# Réseaux neuronaux convolutionnels et reconnaissance d'images : évaluation des risques

Franck Leprévost, Ali Osman Topal & Enea Mancellari
University of Luxembourg, house of numbers

**La profusion d'images et leur utilisation dans un nombre important d'applications, certaines d'entre elles particulièrement stratégiques, a conduit au développement de processus automatisés de classification. Les réseaux neuronaux convolutionnels entraînés sont parmi les outils dominants pour ces tâches à l'heure actuelle. Ils sont cependant exposés à des attaques pouvant les conduire à des erreurs, aux conséquences potentiellement dramatiques. Cet article présente certains des risques auxquels sont exposés les réseaux neuronaux convolutionnels. Il décrit également une série d'attaques récentes contribuant à l'état de l'art du domaine, des méthodes de défense, et des pistes de recherche concrètes.**

Le nombre de contextes dans lesquels la reconnaissance et la classification automatique d'images est cruciale n'a cessé d'augmenter depuis plusieurs années. Elle englobe les applications en matière de conduite autonome, d'accès à des bâtiments, de surveillance satellitaires, de diagnostics médicaux, etc. Parmi les outils permettant le traitement rapide et massif de grandes quantités d'images, les réseaux neuronaux convolutionnels (CNN) entraînés sont, avec les *transformers* que nous n'aborderons pas ici, probablement parmi les plus efficaces et les plus utilisés aujourd'hui.

Les CNN ne sont cependant pas infaillibles. Ils peuvent également commettre des erreurs aux conséquences potentiellement dramatiques. Ces

DOI:10.48556/SIF.1024.25.121

erreurs peuvent être le résultat d'images ambiguës, mais elles peuvent aussi être le fruit d'attaques conçues sciemment à ces fins.

L'objet du présent article est d'établir un état des lieux succinct d'une partie de l'état de l'art concernant ces attaques, la manière dont elles sont conçues et les défis qu'elles rencontrent. Il aborde également des méthodes de défense que l'on peut mettre en place pour détecter leur existence et limiter leur impact, et enfin présente certaines directions de recherche actuelles.

L'article est organisé comme suit. La section «*Images*» fournit quelques éléments introductifs sur les images (taille, source, nature, usage). La section «*Réseaux neuronaux convolutionnels*» décrit succinctement ce qu'est un CNN, les méthodes d'entraînement, les ensembles utilisés à ces fins, et le formalisme des données en sortie. On illustre dans la section «*Les risques d'erreurs de classification*» les risques d'erreurs de classification et leurs conséquences potentielles. La section 5 explique comment fonctionnent les attaques sur les CNN, décrit la typologie et les scénarii des attaques les plus fréquentes, ainsi que les attentes en matière de qualité visuelle des images hostiles créées par ces attaques. La section «*Où attaquer ?*» discute les défis technologiques auxquels fait face un attaquant selon qu'il attaque des images qui appartiennent au domaine basse résolution ou au domaine haute résolution, et donne une liste de méthodes génériques, qui fonctionnent pour tout CNN, toute taille d'image, toute attaque et tout scénario. Des méthodes de défense des applications utilisant les CNN sont présentées dans la section «*Les méthodes de défense*». La «*Conclusion*» annonce certains résultats non encore publiés et fournit une série de pistes de recherche.

## Images

Les images traitées automatiquement ont des tailles variables comme illustré sur la grille d'images de la figure 1. Celles de la première ligne, extraites de la base de donnée MNIST [9], sont de taille $28 \times 28$ pixels ; celles de la deuxième ligne, extraites de CIFAR-10 [16], sont de taille $32 \times 32$ ; celles de la troisième sont de taille (proche de) $224 \times 224$ et proviennent de ImageNet [8], et celles de la quatrième ligne sont des images de haute résolution collectées sur Internet (et sujettes au *Creative commons licenses*) et, pour la troisième en partant de la gauche, fournie gracieusement par l'artiste français Speedy Graphito [26]. La plus grande image représentée est de taille $2448 \times 3264$.

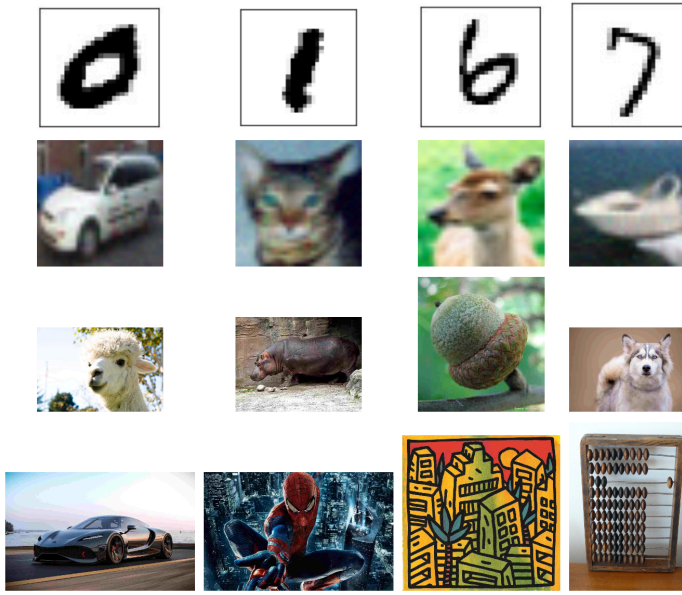

Fig. 1. Exemples d'images avec des tailles *variant* de $28 \times 28$ à $2448 \times 3264$.

Les images diffèrent non seulement en ce qui concerne leur taille, mais également selon leur nature et leur mode d'utilisation, ainsi qu'illustré succinctement dans la table 1.

| Nature | Mode d'utilisation |
|---|---|
| Privée | Réseaux sociaux |
| Artistique | Catalogues raisonnés, enchères |
| Médicale | Diagnostic & traitement |
| Pseudo-aléatoire | Captchas |
| Routes | Véhicules autonomes |
| Aéroports | Securité & antiterrorisme |
| Satellites | Militaire |
| etc. | etc. |

Table 1. Images : modes d'utilisation selon leur nature.

Compte tenu de la profusion d'images, de leur utilisation dans de nombreuses applications, et de la nécessité de les traiter rapidement, il est donc pertinent de disposer de systèmes automatisés de traitement pour classifier les images selon leurs contenus.

## Réseaux neuronaux convolutionnels

Les réseaux neuronaux convolutionnels (CNN) répondent à ces besoins. En résumé, un CNN est un système complexe de couches connectées, et dont les couches ont des objectifs différents. Un exemple schématique de CNN est illustré dans la figure 2 ; les premières couches de taille $224 \times 224$ capturent les données essentielles de l'image, les couches suivantes procèdent à des calculs sur les données collectées, aboutissant à une dernière couche qui donne un vecteur de classification de l'image de $1000$ valeurs (voir plus loin dans cette section pour une description conceptuelle du processus générique de classification).

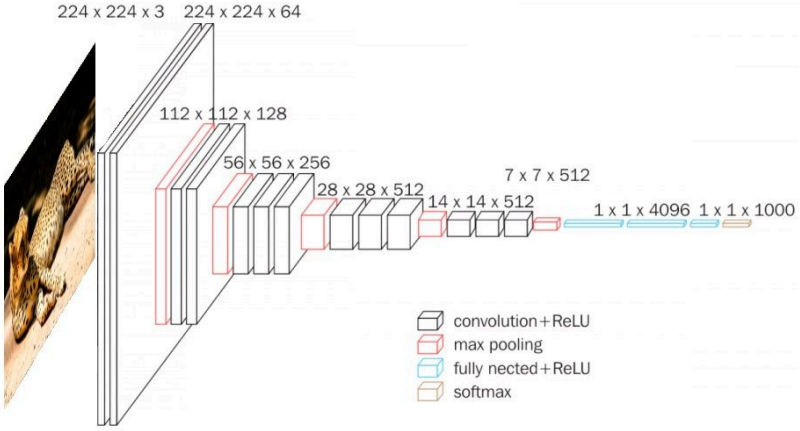


Fig. 2. Exemple typique de représentation d'un CNN [25].

La Table 2 donne une liste de 10 CNN $\mathcal{C}_k$ entraînés sur ImageNet, représentatifs des réseaux classiques historiques du domaine. D'un CNN à l'autre, le nombre de couches diffère (par exemple VGG-19 possède 19 telles couches), ainsi que le nombre total de paramètres qui les régissent, en l'occurrence de quatre millions dans le cas de MobileNet et NASNEtMobile à 144 millions pour VGG-19. Les notions de top-1 et top-5 exactitude sont clarifiées un peu plus loin.

| $\mathcal{C}_k$ | Nom du CNN | Paramètres | Top-1 Exactitude | Top-5 Exactitude |
|---|---|---|---|---|
| $\mathcal{C}_1$ | DenseNet121 [14] | 8M | 0.750 | 0.923 |
| $\mathcal{C}_2$ | DenseNet169 [14] | 14M | 0.762 | 0.932 |
| $\mathcal{C}_3$ | DenseNet201 [14] | 20M | 0.773 | 0.936 |
| $\mathcal{C}_4$ | MobileNet [13] | 4M | 0.704 | 0.895 |
| $\mathcal{C}_5$ | NASNetMobile [36] | 4M | 0.744 | 0.919 |
| $\mathcal{C}_6$ | ResNet50 [12] | 26M | 0.749 | 0.921 |
| $\mathcal{C}_7$ | ResNet101 [12] | 45M | 0.764 | 0.928 |
| $\mathcal{C}_8$ | ResNet152 [12] | 60M | 0.766 | 0.931 |
| $\mathcal{C}_9$ | VGG16 [25] | 138M | 0.713 | 0.901 |
| $\mathcal{C}_{10}$ | VGG19 [25] | 144M | 0.713 | 0.900 |

Table 2. Exemples de 10 CNN entraînés sur ImageNet, leur nombre de paramètres (en millions), et leur Top-1 et Top-5 exactitude.

Que fait un CNN dans le cadre de la reconnaissance d'images ? Schématiquement, il reçoit en entrée une image, et produit en sortie un vecteur de classification de l'image selon des catégories. Plus précisément (tout en restant schématique dans cet article), un CNN, successivement, (1) est entraîné, (2) passe un examen, (3) travaille.

Un CNN entraîné est donc un CNN qui a d'abord été exposé à des milliers ou des millions d'images. Lors de la phase (1) d'entraînement, le CNN reçoit essentiellement deux informations : l'image et ce qu'elle représente (formalisé selon des termes décrits plus bas). Ensuite, lors de la phase (2) d'examen, le CNN est exposé à des images de test, et doit « dire » ce qu'elles représentent. S'il réussit, il est alors considéré comme apte à être exposé à des images ne faisant pas partie de celles qu'il a déjà reçues en entrée lors de l'entraînement ou de l'examen : il sera en mesure de fournir une classification de ce que cette nouvelle image représente, sur la base de l'expérience acquise au cours de l'entraînement.

La table 3 donne des exemples d'ensembles d'entraînement. Pour chacun, elle précise la taille de ces images, et le nombre d'images utilisées pour la phase (1) et la phase (2). Elle donne aussi le nombre de catégories dans lesquelles le CNN est amené à classifier les images. Par exemple, les CNN entraînés sur MNIST classent des images représentant des chiffres (voir première ligne de la figure 1), donc dans les 10 catégories $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. Les CNN entraînés sur CIFAR-10 classent les images également dans 10 catégories, mais qui sont différentes des précédentes, puisqu'il s'agit (voir

deuxième ligne de la figure 1) d'animaux (oiseau, chat, cerf, chien, grenouille, cheval) ou d'objets (avion, voiture, bateau, camion). Les CNN entraînés sur ImageNet classifient les images dans 1000 catégories (des exemples provenant d'ImageNet sont donnés en troisième ligne de la figure 1). Les notions de « top-1 exactitude » et « top-5 exactitude » spécifiés dans la table 2 traduisent le taux de réussite à l'examen d'un CNN entraîné. Par exemple, VGG-19 entraîné sur ImageNet a une top-1 exactitude de 0.713 et une top-5 exactitude de 0.900. Cela signifie que VGG-19 a correctement identifié la bonne catégorie des 100 000 images de test dans 71.3 % des cas, et que cette bonne catégorie se trouvait parmi les cinq catégories dominantes identifiées par le CNN dans 90.0 % des cas.

| Ensemble d'entraînement | # images d'entraînement | # images de test | Taille des images | # Catégories |
|---|---|---|---|---|
| MNIST | 60 000 | 10 000 | $28 \times 28$ | 10 |
| CIFAR-10 | 50 000 | 10 000 | $32 \times 32$ | 10 |
| ImageNet | 1.2 million | 100 000 | $224 \times 224$ | 1000 |

Table 3. Exemples d'ensembles d'entraînement pour CNN.

Mathématiquement, le processus de classification d'une image $\mathcal{I}$ par un CNN $\mathcal{C}$ peut être visualisé comme suit :

$$\mathcal{I} \longrightarrow \mathcal{C} \longrightarrow \mathbf{o}_{\mathcal{C}}(\mathcal{I}) = [v_1, \cdots, v_\ell], \qquad \text{où} \qquad 0 \leq v_j \leq 1 \quad \text{et} \quad \sum_{j=1}^{\ell} v_j = 1. \tag{1}$$

En ce cas, la longueur $\ell$ du vecteur de classification $\mathbf{o}_{\mathcal{C}}(\mathcal{I})$ désigne le nombre de catégories $c_1, \cdots, c_\ell$ dans lesquelles $\mathcal{C}$ classe les images auxquelles il est exposé ($\ell = 10$ pour les CNN entraînés sur MNIST ou CIFAR-10, et $\ell = 1000$ pour ceux entraînés sur ImageNet). Chaque valeur $v_j$ dépend de $\mathcal{C}$ et de $\mathcal{I}$, et mesure la plausibilité déclarée par $\mathcal{C}$ que $\mathcal{I}$ appartienne à la catégorie $c_j$. Le CNN $\mathcal{C}$ classifie l'image $\mathcal{I}$ dans la catégorie $c_k$ correspondant à la valeur $v_k$ dominante parmi toutes les valeurs $v_j$.

La figure 3 montre un exemple typique de classification fournie par un CNN entraîné sur CIFAR-10, lorsqu'il est exposé à l'image représentée. Essentiellement (et de nouveau, de manière schématique), la valeur donnée par le CNN pour la catégorie « chat » est ici $v_{\text{chat}} = 0.97$, celle pour la catégorie « chien » est $v_{\text{chien}} = 0.01$, et la somme des huit autres valeurs est 0.02.

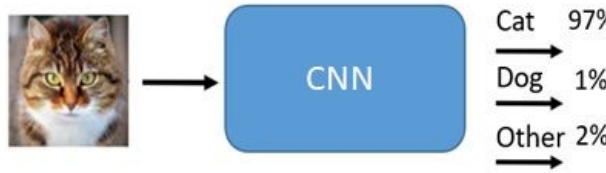


Fig. 3. Exemple représentatif de classification d'une image par un CNN entraîné.

## Les risques d'erreurs de classification

Les risques liés à la classification d'images peuvent se résumer en deux phrases, en quelque sorte duales l'une de l'autre : « ne pas voir ce qui est » et « voir ce qui n'est pas ». Ainsi, la figure 4 nécessite une grande attention de la part de l'œil humain pour déceler l'existence d'un chien couché sur le sofa.

Fig. 4. Illustration de la difficulté pour l'œil humain à déceler ce qui est.

À l'instar des êtres humains, les CNN peuvent aussi commettre des erreurs de classification d'images. Des images peuvent tromper et les uns, et les autres, mais en général elles le font différemment. Les erreurs peuvent certes être dues à des images très particulières. Elles peuvent aussi résulter d'attaques, potentiellement avec des conséquences dramatiques, comme illustré sur deux cas dans la figure 5 [10,23].


Fig. 5. Attaques et conséquences.

Pour les deux images de gauche de la figure 5, l'ajout de lunettes sur le visage de l'homme a conduit le CNN à classifier l'image résultante comme étant la célèbre actrice représentée au dessous de lui. Pour les deux images de droite, l'ajout de petites bandes noires et de petites bandes blanches sur le signe « STOP » a conduit le CNN à interpréter l'image résultante comme étant le signal d'une limitation de vitesse à 45 miles par heure. Les conséquences de telles méthodes peuvent être catastrophiques.

## Comment fonctionnent les attaques ?

Les attaques procèdent par l'ajout de bruit hostile à des images selon un schéma illustré par la figure 6 : un CNN classifie correctement dans la catégorie « chat » une image « propre » de chat ; on y ajoute du bruit hostile ; le CNN classifie l'image « hostile » résultante comme étant dans la catégorie « chien », alors même qu'un humain classifierait l'image hostile dans la catégorie « chat » (dans le cas présent, les deux images sont même indistinguables pour l'œil humain, nous revenons plus loin sur cette notion).


Fig. 6. Ajout de bruit hostile à une image pour tromper un CNN.

Le bruit hostile dépend de :
– l'image propre ;
– le CNN ;
– le scenario de l'attaque ;
– le niveau de connaissance qu'a l'attaquant sur le CNN ;
– la qualité visuelle attendue pour l'image hostile.
Il est important de noter que l'ajout de bruit aléatoire ne fonctionne pas, en particulier si l'on souhaite tromper aussi bien les humains — « ceci est un chat » — et les CNN — « l'image appartient à la catégorie chien ».

Une attaque dépend fortement du niveau de connaissance qu'a l'attaquant du CNN à le tromper. Dans une attaque de type « boîte blanche », l'attaquant connaît parfaitement le CNN visé (son architecture, le nombre de couches, ses paramètres, etc.). Une attaque de type « boîte noire » est beaucoup plus difficile (et correspond aux cas rencontrés en pratique), puisque l'attaquant ne sait rien du CNN qu'il cherche à attaquer.

Par ailleurs, à partir d'une image propre classée dans une catégorie $c_a$, une attaque peut suivre plusieurs scenarii. Les plus fréquents sont, d'une part, le « *untargeted scenario* », où l'objectif est de créer une image hostile classifiée par le CNN dans n'importe quelle catégorie sauf celle de l'image propre, et, d'autre part, le « *targeted scenario* », où une catégorie est fixée à l'avance, l'objectif étant de créer une image hostile classée par le CNN dans cette catégorie précise et pas une autre. Dans le cas du *targeted scenario*, on peut aussi poser des conditions sur le degré d'exigence souhaité pour la classification du CNN de l'image hostile dans la catégorie visée $c_t$ (par exemple en demandant qu'elle excède une valeur fixée à l'avance, ou que la distance avec la seconde catégorie la mieux placée excède également une valeur fixée à l'avance, etc.).


Fig. 7. Exemples d'images hostiles.

Enfin, la qualité visuelle des images hostiles peut être « distinguable », auquel cas l'œil humain verra que des modifications ont été apportées, ou non. La figure 7 illustre cela. Les images de la première rangée sont les images propres, classées par le CNN dans les bonnes catégories. Les images de la seconde rangée sont les images hostiles. L'œil humain remarque que des modifications ont été apportées aux quatre premières (voir [27,15,22,32]). Il en est incapable pour la dernière ([18,30], voir également la section « Où attaquer ? »).

En pratique, les cas réels d'attaques nécessitent la combinaison la plus difficile : boîte noire, scénario *targeted*, des images hostiles indistiguables. De telles attaques existent, notamment [11,5,4,2,1,33], ainsi que notre attaque à base d'un algorithme évolutionnaire [28] (nous évoquons une autre de nos attaques dans la conclusion).

## Où attaquer ?

La taille de plus en plus importante des images utilisées en pratique pose des défis dont l'ampleur va grandissant. En effet, d'une part, la taille d'entrée des CNN est souvent petite ($32 \times 32$ ou $224 \times 224$), les images sont au besoin retaillées, et les attaques portent traditionnellement sur des images de la taille d'entrée des CNN ; autrement dit : les attaques sont dans le domaine basse résolution (LR), comme illustré dans la figure 8.

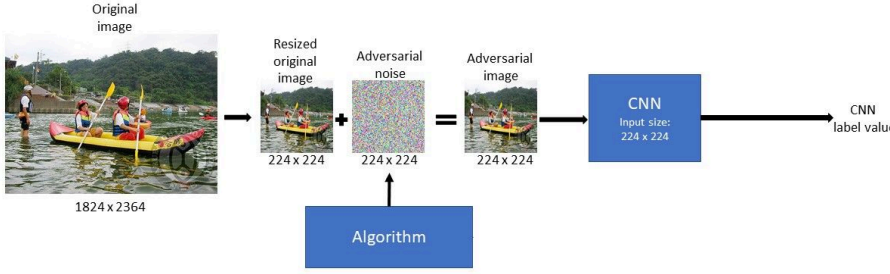


Fig. 8. Attaque dans le domaine LR.

Or, d'autre part, les images propres en entrée peuvent être grandes, et appartenir au domaine haute résolution (HR). C'est notamment le cas de la plupart des images mises en ligne sur les réseaux sociaux. Il peut néanmoins s'avérer souhaitable de protéger la sphère privée des individus, en limitant leur traçage par des CNN à partir des images mises (parfois inconsidérément) sur les réseaux sociaux. Se pose alors la question de la possibilité d'attaquer directement dans le domaine HR. Un tel schéma d'attaque est illustré dans la figure 9, où l'intention est d'introduire du bruit hostile directement dans la grande image propre.

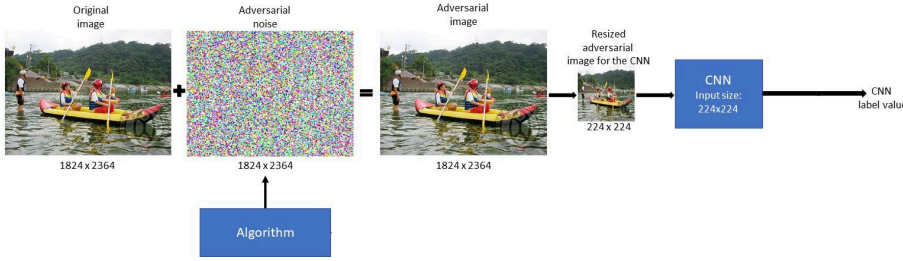


Fig. 9. Attaque dans le domaine HR.

La construction d'images HR hostiles conduit à trois défis : vitesse de création de telles images hostiles, adversité des images une fois retaillées pour s'adapter à la taille d'entrée des CNN, qualité visuelle des images hostiles dans le domaine HR.

Il se pose alors notamment la question de l'existence de méthodes **génériques** qui résolvent ces trois défis tout en fonctionnant pour tout CNN spécifique, toute taille d'image, toute attaque et tout scenario. À notre connaissance, notre équipe est la première à avoir apporté une réponse positive à cette question sous la forme de trois méthodes. En résumé, la première [17] consiste à « relever » dans le domaine HR une image hostile construite dans le domaine LR, c'est-à-dire essentiellement à étendre au domaine HR une image hostile provenant du domaine LR ; la deuxième [18,30] consiste à ne relever que le bruit hostile, et à l'additionner à l'image propre dans le domaine HR ; la troisième [19] identifie les zones spécifiques de l'image sur lesquelles faire porter l'attaque dans le domaine HR. Il est à noter que la méthode 2 raffine et simplifie méthode 1, et que la méthode 3 peut être combinée avec la méthode 2.

À titre d'exemple, la figure 10 extraite de [30], illustre la deuxième méthode. Une image propre de haute résolution (domaine HR), de taille $960 \times 1280$ en l'occurrence, est réduite à la taille $224 \times 224$ de basse résolution (domaine LR). Une attaque (peu importe laquelle) sur cette image de basse résolution crée une image hostile également dans le domaine LR. la différence entre l'image hostile et l'image propre réduite dans le domaine LR fournit le bruit hostile dans le domaine LR. Ce bruit hostile (et rien que lui) est « relevé » du domaine LR au domaine HR par une fonction d'interpolation convenable. Ce bruit hostile est ajouté à l'image propre dans le domaine HR, ce qui fournit une image dans le domaine HR qui est potentiellement hostile. Cette image est réduite au format LR et donnée en entrée au CNN. Si le CNN classifie cette image réduite dans une classe différente de celle de l'image propre, alors l'image HR dont elle provient est une image hostile dans le domaine HR.

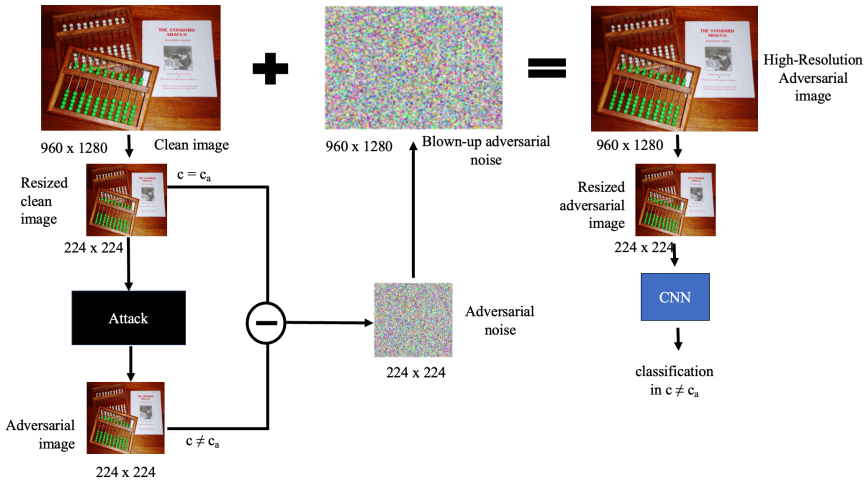


Fig. 10. La stratégie « *noise blowing up* ».

De telles méthodes mènent à des images hostiles d'une très grande qualité visuelle, comme l'illustre la figure 11, obtenue à partir de notre attaque [18,30] de type boîte noire pour le scénario *targeted* sur une image de grande taille (sur les catégories « guépard » pour l'image propre et « minibus » pour la catégorie cible dans laquelle l'attaque doit conduire le CNN à ranger l'image hostile).

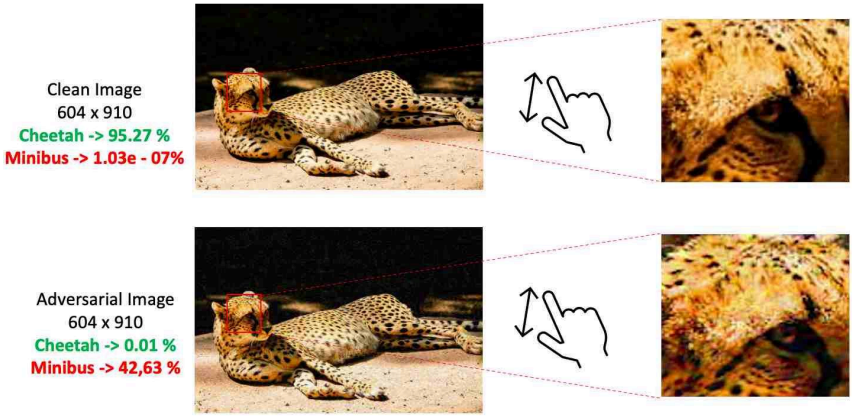


Fig. 11. Stratégie « Noise Blowing-Up » avec notre attaque à base d'algorithme évolutionnaire sur VGG-16 entraîné sur ImageNet.

## Les méthodes de défense

Des méthodes destinées à protéger les CNN – ou, plus exactement, les applications utilisant des CNN – de ce type d'attaque ont été développées (avec des succès variables). On attend en général d'un système de défense qu'il détecte les images hostiles, potentiellement qu'il procède à leur exclusion avant tout autre traitement, et qu'il alerte l'utilisateur du fait qu'une attaque est en cours.

Ces détecteurs peuvent être supervisés ou non. Un détecteur supervisé a connaissance à l'avance des attaques qu'il subira, et renforcera la défense des CNN en ajoutant des images hostiles dans leur ensemble d'entraînement, en spécifiant évidemment que ces images sont hostiles et doivent être reconnues comme telles. Un détecteur non supervisé est plus réaliste, puisqu'on ne suppose alors pas connues les attaques à l'avance. Les performances de ces détecteurs sont mesurées par une série d'indicateurs (taux de détection, taux de faux positifs, complexité, *overhead*, précision, *recall*, F1, *inference time*, etc).

LID [21] est un exemple de détecteur supervisé. NIC [24], ANR [20], FS [35] et ShuffleDetect [7] sont des exemples de détecteurs non supervisés. À titre d'illustration, ShuffleDetect s'inspire du test probabiliste de primalité de Fermat. Concrètement, une image $\mathcal{I}$ (carrée dans notre exemple, de taille $224 \times 224$) est totalement partitionnée en $N$ morceaux ($N = 16$ dans notre exemple).

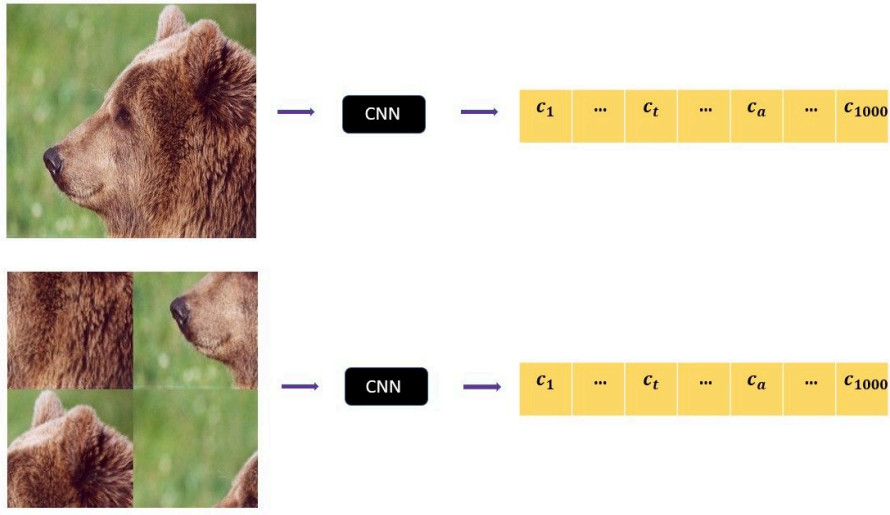


Fig. 12. ShuffleDetect : action de $\sigma$, classification de $\mathcal{I}$ et de $sh_\sigma(\mathcal{I})$.

Or il y a une action sur ces morceaux du groupe symétrique[1,2] $\mathcal{S}_N$. L'on prend 100 permutations aléatoires $\sigma_1, \cdots, \sigma_{100} \in \mathcal{S}_N$ (ce qui est largement possible parmi les $N!$ permutations du groupe $\mathcal{S}_N$ considéré). Pour chaque telle permutation $\sigma$, on construit l'image mélangée $sh_\sigma(\mathcal{I})$ en permutant les morceaux de $\mathcal{I}$ selon l'action de $\sigma$. On obtient les vecteurs de classifications du CNN pour $\mathcal{I}$ et pour $sh_\sigma(\mathcal{I})$. La figure 12 illustre ce processus.

On compare les catégories dominantes et l'on décrète l'image $\mathcal{I}$ comme propre si, pour au moins la majorité des permutations $\sigma$, les images mélangées $sh_\sigma(\mathcal{I})$ sont classées par le CNN dans la même catégorie que $\mathcal{I}$, et comme adversaire sinon. Cette approche, validée expérimentalement dans [7], fait de ShuffleDetect une première rapide ligne de défense susceptible de déceler des images hostiles ou nécessitant des traitements complémentaires (le plus souvent plus coûteux en temps).

## Conclusion

Cet article a illustré par un survol de certaines techniques existantes les défis des attaques sur les mécanismes automatiques de reconnaissance d'image par des CNN. Dans la course entre attaques et défenses, l'avantage semble rester à l'attaquant, d'autant plus que, malgré l'existence d'algorithmes de défense, il semble de plus en plus difficile de détecter l'existence même d'une attaque. La qualité visuelle des images hostiles est devenue impressionnante, même pour des images haute résolution, des attaques de type boîte noire et des scénarios *targeted*.

Jusqu'à récemment, toutefois, les attaques dans ces contextes très exigeants semblaient particulièrement chronophages en comparaison des attaques de type boîte blanche. En effet, là où les attaques de type boîte blanche (c'est-à-dire celles où l'attaquant connaît parfaitement le CNN à attaquer) demandaient de l'ordre d'une minute dans un environnement expérimental donné, les attaques de type boîte noire (celles où, schématiquement, l'attaquant n'a aucune idée du CNN à attaquer) pouvaient nécessiter au moins quinze fois plus de temps (parfois bien davantage) dans le même environnement expérimental (voir par exemple [30]). Cette situation a récemment changé. Nous montrons dans [29] comment construire, en des temps comparables (voire inférieurs) aux attaques de type boîte blanche, des images hostiles par une attaque de type boîte noire, et cela, contre les dix CNN répertoriés dans la table 2 [5], et pour des images HR. Notre attaque NBUGan utilise la technique de relèvement du bruit hostile de [30] (consistant à identifier le bruit ajouté pour créer une image hostile du domaine LR, et à « relever » ce bruit du domaine LR vers le domaine HR et à l'ajouter à l'image propre de départ pour

1. Le groupe symétrique $\mathcal{S}_N$ est le groupe de toutes les permutations d'un ensemble à $N$ éléments.
2. Dans notre cas, il y a également une action du groupe diédral $D_4$ sur nos morceaux, mais nous ne la considérons pas ici.

créer une image hostile dans le domaine HR) et une méthode de GAN (generative adversarial network ; un modèle dans lequel schématiquement deux réseaux de neurones sont en compétition, le générateur ayant pour objectif de créer une image hostile et le discriminateur ayant pour objectif de déceler si l'image provient ou non du générateur) que nous avons développée pour nos besoins. Cette méthode, validée expérimentellement, atteint un taux de succès de 93 % sur des images haute définition et dans des temps de l'ordre d'une minute (dans l'environnement Nvidia Tesla V100 GPGPU de l'IRIS HPC cluster de l'université du Luxembourg [31]).

La qualité visuelle est illustrée dans les figures 13 et 14 avec des images de taille $2336 \times 3504$. L'image propre (figure 13) est classée comme « baseball » par ResNet-50 entraîné sur ImageNet, avec un degré de confiance de 99.9 %. L'image hostile (figure 14), que NBUGan a construite en une minute environ, est classée par ResNet-50 dans la catégorie « *ladle* » (prise au hasard parmi les 999 catégories différentes de « baseball ») avec un degré de confiance de 90.9 % (nous avions exigé que cette valeur soit au moins 90 %).

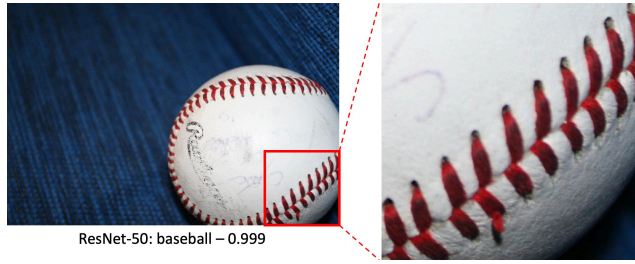


Fig. 13. Image propre.

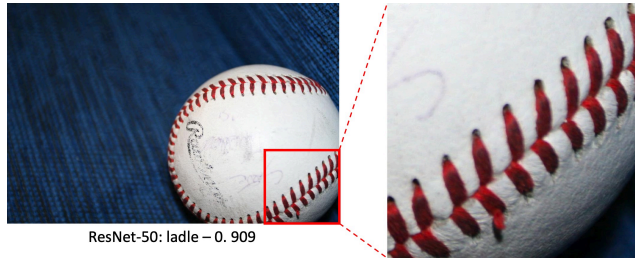


Fig. 14. Image hostile.

Comparaison visuelle d'images propres et d'images hostiles de haute résolution obtenues par NBUGan contre ResNet-50 pour la paire (« baseball-ladle »), avec un degré de confiance ≥ 0.90 %.

Les questions d'attaque sur les CNN dans leurs tâches de classification d'images comportent d'autres défis que nous n'avons pu détailler ici. Parmi eux, on peut citer la difficulté à disposer de métriques permettant de vraiment déceler la perception humaine d'images identiques ; la transférabilité du bruit hostile (en d'autres termes, la capacité de créer une seule et même image capable de tromper plusieurs CNN simultanément) ; la sensibilité des classifications des CNN aux perturbations et leurs conséquences sur la fragilité des images hostiles. Enfin, il convient de noter que les techniques d'attaques sur les images sont susceptibles de s'appliquer à d'autres domaines, tels la voix [3], [6] et les vidéos [34], et susceptibles, là aussi, de créer des situations hautement problématiques. Notre équipe travaille actuellement sur certaines de ces pistes.

## Références


**[1]** Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, et Matthias Hein. 2020. Square Attack: a query-efficient black-box adversarial attack via random search. https://arxiv.org/abs/1912.00049.

**[2]** Wieland Brendel, Jonas Rauber, et Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. https://arxiv.org/abs/1712.04248.

**[3]** Nicholas Carlini et David Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. https://arxiv.org/abs/1801.01944.

**[4]** Jianbo Chen, Michael I. Jordan, et Martin J. Wainwright. 2020. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. https://arxiv.org/abs/1904.02144.

**[5]** Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, et Cho-Jui Hsieh. 2017. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (CCS '17). https://doi.org/10.1145/3128572.3140448.

**[6]** P. Cheng, Y. Wang, P. Huang, Z. Ba, X. Lin, F. Lin, L. Lu, et K. Ren. 2024. ALIF: Low-Cost Adversarial Audio Attacks on Black-Box Speech Platforms using Linguistic Features. In *2024 IEEE Symposium on Security and Privacy (SP)*, 59-59. https://doi.org/10.1109/SP54263.2024.00056.

**[7]** Raluca Chitic, Ali Osman Topal, et Franck Leprévost. 2023. ShuffleDetect: Detecting Adversarial Images against Convolutional Neural Networks. *Applied Sciences* 13, 6. https://doi.org/10.3390/app13064068.

**[8]** Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, et Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. https://doi.org/10.1109/CVPR.2009.5206848.

**[9]** Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6: 141-142. https://doi.org/10.1109/MSP.2012.2211477.

**[10]** Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, et Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1625-1634. https://doi.org/10.1109/CVPR.2018.00175.

**[11]** Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, et Kilian Weinberger. 2019. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, 2484-2493.

**[12]** Kaiming He, Xiangyu Zhang, Shaoqing Ren, et Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

**[13]** Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, et Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

**[14]** Gao Huang, Zhuang Liu, Laurens Van Der Maaten, et Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700-4708.

**[15]** Malhar Jere, Loris Rossi, Briland Hitaj, Gabriela Ciocarlie, Giacomo Boracchi, et Farinaz Koushanfar. 2019. Scratch that! An evolution-based adversarial attack against neural networks. *arXiv preprint arXiv:1912.02316*.

**[16]** Alex Krizhevsky, Vinod Nair, et Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). http://www.cs.toronto.edu/~kriz/cifar.html.

**[17]** Franck Leprévost, Ali Osman Topal, Elmir Avdusinovic, et Raluca Chitic. 2022. Strategy and Feasibility Study for the Construction of High Resolution Images Adversarial against Convolutional Neural Networks. In *Intelligent Information and Database Systems, 13th Asian Conference, ACIIDS 2022 (Ho-Chi-Minh-City, Vietnam, November 28-30, 2022)*, 467-480.

**[18]** Franck Leprévost, Ali Osman Topal, et Enea Mancellari. 2023. Creating High-Resolution Adversarial Images Against Convolutional Neural Networks with the Noise Blowing-Up Method. In *Intelligent Information and Database Systems*, 121-134.

**[19]** Franck Leprévost, Ali Osman Topal, Enea Mancellari, et Kittichai Lavangnananda. 2023. Zone-of-Interest Strategy for the Creation of High-Resolution Adversarial Images Against Convolutional Neural Networks. In *2023 15th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 127-132. https://doi.org/10.1109/ICITEE59582.2023.10317668.

**[20]** Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, et Xiaofeng Wang. 2021. Detecting Adversarial Image Examples in Deep Neural Networks with Adaptive Noise Reduction. *IEEE Transactions on Dependable and Secure Computing* 18, 1: 72-85. https://doi.org/10.1109/tdsc.2018.2874243.

**[21]** Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, et James Bailey. 2018. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. https://arxiv.org/abs/1801.02613.

**[22]** Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, et Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 372-387. https://ieeexplore.ieee.org/document/7467366.

**[23]** Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, et Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS '16), 1528-1540. https://doi.org/10.1145/2976749.2978392.

**[24]** Ma Shiqing, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, et Xiangyu Zhang. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. https://doi.org/10.14722/ndss.2019.23415.

**[25]** Karen Simonyan et Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

**[26]** SpeedyGraphito. 2020. *Mes 400 Coups*. Panoramart.

**[27]** Jiawei Su, Danilo Vasconcellos Vargas, et Kouichi Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* 23, 5.

**[28]** Ali Osman Topal, Raluca Chitic, et Franck Leprévost. 2023. One evolutionary algorithm deceives humans and ten convolutional neural networks trained on ImageNet at image recognition. *Applied Soft Computing* 143: 110397.

**[29]** Ali Osman Topal, Enea Mancellari, Thomas Gillet, et Franck Leprévost. 2024. Defying Expectations: High-Speed Black-Box Attack for High-Resolution Adversarial Image Generation. *(Submitted)*.

**[30]** Ali Osman Topal, Enea Mancellari, Franck Leprévost, Elmir Avdusinovic, et Thomas Gillet. 2024. The Noise Blowing-Up Strategy Creates High Quality High Resolution Adversarial Images against Convolutional Neural Networks. *Applied Sciences* 14, 8: 3493.

**[31]** S. Varrette, P. Bouvry, H. Cartiaux, et F. Georgatos. 2014. Management of an Academic HPC Cluster: The UL Experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, 959-967.

**[32]** Junde Wu. 2020. Generating adversarial examples in the harsh conditions. *CoRR* abs/1908.11332. https://arxiv.org/abs/1908.11332.

**[33]** Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, et Dawn Song. 2019. Generating Adversarial Examples with Adversarial Networks. https://arxiv.org/abs/1801.02610.

**[34]** Sicheng Xu, Guojun Chen, Yu-Xiao Guo, Jiaolong Yang, Chong Li, Zhenyu Zang, Yizhong Zhang, Xin Tong, et Baining Guo. 2024. VASA-1: Lifelike Audio-Driven Talking Faces Generated in Real Time. https://arxiv.org/abs/2404.10667.

**[35]** Weilin Xu, David Evans, et Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings 2018 Network and Distributed System Security Symposium* (NDSS 2018). https://doi.org/10.14722/ndss.2018.23198.

**[36]** Barret Zoph, Vijay Vasudevan, Jonathon Shlens, et Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697-8710.