'emploi croissant de modèles pré-entrainés augmente le risque d'attaque par transfert et les bibliothèques dédiées réduisent les coûts de mise en œuvre.

2.4.7 Ressources

Le site de Nicholas Carlini [adv28] propose une liste des dernières publications liées de près ou de loin aux attaques *adversarials* mais aussi aux problématiques de *backdoor*, d'*inference* et de *privacy*.

[bd7] est une bibliothèque python développée par IBM qui propose une implémentation de différentes attaques et défenses. Cette bibliothèque est régulièrement mise à jour et des jupyter notebooks sont disponibles pour prendre le code en main rapidement.

RobustBenchmark [adv29] est un *benchmark* pour évaluer la robustesse de défenses proposées dans la littérature. Ce *benchmark* réalise des attaques sur la norme L1, L2 et Linf et sur les jeux de données CIFAR-10, CIFAR-100 et ImageNet.

AutoAttack [adv30] est l'attaque principalement utilisée pour réaliser *RobustBenchmark*. Il s'agit d'un regroupement de 4 attaques différentes (certaines en boîte blanche d'autres en boîte noire) avec les hyper-paramètres de ces attaques déjà initialisés pour standardiser les attaques. Le code est disponible pour tester *AutoAttack* sur son propre modèle.

Pour approfondir le domaine des attaques *adversarials* le lecteur est invité à lire [adv31].

2.5 Attaque de confidentialité/inférence

Le phénomène de sur-apprentissage montre que les modèles sont capables d'enregistrer certains détails des données d'apprentissage. Ainsi, certaines informations du jeu de données peuvent être reconstruites à partir du modèle entrainé uniquement. Une taxonomie de ces risques est présentée en Fig. 9. Considérant les risques directs comme relevant de la cybersécurité, ce rapport se concentre sur les risques indirects. Des méthodes de robustification ou de détection peuvent être mises en place spécifiquement pour un type d'attaque, ou bien des mécanismes généraux garantissant la confidentialité peuvent être proposés.

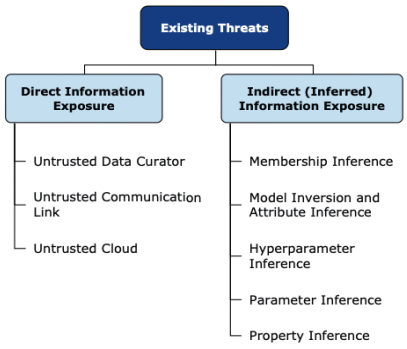


Fig. 9. Taxonomie des vulnérabilités de confidentialité, d'après [pv1].

2.5.1 Attaque d'appartenance (membership)

Cette attaque proposée en 2017 par [pv2] consiste à inférer si une donnée particulière appartient ou non au jeu d'entrainement du modèle exploré. Elle s'appuie sur l'hypothèse que si la donnée a effectivement servi à l'entrainement, alors la confiance du modèle pour cette donnée doit être plus marquée que pour une donnée hors du jeu d'entrainement. Parmi les attaques de confidentialité, l'attaque d'appartenance est la plus facile à réaliser et est donc plus largement explorée. Récemment, [pv3] montre que la métrique usuelle (et donc celle qui guide les travaux de la communauté) d'*accuracy* globale n'est pas adaptée et devrait être substituée par le

taux de vrai positif pour une faible valeur de faux positif fixée. Il propose en ce sens la méthode LiRA qui s'annonce comme l'état de l'art actuel (Fig. 10.).

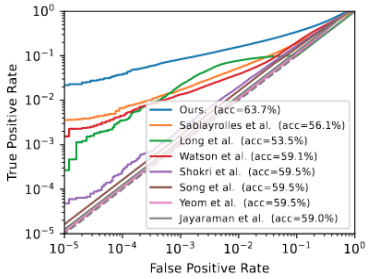


Fig. 10. Taux de vrai-positif en fonction du taux de faux-positif pour différentes méthodes d'attaque d'appartenance, d'après [pv3].

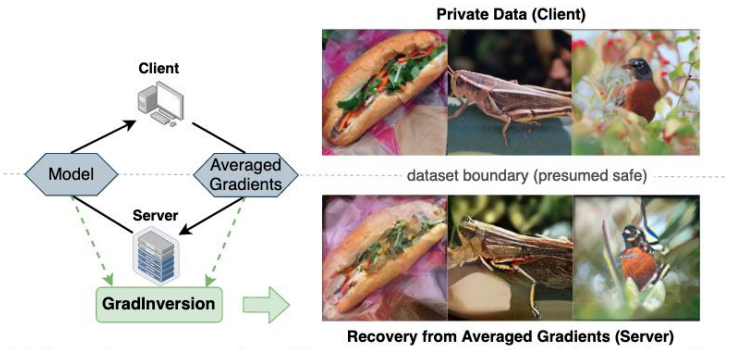
2.5.2 *Attaque de reconstruction (inversion, attribute inference)*

Cette attaque consiste à reconstruire une partie ou l'ensemble d'une donnée d'entraînement. La première illustration [pv4] portait sur des données structurées dans le domaine médical mais les mêmes auteurs [pv5] ont également reconstruit des visages entiers (Fig. 11).

Figure 1: An image recovered using a new model inversion attack (left) and a training set image of the victim (right). The attacker is given only the person's name and access to a facial recognition system that returns a class confidence score.

Fig. 11. Attaque d'inversion, d'après [pv5].

Ce risque semble particulièrement marqué dans le cadre de l'apprentissage fédéré (*federated learning)* [pv6] où les données ne sont pas directement transmises mais où les gradients le sont (il était jusqu'alors considéré que le mélange au sein d'un *batch* rendait l'inversion impossible) (Fig. 12).



(a) Inverting averaged gradients to recover original image batches

Fig. 12. Attaque d'inversion, d'après [pv6].

2.5.3 Extraction/vol de modèle

Les attaques d'inférence des (hyper)paramètres sont regroupées dans la vulnérabilité plus vaste de l'extraction/vol de modèle. Dans une démarche similaire aux techniques de distillation (*teacher-student*), il est possible de concevoir un modèle boite blanche à partir de multiples requêtes à un modèle boite noire. Cela demande néanmoins d'avoir accès au jeu d'entrainement ou à un jeu de données similaire. La communauté cherche à extraire un modèle cible avec la plus grande fidélité tout en minimisant le nombre de requêtes en utilisant par exemple des modèles de substitution pour optimiser les prochaines requêtes ou encore en utilisant des données synthétiques [pv7].

Ces requêtes sont souvent assez singulières par rapport aux données du jeu d'entrainement ou des requêtes d'utilisateurs nominaux (*ood* : *out-of-distribution*). Les défenses peuvent alors détecter cet écart à la norme et retourner des valeurs erronées pour induire en erreur l'attaquant [pv8].

2.5.4 Défense - Mécanismes généraux

Sur le schéma du cycle de vie des modèles et des vulnérabilités associées, [pv1] propose une taxonomie des schémas de défense mis en place dans la phase de préparation de données, d'entrainement et d'inférence (Fig. 13).

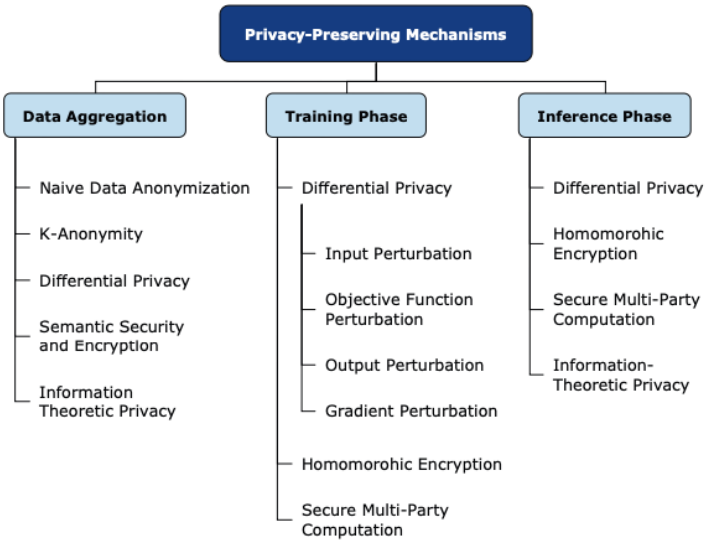


Fig. 13. Taxonomie des mécanismes de confidentialité, d'après [pv1].

- *Anonymisation naïve*

L'anonymisation naïve est à rapprocher du traitement des biais dans les jeux de données. L'idée est simplement de retirer de l'apprentissage les informations à partir desquelles le modèle ne doit pas baser son apprentissage. Ces informations vont du genre (dans le cas de la gestion des biais) aux noms ou identifiants personnels (dans le cas de la confidentialité). [pv9] établit une liste conséquente de recommandations pour la préparation de données, au-delà des considérations de confidentialité.

- *K-anonymity/Differential privacy*

Ces deux notions sont davantage des mesures que des méthodes.

La *K-anonymity* est la propriété d'un jeu de données assurant que l'on ne peut distinguer un échantillon d'au moins *k-1* autres échantillons. Cette approche a été employée principalement sur des données structurées et est rapidement limitée en grandes dimensions.

La ε-*differential privacy* est la propriété d'une paire de jeu de données (D et D'), qui ne diffèrent que d'un échantillon, assurant que le rapport des prédictions d'un même modèle entrainé sur D et sur D' ne dépasse pas exp(ε). Formulée ainsi, la ε-*differential privacy* donne une borne inférieure sur la confidentialité réelle/moyenne. Autrement dit, la *differential privacy* mesure le poids dans l'apprentissage que peut avoir une

unique donnée par rapport à l'ensemble du jeu de donnée. En pratique, un simple ajout de bruit (souvent de norme l1, « mécanisme de Laplace ») permet d'augmenter la valeur de ε. Il peut être ajouté en différents stades du processus (sur les données, dans le calcul de la fonction coût, dans les prédictions, etc.). Ce bruit dégrade logiquement les performances du modèle.

- *Chiffrement*

Le chiffrement conventionnel peut être appliqué pour la communication entre acteurs ou pour le stockage d'information mais il n'est pas compatible avec les phases d'entrainement ou d'inférence du modèle car il faut opérer des calculs sur ces données chiffrées. Le chiffrement homomorphe répond à cette contrainte en permettant un certain nombre d'opérations élémentaires (addition, soustraction, etc.) dans le domaine chiffré. Cependant, l'ensemble des opérations classiquement employées en traitement d'image, du signal, ou des langues ne sont pas encore traduites dans le domaine chiffré (les opérations non-linéaires sont particulièrement complexes). Aussi, le coût numérique des opérations dans le domaine chiffré ne permet pas actuellement le traitement complet des modèles à l'état de l'art. Un compromis consiste alors à utiliser le chiffrement homomorphe uniquement à partir d'espaces latents proches de la sortie du modèle. D'autres protocoles de chiffrement sont proposés dans la littérature, comme le calcul multipartite sécurisé (SMPC) qui permet de distribuer le calcul entre différents agents sur la base du secret réparti [pv10]. Ce partage sécurisé implique un débit d'informations échangées entre agents plus important.

2.5.5 Remarques

En phase d'entrainement, le coût du chiffrement est prohibitif et la *differential privacy* lui est préférée. En revanche en phase d'inférence, à cause de la borne inférieure sur la ε-*differential privacy,* le bruit à ajouter peut devenir important et trop dégrader les performances. L'inférence étant moins coûteuse que l'entrainement, le chiffrement est faisable en inférence et est donc préféré à la ε-*differential privacy.*

La détection de *backdoor* repose principalement sur de la détection d'anomalie non-supervisée. Le chiffrement ou la *differential privacy* complique cette détection et augmente le risque de *backdoor*.

2.5.6 Niveau de risque

Croissant dans le cadre de l'apprentissage fédéré, plutôt limité dans les autres contextes.

2.5.7 Ressources

[pv11] et [pv12] proposent une veille mise à jour fréquemment sur le sujet de la confidentialité,

[pv13] et [pv14] proposent une bibliothèque pour tester le niveau de confidentialité d'un modèle.

Pour approfondir la notion de confidentialité, le lecteur est invité à lire [pv1, pv15, pv16].

2.6 Apprentissage fédéré

A la différence des méthodes d'ensemble qui agrègent les prédictions de modèles entrainés, l'apprentissage fédéré agrège les poids de modèles (souvent par simple moyennage) en cours d'apprentissage. Cette technique est vue comme une réponse à la question de la confidentialité puisque les données d'entrainement de chaque modèle ne sont pas partagées directement.

Or, la littérature montre que ce mécanisme d'apprentissage est particulièrement sensible dans la phase d'entrainement aux attaques par *backdoors* (puisqu'elle peut être propagée aux autres modèles fédérés), aux attaques de confidentialité (la donnée fuite par le gradient partagé aux autres modèles fédérés. c.f. Fig. 12.) mais aussi dans la phase d'inférence aux attaques *adversarials* (puisque le modèle est partagé, l'attaquant a une connaissance boite blanche) (Fig. 14). Par ailleurs, les mécanismes de défense sont concentrés au niveau du serveur avant l'étape d'agrégation. Le modèle de menace distingue l'attaquant participant ou non à l'entrainement (*insider vs outsider*) en se concentrant naturellement sur le premier. L'objectif de l'attaquant peut être d'empêcher la convergence du modèle global, d'injecter des *backdoors*, ou encore d'inférer de

l'information des autres participants. Les défenses décrites pour les attaques de *backdoor* et de confidentialité sont appliquées dans le contexte fédéré.

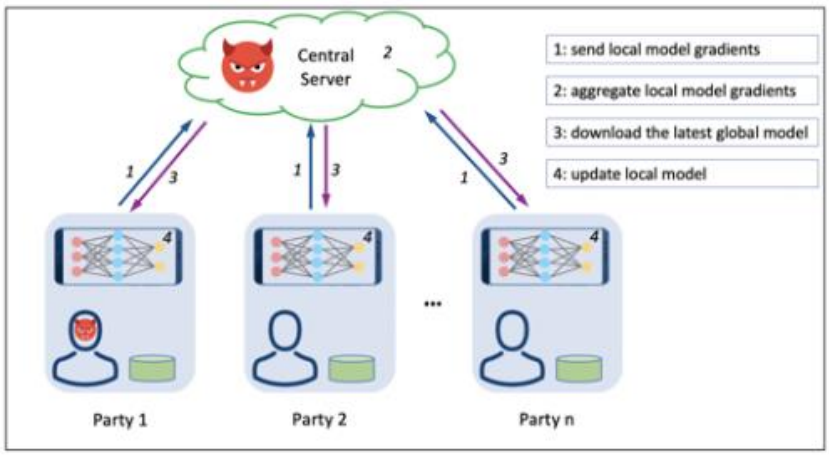


Fig. 14. Schéma de l'apprentissage fédéré, où les participants et/ou le serveur d'agrégation peuvent être malveillants, d'après [fl1].

Ressources

[An.fl1] présente une taxonomie de l'apprentissage fédéré garantissant la confidentialité.

Pour approfondir la notion d'apprentissage fédéré, le lecteur est invité à lire [fl1, fl2].

2.7 Attaque matérielle

Les sections précédentes se concentraient sur l'aspect logiciel/algorithmique des vulnérabilités. Une littérature sur les vulnérabilités matérielles existe également, bien qu'elle soit moins dense car elle demande plus d'investissement ou d'équipement. La plupart de ces attaques interviennent dans la phase d'inférence, quand le modèle est embarqué sur un matériel spécifique. [hw1] propose une première taxonomie de ces vulnérabilités (Fig. 15) tandis que [hw2] en propose une spécifique aux attaques par canaux auxiliaires (Fig. 16).

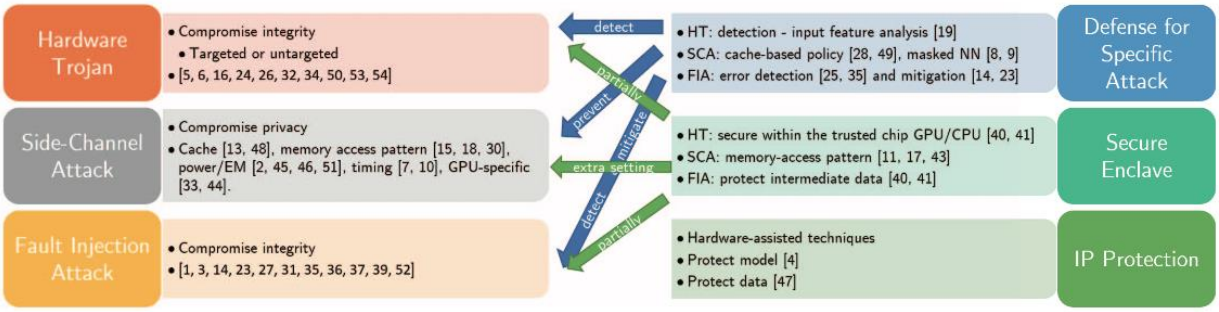


Figure 2: Hardware-related attacks and countermeasures for neural networks. The left column lists the goals for each type of attack. The right column lists how each type of countermeasure helps in improving security for neural network.

Fig. 15. Taxonomie des attaques matérielles des réseaux de neurones, d'après [hw1].

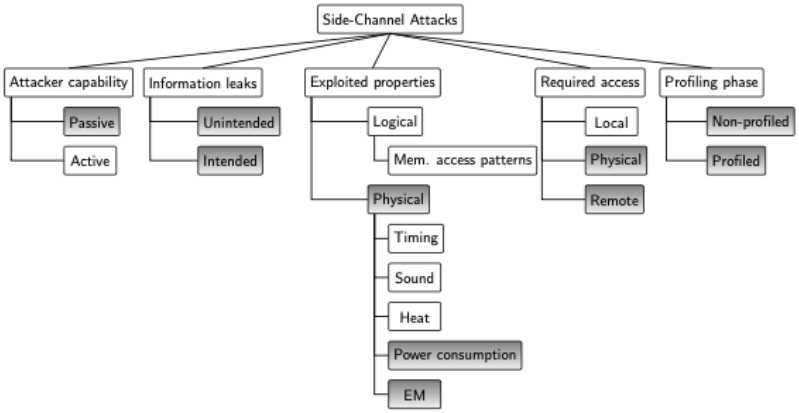


Fig. 16. Taxonomie des attaques par canaux auxiliaires, d'après [hw2].

2.7.1 Trojan matériel

L'attaquant peut modifier le circuit pour contrôler le passage du mode nominal au mode attaqué en réponse à un *trigger* [hw3]. Le nombre d'acteurs potentiels semble cependant très limité.

Les défenses proposées s'appuient sur la défense contre les *backdoor/trojan* logiciels : l'analyse statistique des anomalies dans les données d'entrée pour tenter de détecter le *trigger*.

2.7.2 Attaque par canaux auxiliaires

Les attaques par canaux auxiliaires correspondent à des attaques de confidentialité, principalement de l'extraction de modèle ou d'inférence de paramètres, à partir de mesures indirectes : consommation de courant, fuite électromagnétique, information de l'horloge ou du cache, etc. On peut imaginer une attaque en deux temps, où une phase extrait le modèle par méthode matérielle, puis une seconde où des données d'entrainement sont inférées par méthode logicielle [hw4].

Les défenses proposées sont soit spécifiques, comme des politiques de gestion de mémoire ou de cache, soit inspirées de la littérature sur la confidentialité, en utilisant par exemple le calcul multipartite sécurisé pour leurrer les attaques basées sur l'horloge.

2.7.3 L'injection de fautes

L'injection de fautes consiste à modifier l'environnement du matériel principalement pour causer des défaillances. L'utilisation de laser, de variation de tension, de collision de mémoire peuvent servir à cet effet, que ce soit pour modifier des paramètres du modèle ou le comportement de ses fonctions élémentaires.

Pour lutter contre l'inversion de bit, des neurones redondants peuvent être ajoutés ou encore des méthodes de vérification de bit. Pour réduire le risque de changement de valeur importante, la quantification, la binarisation ou le seuillage peuvent aussi être mis en place.

2.7.4 Niveau de risque

Limité. Moins d'acteurs malveillants potentiels, publications uniquement sur des preuves de concept et sur des jeux de données réduits (MNIST, cifar10).

2.7.5 Ressources

Pour approfondir la notion d'attaque matérielle, le lecteur est invité à lire [hw1, hw2].

3. DOCUMENTS DE REFERENCE

Vue d'ensemble des vulnérabilités IA :

[vuln0] Guide de recommandations pour la spécification et la qualification de systèmes intégrant de l'intelligence artificielle (IA), édidion 2. DR-Guide S-CAT n°22000. http://sysman-dga.intradef.gouv.fr/dga/?type=mail&id=238163&latest=Y&mode=mail

[vuln1] XUE, Mingfu, YUAN, Chengxiang, WU, Heyi, et al. Machine learning security: Threats, countermeasures, and evaluations. IEEE Access, 2020, vol. 8, p. 74720-74742.

[vuln2] SHAFIQUE, Muhammad, NASEER, Mahum, THEOCHARIDES, Theocharis, et al. Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead. IEEE Design & Test, 2020, vol. 37, no 2, p. 30-57.

[vuln3] QAYYUM, Adnan, QADIR, Junaid, BILAL, Muhammad, et al. Secure and robust machine learning for healthcare: A survey. IEEE Reviews in Biomedical Engineering, 2020, vol. 14, p. 156-180.

[vuln4] KUMAR, Ram Shankar Siva, BRIEN, David O., ALBERT, Kendra, et al. Failure modes in machine learning systems. arXiv preprint arXiv:1911.11034, 2019.

[vuln5] https://docs.microsoft.com/en-us/security/engineering/threat-modeling-aiml

[vuln6] https://atlas.mitre.org/

Backdooring/Poisoning :

[bd1] GU, Tianyu, DOLAN-GAVITT, Brendan, et GARG, Siddharth. Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733, 2017.

[bd2] WENGER, Emily, PASSANANTI, Josephine, BHAGOJI, Arjun Nitin, et al. Backdoor Attacks Against Deep Learning Systems in the Physical World. In : Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. p. 6206-6215.

[bd3] SHAFAHI, Ali, HUANG, W. Ronny, NAJIBI, Mahyar, et al. Poison frogs! targeted clean-label poisoning attacks on neural networks. arXiv preprint arXiv:1804.00792, 2018.

[bd3b] SAHA, Aniruddha, TEJANKAR, Ajinkya, KOOHPAYEGANI, Soroush Abbasi, et al. Backdoor Attacks on Self-Supervised Learning. arXiv preprint arXiv:2105.10123, 2021.

[bd3c] CARLINI, Nicholas et TERZIS, Andreas. Poisoning and Backdooring Contrastive Learning. arXiv preprint arXiv:2106.09667, 2021.

[bd4]WANG, Shuo, NEPAL, Surya, RUDOLPH, Carsten, et al. Backdoor attacks against transfer learning with pre-trained deep learning models. IEEE Transactions on Services Computing, 2020.

[bd5] https://github.com/THUYimingLi/backdoor-learning-ressiyrces

[bd6] https://github.com/ain-soph/trojanzoo

[bd7]https://github.com/Trusted-AI/adversarial-robustness-toolbox

[bd8] https://pages.nist.gov/trojai/docs/about.html

[bd9] GAO, Yansong, DOAN, Bao Gia, ZHANG, Zhi, et al. Backdoor attacks and countermeasures on deep learning: A comprehensive review. arXiv preprint arXiv:2007.10760, 2020.
[bd10] PANG, Ren, ZHANG, Zheng, GAO, Xiangshan, et al. TROJANZOO: Everything you ever wanted to know about neural backdoors (but were afraid to ask). arXiv preprint arXiv:2012.09302, 2020.

Adversarial attacks :

[adv1] Ilahi, I., "Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning", arXiv e-prints, 2020.

[adv2] Gong, Y. and Poellabauer, C., "Protecting Voice Controlled Systems Using Sound Source Identification Based on Acoustic Cues", arXiv e-prints, 2018.

[adv3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing

adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[adv4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in 2017 ieee symposium on security and privacy (sp). IEEE, 2017, pp. 39–57.

[adv5] PAPERNOT, N., MCDANIEL, P., WU, X., JHA, S.,ANDSWAMI, A.Distillation as a defense to adversarial perturbations against deep neuralnetworks.IEEE Symposium on Security and Privacy(2016)

[adv6] Z. Yao, A. Gholami, P. Xu, K. Keutzer, and M. W. Mahoney, "Trust region based adversarial attack on neural networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11 350–11 359

[adv7] A. R. Conn, N. I. Gould, and P. L. Toint, Trust region methods. SIAM, 2000.

[adv8] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradientbased l2 adversarial attacks and defenses," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4322–4330.

[adv9] T. B. Brown, D. Man´e, A. Roy, M. Abadi, and J. Gilmer, "Adversarial

patch," arXiv preprint arXiv:1712.09665, 2017

[adv10] Thys, S., Van Ranst, W., and Goedemé, T., "Fooling automated surveillance cameras: adversarial patches to attack person detection", arXiv e-prints, 2019.

[adv11] Du, A., "Physical Adversarial Attacks on an Aerial Imagery Object Detector", arXiv e-prints, 2021.

[adv12] Chen, S.-T., Cornelius, C., Martin, J., and Chau, D. H., "ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector", arXiv e-prints, 2018.

[adv13] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J., "ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models", arXiv e-prints, 2017.

[adv14] Cheng, S., Dong, Y., Pang, T., Su, H., and Zhu, J., "Improving Black-box Adversarial Attacks with a Transfer-based Prior", arXiv e-prints, 2019.

[adv15] Chen, J., Su, M., Shen, S., Xiong, H., and Zheng, H., "POBA-GA: Perturbation Optimized Black-Box Adversarial Attacks via Genetic Algorithm", arXiv e-prints, 2019.

[adv16] Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M., "Square Attack: a query-efficient black-box adversarial attack via random search", arXiv e-prints, 2019.

[adv17] Chen, J. and Gu, Q., "RayS: A Ray Searching Method for Hard-label Adversarial Attack", arXiv e-prints, 2020.

[adv18] Y. Shi, S. Wang, and Y. Han, "Curls & whey: Boosting black-box adversarial attacks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6519–6527.

[adv19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv preprint arXiv:1706.06083, 2017.

[adv20] Y. Zhang, O. Plevrakis, S. S. Du, X. Li, Z. Song, and S. Arora, "Overparameterized adversarial training: An analysis overcoming the curse of dimensionality," arXiv preprint arXiv:2002.06668, 2020.

[adv21] Y. Qin, N. Frosst, S. Sabour, C. Raffel, G. Cottrell, and G. Hinton, "Detecting and diagnosing adversarial images with class-conditional capsule reconstructions," ICLR, 2020.

[adv22] Y. Qiu, J. Leng, C. Guo, Q. Chen, C. Li, M. Guo, and Y. Zhu, "Adversarial defense through network profiling based path extraction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4777–4786.

[adv23] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," ICLR, 2018.

[adv24] X. Jia, X. Wei, X. Cao, and H. Foroosh, "Comdefend: An efficient image compression model to defend adversarial examples," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6084–6092.

[adv25] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," ICLR, 2018.

[adv26] F. Croce and M. Hein, "Provable robustness against all adversarial l p_perturbations

for p>_1." in ICLR, 2020.

[adv27] A. Levine and S. Feizi, "(de) randomized smoothing for certifiable defense against patch attacks," arXiv preprint arXiv:2002.10733, 2020.

[adv28] https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[adv29] https://robustbench.github.io

[adv30] https://github.com/fra31/auto-attack

[adv31] Akhtar, N., Mian, A., Kardan, N., & Shah, M. (2021). Advances in adversarial attacks and defenses in computer vision: A survey. IEEE Access, 9, 155161-155196

[adv32] Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2574-2582).

Trojaning :

[tr1] LIU, Yuntao, XIE, Yang, et SRIVASTAVA, Ankur. Neural trojans. In : 2017 IEEE International Conference on Computer Design (ICCD). IEEE, 2017. p. 45-48.

[tr2] TANG, Ruixiang, DU, Mengnan, LIU, Ninghao, et al. An embarrassingly simple approach for trojan attack in deep neural networks. In : Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020. p. 218-228.

Confidentialité :

[pv1] MIRESHGHALLAH, Fatemehsadat, TARAM, Mohammadkazem, VEPAKOMMA, Praneeth, et al. Privacy in deep learning: A survey. arXiv preprint arXiv:2004.12254, 2020.

[pv2] SHOKRI, Reza, STRONATI, Marco, SONG, Congzheng, et al. Membership inference attacks against machine learning models. In : 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017. p. 3-18.

[pv3] CARLINI, Nicholas, CHIEN, Steve, NASR, Milad, et al. Membership Inference Attacks From First Principles. arXiv preprint arXiv:2112.03570, 2021.

[pv4] FREDRIKSON, Matthew, LANTZ, Eric, JHA, Somesh, et al. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In : 23rd {USENIX} Security Symposium ({USENIX} Security 14). 2014. p. 17-32.

[pv5] FREDRIKSON, Matt, JHA, Somesh, et RISTENPART, Thomas. Model inversion attacks that exploit confidence information and basic countermeasures. In : Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015. p. 1322-1333.

[pv6] YIN, Hongxu, MALLYA, Arun, VAHDAT, Arash, et al. See through Gradients: Image Batch Recovery via GradInversion. In : Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. p. 16337-16346.

[pv7] KARIYAPPA, Sanjay, PRAKASH, Atul, et QURESHI, Moinuddin K. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In : Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. p. 13814-13823.

[pv8] KARIYAPPA, Sanjay et QURESHI, Moinuddin K. Defending against model stealing attacks with adaptive misinformation. In : Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. p. 770-778.

[pv9] GEBRU, Timnit, MORGENSTERN, Jamie, VECCHIONE, Briana, et al. Datasheets for datasets. Communications of the ACM, 2021, vol. 64, no 12, p. 86-92.

[pv10] https://sunfish-platform-documentation.readthedocs.io/en/latest/smc.html

[pv11] https://github.com/stratosphereips/awesome-ml-privacy-attacks

[pv12]https://github.com/HongshengHu/membership-inference-machine-learning-literature

[pv13] https://github.com/privacytrustlab/ml_privacy_meter

[pv14] https://github.com/trailofbits/PrivacyRaven

[pv15] JEGOROVA, Marija, KAUL, Chaitanya, MAYOR, Charlie, et al. Survey: Leakage and Privacy at Inference Time. arXiv preprint arXiv:2107.01614, 2021.

[pv16] HU, Hongsheng, SALCIC, Zoran, DOBBIE, Gillian, et al. Membership Inference Attacks on Machine Learning: A Survey. arXiv preprint arXiv:2103.07853, 2021.

Apprentissage fédéré :

[fl1] LYU, Lingjuan, YU, Han, MA, Xingjun, et al. Privacy and robustness in federated learning: Attacks and defenses. arXiv preprint arXiv:2012.06337, 2020.

[fl2] YIN, Xuefei, ZHU, Yanming, et HU, Jiankun. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR), 2021, vol. 54, no 6, p. 1-36.

Attaque matérielle :

[hw1] XU, Qian, ARAFIN, Md Tanvir, et QU, Gang. Security of Neural Networks from Hardware Perspective: A Survey and Beyond. In : 2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2021. p. 449-454.

[hd2] MÉNDEZ REAL, Maria et SALVADOR, Ruben. Physical Side-Channel Attacks on Embedded Neural Networks: A Survey. Applied Sciences, 2021, vol. 11, no 15, p. 6790.

[hw3] W. Li, J. Yu, X. Ning, P. Wang, Q. Wei, Y. Wang, and H. Yang. 2018. Hu-Fu: Hard- ware and Software Collaborative Attack Framework Against Neural Networks. In 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). 482–487.

[hw4] Q. Xu, M.T. Arafin, and G. Qu. 2020. MIDAS: Model Inversion Defenses Using anApproximate Memory System. In 2020 IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST).

ANNEXE I -

Neural Backdoor Attack	Architecture Modifiability	Trigger Optimizability	Training Controllability	Fine-tuning Survivability	Defense Adaptivity
BadNet (BN) [21]	○	○	○	○	○
Embarrassingly Simple Backdoor (ESB) [58]	●	○	○	○	○
TrojanNN (TNN) [40]	○	◐	○	○	○
Reflection Backdoor (RB) [41]	○	◐	○	○	○
Targeted Backdoor (TB) [11]	○	◐	○	○	○
Dynamic Backdoor (DB) [48]	○	●	○	○	○
Clean-Label Backdoor (CLB) [61]	○	○	●	○	○
Hidden Trigger Backdoor (HTB) [47]	○	○	●	○	○
Blind Backdoor (BB) [4]	○	○	●	○	○
Latent Backdoor (LB) [70]	○	○	○	●	○
Adversarial Backdoor Embedding (ABE) [32]	○	○	○	○	●
Input-Model Co-optimization (IMC) [45]	○	●	○	●	●

Table 1. Summary of representative neural backdoor attacks currently implemented in TROJANZOO (● – full optimization, ◐ – partial optimization, ○ – no optimization)

[An.bd1] Liste des principales attaques backdoors, d'après [bd10]

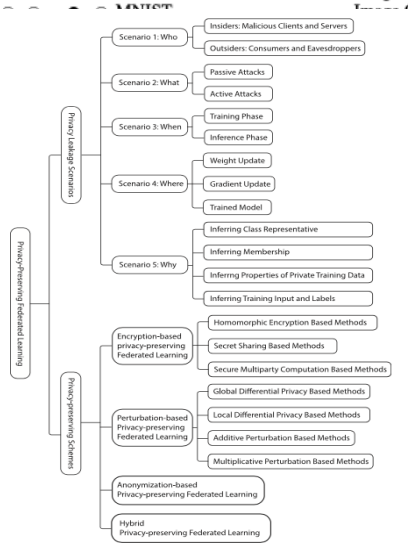
Neural Backdoor Defense	Category	Mitigation		Detection Target			Design Rationale
		Input	Model	Input	Model	Trigger	
Randomized-Smoothing (RS) [12]	Input Reformation	✓					$\mathcal{A}$'s fidelity (x's and x^*'s surrounding class boundaries)
Down-Upsampling (DU) [68]		✓					$\mathcal{A}$'s fidelity (x's and x^*'s high-level features)
Manifold-Projection (MP) [43]		✓					$\mathcal{A}$'s fidelity (x's and x^*'s manifold projections)
Activation-Clustering (AC) [9]	Input Filtering			✓			distinct activation patterns of $\{x\}$ and $\{x^*\}$
Spectral-Signature (SS) [60]				✓			distinct activation patterns of $\{x\}$ and $\{x^*\}$ (spectral space)
STRIP (STRIP) [19]				✓			distinct self-entropy of x's and x^*'s mixtures with clean inputs
NEO (NEO) [62]				✓			sensitivity of f^*'s prediction to trigger perturbation
Februus (FEBRUUS) [15]				✓		✓	sensitivity of f^*'s prediction to trigger perturbation
Adversarial-Retraining (AR) [42]	Model Sanitization		✓				$\mathcal{A}$'s fidelity (x's and x^*'s surrounding class boundaries)
Fine-Pruning (FP) [37]			✓				$\mathcal{A}$'s use of neurons rarely activated by clean inputs
NeuralCleanse (NC) [64]	Model Inpsection				✓	✓	abnormally small perturbation from other classes to t in f
DeepInspect (DI) [10]					✓	✓	abnormally small perturbation from other classes to t in f^*
TABOR (TABOR) [23]					✓	✓	abnormally small perturbation from other classes to t in f
NeuronInspect (NI) [26]					✓		distinct explanations of f and f^* with respect to clean inputs
ABS (ABS) [39]					✓	✓	$\mathcal{A}$'s use of neurons elevating t's prediction

Table 2. Summary of representative neural backdoor defenses currently implemented in TROJANZOO ($\mathcal{A}$ – backdoor attack, x – clean input, x^* – trigger input, f – benign model, f^* – trojan model, t – target class)

[An.bd2] Liste des principales défenses backdoors, d'après [bd10]

[An.pv2] Liste des principaux mécanismes préservant la confidentialité en phase d'entrainement, d'après [pv1]

Method	DP	SMC	HE	IT	Dataset(s)	Task
ARDEN [128]	●	○	○	○	MNIST, CIFAR-10, SVHN	Image Classification w/ DNN
Cryptonets [129]	○	○	●	○	MNIST	Image Classification w/ DNN
Private Classification [130]						Classification w/ DNN
TAPAS [131]						Classification w/ DNN
FHE–DiNN [132]						Classification w/ DNN
Face Match [133]						cognition with CNNs
Cheetah [134]						Classification w/ DNN
EPIC [119]						Classification w/ DNN
DeepSecure [135]						cation w/ DNN
XONN [118]						Classification w/ DNN
Chameleon [136]						cation w/ DNN and SVM
CRYPTFLOW [137]						cation w/ DNN
Classification over Encrypted Data[138						cation w/ NB, DT
MiniONN [139]						Classification w/ DNN
GAZELLE [140]						Classification w/ DNN
DELPHI [141]						Classification w/ DNN
Shredder [31]						cation w/ DNN
Sensor Data Obfuscation [142]						Recognition w/ DNN
Olympus [143]						Recognition w/ DNN
DPFE [29]						Classification w/ DNN
Cloak [144]						Classification w/ DNN
DSSGD [25]	●					ge Classification w/ DNN
Private Collaborative NN [106]	●					ge Classification w/ DNN
Secure Aggregation for ML [107]	○					rated Learning
QUOTIENT [108]	○					sification w/ DNN
SecureNN [109]	○					ge Classification w/ DNN
ABY3 [110]	○					LOR, NN
Trident [111]	○					LOR, NN
SecureML [112]	○	●	●		MNIST, Gisette, Arcene	LIR, LOR, NN
Deep Learning w/ AHE [113]	○	○	●		MNIST	Image Classification w/ DNN
ML Confidential [114]	○	○	●		Wisconsin Breast Cancer	LM, FLD
Encrypted Statistical ML [115]	○	○	●		20 datasets from UCI ML	LOR, NB, RF
CryptoDL [27]	○	○	●		MNIST, CIFAR-10	Image Classification w/ DNN
DPHE [116]	○	○	●		Caltech101/256, CelebA	Image Classification w/ SVM



[An.pv3] Liste des principaux mécanismes préservant la confidentialité en phase d'inférence, d'après [pv1]

[An.fl1] Taxonomie de l'apprentissage fédéré garantissant la confidentialité, d'après [fl2].

Reconnaissance	Initial Access	Execution	Persistence	Model Evasion	Exfiltration	Impact
Acquire OSINT information: (Sub Techniques) 1. Arxiv 2. Public blogs 3. Press Releases 4. Conference Proceedings 5. Github Repository 6. Tweets	Pre-trained ML model with backdoor	Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding	Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding	Evasion Attack (Sub Techniques) 1. Offline Evasion 2. Online Evasion	Exfiltrate Training Data (Sub Techniques) 1. Membership inference attack 2. Model inversion	Defacement
ML Model Discovery (Sub Techniques) 1. Reveal ML model ontology – 2. Reveal ML model family –	Valid account	Execution via API	Account Manipulation		Model Stealing	Denial of Service
Gathering datasets	Phishing	Traditional Software attacks	Implant Container Image	Model Poisoning	Insecure Storage 1. Model File 2. Training data	Stolen Intellectual Property
Exploit physcial environment	External remote services			Data Poisoning (Sub Techniques) 1. Tainting data from acquisition – Label corruption 2. Tainting data from open source supply chains 3. Tainting data from acquisition – Chaff data 4. Tainting data in training environment – Label corruption		Data Encrypted for Impact Defacement
Model Replication (Sub Techniques) 1. Exploit API – Shadow Model 2. Alter publicly available, pre-trained weights	Exploit public facing application					Stop System Shutdown/Reboot
Model Stealing	Trusted Relationship					

[An.vuln1] Matrice des menaces des systèmes d'apprentissage automatique, d'après [Vuln6].