Tatouage robuste et sûr de réseaux de neurones en boîte noire

Kassem KALLAS Teddy FURON
Centre Inria de l'Université de Rennes

Résumé – Protéger la propriété intellectuelle des réseaux de neurones fait sens du fait de leur valeur industrielle. Cet article propose un protocole de tatouage léger, robuste et sûr pour les modèles en boîte noire. Il utilise une fonction de hachage à sens unique et des paires donnée / classe pour prouver la propriété du modèle en phase de test. La principale caractéristique est que la valeur de la preuve et sa sécurité sont mesurables. Les expériences montrent une protection adéquate contre diverses attaques.

Abstract – Protecting the intellectual property of neural networks makes sense because of their industrial value. This paper proposes a lightweight, robust and secure watermarking protocol for black box models. It uses a one-way hash function and data/class pairs to prove model ownership in the test phase. The main feature is that the value of the proof and its security are measurable. Experiments show adequate protection against various attacks.

1 Introduction

Les réseaux de neurones profonds (DNN) sont largement utilisés dans les applications critiques, mais leur entraînement est coûteux en données d'apprentissage de qualité et en temps de calcul. La protection de la propriété des DNN devient alors essentielle, car ils sont considérés comme des actifs industriels précieux [7]. La protection de la propriété intellectuelle nécessite une preuve de propriété d'un modèle déployé. Le tatouage numérique de DNN est une méthode robuste qui demande autant de ressources pour être supprimé que pour entraîner un nouveau réseau. Cela en fait l'outil de protection principal.

Cet article ne vise pas la robustesse du tatouage numérique de DNN, mais plutôt la *valeur de la preuve de propriété*, mesurée par la probabilité p qu'une clé secrète choisie au hasard produise un score plus élevé que celui produit par le prétendu propriétaire. Nous proposons un protocole pour valider le tatouage en éliminant les doutes du vérificateur. En effet, ce dernier part du principe que le prétendu propriétaire est un Usurpateur et il mesure le niveau de sécurité en quantifiant la quantité de *travail* nécessaire à un Usurpateur pour trouver une pseudo clé secrète.

Enfin, le mécanisme proposé accorde la propriété uniquement si le *travail* et la *valeur de la preuve* sont élevés. La valeur de la preuve est linéaire avec la taille de la clé et le travail exponentiel avec la valeur de la preuve, ce qui rend la recherche d'une pseudo clé secrète pseudo a posteriori NP difficile avec la taille de la clé. Ce schéma préserve la robustesse du tatouage numérique, comme le montrent les tests expérimentaux effectués sur plusieurs ensembles de données.

2 État de l'art

Le premier tatouage de réseau de neurones est proposé en 2017 [6] dans un cadre "boîte blanche" où le signal de tatouage est directement ajouté aux poids du DNN. Cette approche est encore d'actualité [5]. Dans le cadre "boîte noire", le tatouage modifie le comportement du DNN. Un exemple de cette technique est le "backdooring", qui force le DNN à sur-apprendre sur un ensemble inhabituel de paires image-classe, appelées en anglais "*backdoors*", "*triggers*", "*watermarked samples*" ou "*secret keys*" [1]. Dans cet article, nous traduirons pas *marqeurs*. Les images de ces marqueurs sont artificielles [1], des exemples adversaires [4], ou des images naturelles avec une superposition d'un signal visible [8] ou invisible [3, 9].

Les trois attaques les plus utilisées pour évaluer la robustesse du tatouage de classifieur sont le transfert de connaissances, l'élagage de modèle et la quantification des poids. Le transfert de connaissances implique le remplacement des couches entièrement connectées par de nouvelles couches adaptées à la tâche, comme utiliser un DNN pré-entraîné sur ImageNet pour classifier CIFAR10. La quantification de poids réduit la taille mémoire du réseau. Une attaque réussie élimine la présence du tatouage sans affecter la précision du modèle sur la tâche principale, sinon l'attaquant obtient un modèle inutile.

3 Méthode proposée

Le tatouage est une vieille technologie née à la fin des années 90 protégeant les contenus multimédia. Cette communauté a déjà théorisé la valeur de la preuve au sens de ce que signifie le fait de détecter la présence d'un tatouage. Notre méthode n'est que la mise en application de ces vieux travaux [2] au tatouage de DNN en boîte noire. A savoir, i) le Propriétaire ne peut pas sélectionner arbitrairement un signal de tatouage, ii) le Vérificateur doit être convaincu que la clé secrète n'a pas été générée après coup. Deux mesures sont proposées afin de convaincre le Vérificateur : la valeur de la preuve du Propriétaire quantifié par la rareté et le niveau de sécurité donné par la quantité travail nécessaire pour usurper le Propriétaire.

Avant tout, nous élaborons un modèle de l'attaquant. Nous supposons qu'il peut facilement faire des exemples adverses ciblés, c'est à dire modifier une donnée avec une perturbation invisible et indétectable pour que la sortie d'un DNN soit la classe ciblée. Le coût d'une inférence du réseau est noté ω_F, tandis que le coût d'un calcul de gradient par rétropropagation est évalué à $2\omega_F$. Les attaques adverses en boîte blanche impliquent généralement une descente de gradient sur $t \approx 100$ itérations, ce qui équivaut à un coût de $2t\omega_F$ pour la fabrication d'un exemple adversaire.

Nous présentons notre méthode progressivement en augmentant étape par étape le niveau de sécurité. La première

étape introduit la construction de base et les définitions de rareté et de travail.

3.1 Niveau 0 : Génération pseudo-aléatoire

Considérons un ensemble d'entraînement $\mathcal{D}_{train}$ qui comprend n paires d'entrées / classes $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathcal{C}$, où $\mathcal{C} = \{0, 1, \ldots, c-1\}$ est l'ensemble des classes.

Avant l'apprentissage, un petit ensemble de s données d'entraînement est sélectionné de manière secrète comme marqueurs. Par convention, disons que ce sont les premiers échantillons d'entraînement $\{x_i\}_{i=1}^s$. Leur étiquetage initial est remplacé par s classes $\{\tilde{y}_i\}_{i=1}^s$ tirées au hasard dans $\mathcal{C}$. Cela génère un ensemble $\mathcal{S} = \{(x_i, \tilde{y}_i)\}_{i=1}^s$ de paires inhabituelles appelées marqueurs.

Pour assurer une bonne précision du modèle, les marqueurs représentent une petite fraction de l'ensemble d'entraînement : $s/n \ll 1$. Après cela, l'apprentissage est effectué comme d'habitude et il est supposé que le modèle entraîné F de grande capacité car sur-paramétré conserve la mémoire de presque tous les marqueurs.

Le Propriétaire envoie les marqueurs au Vérificateur qui soumet les images au modèle B dans la boîte noire. La détection de tatouage consiste à vérifier que $\tilde{y}_i = B(x_i)$ pour la plupart des paires de $\mathcal{S}$ (disons au moins m correspondances). Cela est sensé prouver que le modèle B est une copie du modèle tatoué F appartenant au Propriétaire. Ce dernier a pu prédire à l'avance comment le modèle B allait se tromper sur ces entrées, alors que d'autres modèles n'auraient pas fait autant d'erreurs de classification. On voit que cet argument est très suggestif et surtout erroné : si la précision du modèle n'est pas parfaite, il existe de nombreuses entrées mal classées. Usurper la propriété revient donc à trouver ces entrées mal classées.

Pour empêcher ceci, il faut au minimum que le Propriétaire n'est pas le choix dans les classes $\{\tilde{y}_i\}_{i=1}^s$. Par exemple, le Propriétaire choisit une clé secrète sk qui sert de graine à un générateur pseudo-aléatoire de classes dans $\mathcal{C}$: $\tilde{y}_i = G(sk, i)$. A la vérification, le Propriétaire envoie les entrées et la clé.

Rareté Nous suggérons de mesurer le côté inhabituel des correspondances par la rareté R, définie comme une fonction logarithmique de la probabilité d'observer au moins m correspondances $\tilde{y}_i = B(x_i)$. Lorsque l'on sélectionne une clé aléatoirement, le nombre de correspondances est une variable aléatoire M, et on mesure la rareté par :

$$R := -\log_2(\mathbb{P}(M \geq m)) \quad \text{en bits.} \tag{1}$$

Si R vaut 50 bits, cela implique qu'il est extrêmement improbable de trouver par hasard une clé secrète qui produise autant de correspondances, avec une probabilité de 2^{-50}. Ceci fournit une preuve solide de la validité de la propriété revendiquée.

Lorsque l'on choisit une clé au hasard, l'événement $\tilde{y}_i = B(x_i)$ se produit avec une probabilité de $1/c$. Par conséquent, le nombre de correspondances M suit une distribution binomiale $M \sim \mathcal{B}(s, 1/c)$. La probabilité exacte de cette distribution est donnée par :

$$\mathbb{P}(M \geq m) = I_{1/c}(m, s+1-m), \tag{2}$$

où $I_x(a, b)$ est la fonction bêta incomplète régularisée. Une expression plus commode est l'inégalité de Hoeffding :

$$\mathbb{P}(M \geq m) \lesssim e^{-2s(r-1/c)^2}, \tag{3}$$

où $r := m/s$ est le taux de récupération de la marque (voir la section 4). Si le Vérificateur accepte la preuve de propriété lorsque la rareté est supérieure à R bits, alors le nombre d'images à soumettre est :

$$s \gtrsim R\frac{\log(2)}{2(r-1/c)^2}. \tag{4}$$

Il s'agit d'une fonction décroissante du taux de récupération r.

Travail En ce qui concerne la sécurité, un Usurpateur peut sélectionner s entrées $\{x'_i\}_{i=1}^s$ et choisir une clé pour générer s classes aléatoires, puis les faire correspondre en construisant des exemples adverses. Il modifie chaque entrée $\tilde{x}_i = \mathsf{Attack}(x'_i, B, \tilde{y}_i)$ tel que $\tilde{y}_i = B(\tilde{x}_i)$ pour revendiquer la propriété du modèle B.

Le travail nécessaire pour cette attaque est de l'ordre de $W = 2t\omega_F s$. Le travail requis est relativement faible car il est seulement linéaire en fonction de la rareté (4).

3.2 Niveau 1 : Lien unidirectionnel

Le niveau suivant ajoute une contrainte supplémentaire à la génération des classes des marqueurs en utilisant une fonction de hachage cryptographique H paramétrée par la clé secrète du propriétaire. On impose que $\tilde{y}_i = H(x_i; sk)\%c$, où $\%$ est l'opérateur modulo. Le Vérificateur accorde la propriété du modèle B si $B(x_i) = H(x_i; sk)\%c$ pour au moins m entrées de $\mathcal{S}$. L'analyse de la rareté est la même que pour le Niveau 0 car les classes générées sont uniformément distribuées sur $\mathcal{C}$.

Ce nouveau schéma augmente le niveau de sécurité, car il est peu probable qu'un Usurpateur puisse créer un exemple adverse qui satisfait la règle $B(\tilde{x}) = H(\tilde{x}; sk')\%c$, où sk' est la clé secrète de l'Usurpateur. Une première attaque est de créer plus d'exemples adverses car statistiquement $1/c$ respecteront la règle. Le travail moyen requis pour trouver s marqueurs est $W = 2t\omega_F.sc$. Une autre possibilité est de créer d'abord un exemple adverse avec une classe cible y puis d'inverser le bit de poids faible d'un pixel aléatoire jusqu'à ce que $H(\tilde{x}; sk')\%c = y$. Cette micro-perturbation ne va pas changer la prédiction du classifieur mais change complètement le hash. Cette attaque nécessite en moyenne c pixels modifiés par marqueur et donc autant de calculs de hachage. Cela réduit la quantité de travail à $W = (2t\omega_F + c\omega_H).s$ avec ω_H le coût de la fonction de hachage. Le travail reste toujours proportionnel à la rareté.

3.3 Niveau 2 : Chainage

Notre dernière proposition est de hacher ensemble les s entrées marqueurs pour générer leurs classes. Pour ce faire, on calcule d'abord $h = H(x_1||\ldots||x_s; sk)$ et on utilise cette valeur comme graine pour un générateur pseudo-aléatoire produisant des entiers dans $\{0, \ldots, 2^b - 1\}$, qui sont ensuite transformés en classes par une opération de modulo. Comme dans le schéma précédent, en choisissant aléatoirement une clé secrète sk, la fonction de hachage $H(\cdot; sk)$ produit des classes uniformément distribuées dans $\mathcal{C}$, de sorte que l'analyse de la rareté R reste la même.

En termes de sécurité, l'Usurpateur commence par préparer s exemples adversaires avec des classes cibles aléatoires, modifie quelques bits de poids faible (LSB), calcule le hachage et

les classes induites, et répète jusqu'à ce qu'au moins m correspondances soient observées. Le fait de modifier un seul bit de poids faible (LSB) d'un des marqueurs modifie la classes de tous les marqueurs. En moyenne, le travail requis est évalué à

$$W = 2st\omega_F + \frac{s\omega_H}{\mathbb{P}(M \geq m)} = s(2t\omega_F + 2^R\omega_H). \quad (5)$$

Le travail est maintenant beaucoup plus grand et surtout une fonction exponentielle de la rareté.

4 Résultats expérimentaux

La méthode est évaluée sur les bases de données suivantes : MNIST avec LeNet5, Fashion MNIST avec un réseau de neurones convolutifs simple, GTSRB avec ResNet18, CIFAR10 avec VGG19, et l'apprentissage par transfert d'ImageNet à CIFAR10 avec ResNet50. Les données d'entraînement sont divisées en 80% pour l'entraînement, 10% pour la validation et 10% pour le fine-tuning, avec un nombre de marqueurs de $s = 40$ ou $s = 128$. Tous les modèles sont entraînés pendant 200 epochs, avec Adam ($lr = 0.001$) pour MNIST et FashionMNIST, et SGD ($lr = 0.001$, momentum de 0.9 et amortissement (weight decay) de $1e^{-4}$) pour CIFAR10 et GTSRB. Les augmentations de données aléatoires, à savoir le retournement horizontal et la rotation de 10 degrés, sont appliquées pour CIFAR10 en apprentissage par transfert.

Les attaques considérées sont l'ajustement fin (fine-tuning), l'élagage, la quantification des poids, la compression JPEG et le bruit Gaussien. Nous élaguons aléatoirement des neurones du DNN avec un taux $k \in (0, 1)$. La quantification des poids est effectuée de quatre manières différentes : la quantification dynamique (Dyn. Q.), la quantification entière complète (Int8 Q.), la quantification entière non signée complète (Uint8 Q.) et la conversion au format Float16 (Float16 Q.). Ces opérations réduisent la taille du DNN et accélèrent le temps de requête. JPEG50 compresse et décompresse l'image avec un QF 50 avant de la transmettre au DNN. Le bruit Gaussien $\mathcal{N}(0, \sigma^2)$ ajoute du bruit de variance $\sigma^2 \in \{0.1, 0.5, 0.9, 1.3, 1.7, 2.0\}$ aux échantillons d'entrée. Le réglage fin (fine-tuning) utilise le même algorithme que l'entraînement mais divise le taux d'apprentissage par 10, pour 15 epochs. La robustesse contre les attaques est mesurée en utilisant la précision du modèle **TA** et le taux de récupération du tatouage **REC**.

4.1 Performance de référence

Premièrement, le tatouage entraîne une légère baisse de la précision. Par exemple, sur MNIST, la perte est de 0.12 pp avec $s = 40$, et de 0,11 pp avec $s = 128$. La même conclusion s'applique aux autres données, avec une baisse maximale de 1.37 pp dans le cas de GSTRB avec $s = 40$ dans le Tab. 1.

Deuxièmement, la méthode récupère correctement les classes des marqueurs. La récupération pour MNIST est de 97.5% pour tous les s, correspondant à une rareté $R = 124$ bits selon (1). Pour Fashion MNIST, CIFAR10, GTSRB et l'apprentissage par transfert, le taux de récupération est de 100% avec une rareté R allant de 132 à 694 bits. Même avec seulement $s = 40$ marqueurs, ces valeurs de rareté sont extrêmement convaincantes pour le Vérificateur. Il convient de noter que la rareté est beaucoup plus grande lorsque le nombre de classes augmente, comme dans GTSRB où $c = 43$, renforçant la valeur de la preuve pour le Vérificateur.

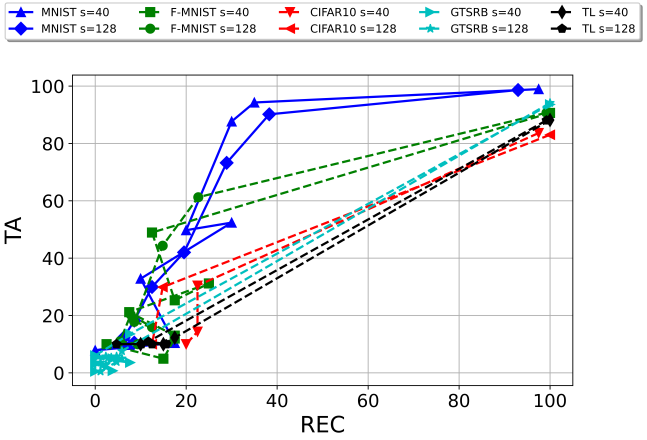

FIGURE 1 : **TA** vs. **REC** pour l'attaque par élagage

4.2 Attaques sur le modèle

L'attaque de fine-tuning affecte différemment les performances avec un effet négligeable sur MNIST et CIFAR10, mais une réduction significative du **TA** et du **REC** sur GTSRB et FashionMNIST. Le transfert d'apprentissage présente un comportement similaire mais moins sévère. La diminution du **REC** a un coût important sur la performance de la tâche principale évaluée par le **TA**, rendant le DNN inutile pour l'attaquant. Les quatre cas de quantification n'ont pas d'effet grave sur les DNN considérés, avec une perte maximale de **TA** de 1.36% pour GTSRB et une perte maximale de **REC** de 2.5% pour CIFAR10 avec $s = 40$.

Enfin, l'élagage est effectué en choisissant aléatoirement des poids de chaque couche avec un taux $k \in (0, 1)$. La performance moyenne de 50 rounds d'élagage est rapportée dans la Fig.1, où l'on observe que le **TA** et le **REC** se dégradent avec l'augmentation de k. C'est un comportement souhaité car le DNN perd de son utilité autant qu'il perd son tatouage. Plus précisément, supposons que le Vérificateur accorde la propriété si la rareté R est supérieure à 20 bits. Cela signifie qu'une clé aléatoire produit une preuve de propriété acceptée avec une probabilité inférieure à $1e^{-6}$. En termes de travail, l'Usurpateur a besoin de dizaines de millions de calculs de hachage pour forger un ensemble de marqueurs suffisamment convaincant. Selon (2), $R \geq 20$ implique un taux de récupération supérieur à 38% pour $s = 40$ ou 25% pour $s = 128$. La précision **TA** s'effondre justement dans cette zone.

4.3 Attaques sur les images

Le Tableau 2 montre les résultats des attaques sur les images d'entrée. Les pires résultats sont observés pour FashionMNIST avec une diminution de 45% ($s = 40$) du **REC** et une légère baisse du **TA** pour JPEG50. L'ajout de bruit Gaussien détériore à la fois le **TA** et le **REC** sauf pour MNIST. Notons que le **REC** ne chute jamais sans une perte substantielle de performance de la tâche principale (**TA**).

Le transfert d'apprentissage est plus robuste en termes de **TA** et de **REC** qu'un apprentissage à partir de zéro sur CIFAR10 car le modèle pré-entraîné a déjà appris des caractéristiques plus utiles pour la tâche à accomplir. En intégrant le tatouage lors du transfert d'apprentissage, un ensemble différent de marqueurs pourrait être utilisé pour décliner une version du DNN

TABLE 1 : **TA** et **REC** contre les attaques au niveau du modèle, et l'intervalle correspondant pour la rareté R

Dataset \ s		Host DNN		WM DNN		Fine-Tuning		Dyn. Q.		Int8 Q.		Uint8 Q.		Float16 Q.		R
		TA	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	bits
MNIST	40	99.04	15.0	98.92	97.5	98.92	97.5	98.92	97.5	98.92	97.5	98.92	97.5	98.92	97.5	124
	128	98.73	9.37	98.62	92.97	98.62	92.97	98.67	93.75	98.67	93.75	98.67	93.75	98.62	92.97	352-359
F-MNIST	40	90.55	17.5	90.64	100	64.27	20.0	90.66	100	90.66	100	90.66	100	90.63	100	4.5-132
	128	90.58	8.59	90.69	99.21	74.78	12.5	90.72	99.21	90.72	99.21	90.72	99.21	90.69	99.21	2.25-414
CIFAR10	40	83.56	5.0	83.01	100	83.62	100	83.6	97.5	83.6	100	83.6	100	83.62	97.5	124–132
	128	83.82	9.37	83.62	95.0	83.01	99.22	82.95	99.22	82.95	98.44	82.95	98.44	82.99	99.22	370–415
GTSRB	40	94.95	3.84	93.58	100	84.94	3.84	93.59	100	93.59	100	93.59	100	93.58	100	1.37-217
	128	94.0	2.35	94.10	100	86.65	8.23	94.09	100	94.09	100	94.09	100	94.1	100	11.2-694
TL	40	88.6	7.5	88.14	100	77.16	30.0	88.19	100	88.19	100	88.19	100	88.15	100	11.3–132
	128	88.14	6.25	88.04	100	76.12	33.59	88.07	99.21	88.07	100	88.07	100	88.02	100	31.6-425

TABLE 2 : **TA** et **REC** contre les attaques au niveau de l'image, et l'intervalle correspondant pour la rareté R

Dataset \ s		JPEG50		$\mathcal{N}(0, 0.1)$		$\mathcal{N}(0, 0.5)$		$\mathcal{N}(0, 0.9)$		$\mathcal{N}(0, 1.3)$		$\mathcal{N}(0, 1.7)$		$\mathcal{N}(0, 2.0)$		R
		TA	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	**TA**	**REC**	bits
MNIST	40	98.9	97.5	98.65	72.5	98.67	72.5	98.67	72.5	98.68	72.5	98.67	72.5	98.66	72.5	66.8-124
	128	98.64	89.84	97.73	76.56	97.76	76.56	97.75	76.56	97.75	76.56	97.77	76.56	97.8	76.56	233-326
F-MNIST	40	88.1	55.0	48.17	17.5	48.75	17.5	49.1	17.5	49.43	17.5	49.56	17.5	49.57	17.5	3.3-38.9
	128	87.45	57.03	43.63	14.06	43.84	14.06	43.76	14.06	43.66	14.06	43.73	14.06	43.83	14.06	3.5-128
CIFAR10	40	80.53	90.0	18.79	5.0	18.77	5.0	18.7	5.0	18.67	5.0	18.65	5.0	18.62	5.0	0.12-103
	128	79.71	80.46	20.03	8.59	20.0	8.59	19.9	8.59	19.94	8.59	19.92	8.59	19.92	8.59	0.42-258
GTSRB	40	92.44	96.15	19.45	3.84	14.40	3.84	12.29	3.84	10.96	3.84	10.11	3.84	9.45	3.84	1.37-201
	128	93.11	87.05	17.04	3.52	12.31	3.52	10.48	3.52	9.44	3.52	8.66	3.52	8.25	3.52	1.98-537
TL	40	83.64	72.5	10.01	10.0	10.01	10.0	10.01	10.0	10.01	10.0	10.01	10.0	10.01	10.0	0.79-66.8
	128	83.6	71.09	12.52	10.15	12.44	10.15	12.48	10.15	12.48	10.15	12.5	10.15	12.49	10.15	0.94-200

par utilisateur. Cependant, pour les modèles plus petits, nous avons constaté que l'injection avec le transfert d'apprentissage entraîne une perte notable de performance.

Suite à l'analyse des différentes attaques, nous pouvons conclure que l'algorithme proposé est robuste car toute perte du tatouage s'accompagne d'une dégradation des performances sur la tâche d'origine. La méthode de protection du modèle nécessite le hachage des images pour calculer les classes pendant l'entraînement et la vérification sur un petit nombre d'images marqueurs, ce qui est assez léger en temps de calcul.

5 Conclusion

Cet article décrit un nouveau protocole de tatouage de réseaux de neurones classifieur en boîte noire. Il est caractérisé par sa non-interférence avec les performances de la tâche originale, sa capacité à quantifier la valeur de la preuve de propriété et du niveau de sécurité, et sa robustesse contre une large gamme d'attaques. Les futures recherches porteront sur la sécurité contre des attaques plus complexes et sur une étude théorique pour comprendre les limites du tatouage en lien avec la capacité d'un modèle et la dimension des données d'entrée.

6 Remerciements

Nous souhaitons remercier les agences françaises ANR et AID pour le financement de la Chaire SAIDA ANR-20-CHIA-0011.

Références

[1] Yossi ADI, Carsten BAUM, Moustapha CISSE, Benny PINKAS et Joseph KESHET : Turning your weakness into a strength : Watermarking deep neural networks by backdooring. *In USENIX Conference on Security Symposium*, 2018.

[2] Scott CRAVER, Nasir MEMON, B-L YEO et Minerva M YEUNG : Resolving rightful ownerships with invisible watermarking techniques : Limitations, attacks, and implications. *IEEE Journal on Selected areas in Communications*, 16(4):573–586, 1998.

[3] Jia GUO et Miodrag POTKONJAK : Watermarking deep neural networks for embedded systems. *In Int. Conf. on Computer-Aided Design*, 2018.

[4] Erwan LE MERRER, Patrick PEREZ et Gilles TRÉDAN : Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32.13: 9233–9244, July 2022.

[5] Yue LI, Benedetta TONDI et Mauro BARNI : Spread-transform dither modulation watermarking of deep neural network. *Journal of Information Security and Applications*, 63, December 2021.

[6] Yusuke UCHIDA, Yuki NAGAI, Shigeyuki SAKAZAWA et Shin'ichi SATOH : Embedding watermarks into deep neural networks. *In ACM Int. Conf. on Multimedia Retrieval*, 2017.

[7] Yingchun WANG, Jingyi WANG, Weizhan ZHANG, Yufeng ZHAN, Song GUO, Qinghua ZHENG et Xuanyu WANG : A survey on deploying mobile deep learning applications : A systemic and technical perspective. *Digital Communications and Networks*, 8 (1):1–17, 2022.

[8] Jialong ZHANG, Zhongshu GU, Jiyong JANG, Hui WU, Marc Ph STOECKLIN, Heqing HUANG et Ian MOLLOY : Protecting intellectual property of deep neural networks with watermarking. *In Asia Conf. on Computer and Communications Security*, 2018.

[9] Renjie ZHU, Xinpeng ZHANG, Mengte SHI et Zhenjun TANG : Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020(1):1–12, 2020.