Aurélien Desvilles 01/02/2026

RCP 209 : Apprentissage statistique 2

Attaques adverses par descente rapide de gradient sur des modèles pré-entrainés sur ImageNet

Introduction

Une Adversarial Attack, ou attaque adverse, consiste à tromper ou à détourner un modèle de Machine Learning grâce à des données. En entraînant un modèle sur des données imprécises, ou volontairement falsifiées, il est possible d'impacter négativement ses performances futures. Cependant, même un modèle déjà entraîné sur des données de qualité peut être corrompu par des données frauduleuses en entrée. Les attaques adverses peuvent être sous-divisées selon leur fonctionnement en « boîte blanche » ou « boîte noire », respectivement dans le cas où l'attaquant a accès ou nom aux paramètres du modèle.

Un domaine particulièrement impacté par ce type d'attaque est la reconnaissance d'image. Ainsi, de petites perturbations (distorsions), quasiment invisible à l'œil humain, ajoutées à une image peuvent facilement perturber le modèle et entrainer une mauvaise classification [1].

Nous nous concentrerons ici sur l'attaque par la méthode du signe du gradient rapide (FGSM), qui utilise les gradients du réseau de neurones pour créer un exemple adverse. Pour une image d'entrée, elle utilise les gradients de la fonction de perte par rapport à cette image afin de créer une nouvelle image qui maximise la perte [5]. Cette nouvelle image est appelée image adverse. On peut résumer cela par l'expression suivante :

$$advX = X + \varepsilon \cdot sign(\nabla_X J(\theta, X, Ypred)) \quad (1)$$

où

- adv_X: Image adverse.
- X: Image d'entrée originale.
- $Ypred$: Classe d'entrée d'origine.
- J: Fonction de perte
- ∇_X : Gradient
- ε : Multiplicateur pour garantir que les perturbations soient faibles.
- θ : Paramètres du modèle.

L'objectif est ainsi de créer une image qui maximise la fonction de perte. Une méthode pour y parvenir consiste à déterminer la contribution de chaque pixel de l'image à la valeur de perte, puis à ajouter une perturbation en conséquence. Cette méthode est assez rapide car il est facile de déterminer la contribution de chaque pixel d'entrée à la perte en utilisant la règle de la chaîne et en calculant les gradients requis. Par conséquent, les gradients sont calculés par rapport à l'image. Cette méthode cible des modèles déjà entrainés pour lesquels les paramètres entrainables ne bougent plus. De plus, il s'agit d'une attaque type « boite blanche » nécessitant préalablement une connaissance de ces paramètres.

Ce travail consistera a implémenter différentes variantes de FGSM sur plusieurs modèles de réseaux de neurones convolutifs pré-entrainés sur la banque d'images publique ImageNet (https://image-net.org/) et d'étudier la robustesse de ces derniers face à ce type d'attaque.

Matériel et méthodes

Jeu de données

N'ayant pas eu accès au jeu officiel d'ImageNet, un autre jeu nommé ImageNet-mini (https://www.kaggle.com/datasets/deeptrial/miniimagenet/data) a donc été utilisé. Bien que non officiel, ce jeu d'images est organisé exactement comme l'original : 1 000 dossiers, chacun portant le nom d'une classe, et un fichier JSON permettant d'associer chaque nom de dossier au nom correspondant, les noms de classes étant les mêmes que pour ImageNet (https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/).

API

Les modèles sont issus de l'API Keras 3 et appelés à l'aide de la bibliothèque TansorFlow. Il s'agit de modèles d'apprentissage profond fournis avec des poids pré-entraînés (https://keras.io/api/applications/). Les poids initialisés sont ceux d'ImageNet et la fonction d'activation finale est de type *softmax* (voire fichier de code). Il n'y a pas de phase d'entrainement sur un nouveau jeu de données.

Modèles utilisés

Afin de rendre compte des disparités de vulnérabilité face aux attaques de type FGSM, trois modèles de réseaux de neurones convolutifs (CNN) aux architectures différentes ont été sélectionnés et testés :

- **VGG-19** : Il s'agit d' un réseau de neurones convolutif profond à 19 couches de poids, comprenant 16 couches convolutives (noyau 3x3, activation ReLU) et 3 couches entièrement connectées. Son architecture, simple et répétitive, facilite sa compréhension et sa mise en œuvre. C'est un réseau performent mais du fait de sa profondeur et de l'utilisation de petits filtres, il requiert une mémoire et une puissance de calcul importantes.[2]
- **ResNet-50** : Il s'agit d'un CNN très connu en vision par ordinateur qui a la particularité de ne contenir que 50 couches (49 couches de convolution et une couche *softmax*), ce qui le rend relativement léger. Sa caractéristique essentielle est son utilisation de connexions résiduelles qui permettent au modèle de sauter certaines étapes du processus d'apprentissage. Ainsi, au lieu de forcer le modèle à faire passer l'information par chaque couche, ces raccourcis lui permettent de transmettre plus directement les détails importants, rendant l'apprentissage plus rapide et plus fiable. [3]
- **Inception-V3** : Plus récent et complexe, son architecture se compose d'une série de modules spécialisés (modules Inception) empilés, avec des opérations de pooling ponctuelles pour réduire la dimensionnalité spatiale. Le modèle implémente une structure symétrique avec plusieurs voies de convolution parallèles dans chaque module. [4]

Les images soumises ont été systématiquement reformatées pour avoir une taille de 224x224 pixels pour VGG-19 et ResNet-50, 299x299 pixels pour Inception-V3. Elles subissent également un prétraitement spécifique à chaque modèle comme décrit dans le **Tableau 1**. Les performances respectives de ces modèles sur le jeu d'images sont similaires à celles obtenues pour ImageNet (**Tableau 2**).

Modèle	Échelle entrée	Centrage	Canal	Espace final
VGG19	[0,255]	Moyenne ImageNet	BGR	≈ [-123, +151]
ResNet50	[0,255]	Moyenne ImageNet	BGR	≈ [-123, +151]
InceptionV3	[0,255]	Normalisation	RGB	[-1, 1]

Tableau 1 : Caractéristique du prétraitement des images par chaque modèle.

	ImageNet official		ImageNet-mini	
Model	**Top-1 Accuracy**	**Top-5 Accuracy**	**Top-1 Accuracy**	**Top-5 Accuracy**
VGG-19	71.3%	90.0%	65.5%	86.7%
ResNet-50	74.9%	92.1%	70.3%	90.1%
Inception-V3	77.9%	93.7%	76.4%	93.3%

Tableau 2 : Performances des modèles sur les jeux de données ImageNet (https://keras.io/api/applications/) et ImageNet-mini. Les valeurs de Top-1 Accuracy et Top-5 Accuracy correspondent à la proportion d'images correctement classées en fonction, respectivement, de la première et des cinq premières probabilités renvoyées.

Résultats

I) Attaque FGSM classique

La première attaque est celle décrite dans l'équation (1) où la perturbation va dans le sens du gradient du réseau de neurones. Dans les exemples ci-dessous, la fonction de perte est de type *crossentropy* pour les trois modèles. Il est important de noter que la perturbation a lieu sur une image prétraitée et non sur l'image originale (**Figure 1**). Elle agit donc comme un filtre spécifique à la fois à l'image et au modèle ciblé (**Figure 2)**.

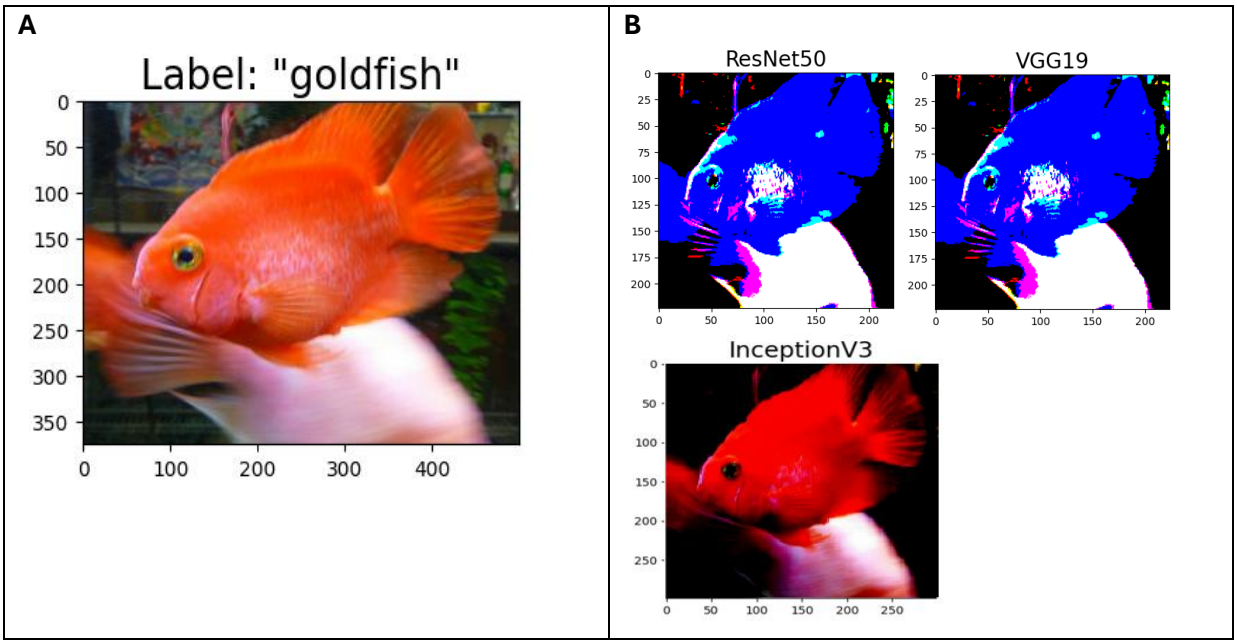


Figure 1 : Exemple de prétraitement d'une image par les différents modèles. **A** : Image ILSVRC2012_val_00000994.JPEG de la classe n01443537 (« goldfish ») du jeu de données ImageNet-mini. **B** : Images prétraitées pour les trois réseaux.

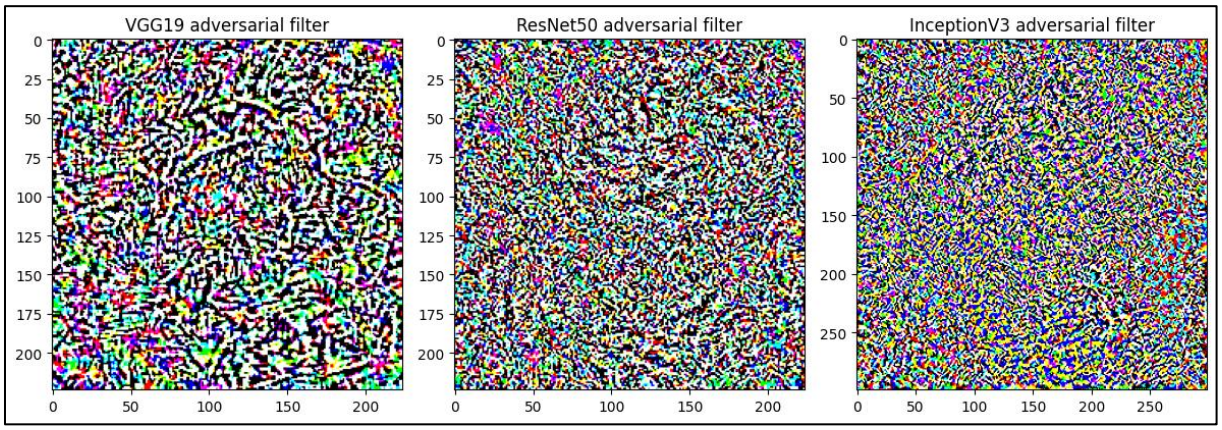


Figure 2 : Filtres obtenus pour chaque modèle à partir de l'image présentée en **Figure 1**. Les valeurs des pixels ont été ramenées dans l'intervalle [0 ;1] pour une meilleure interprétabilité de la perturbation.

Les effets de ces attaques sur l'image prise en exemple peuvent être observés dans la **Figure 3** pour plusieurs valeurs de ε.

Ainsi, sur la **Figure 3 A**, VGG19 reconnait correctement l'image comme « goldfish » (poisson rouge) avec une probabilité de 97.5% pour ε = 0.5, mais lorsque ε dépasse 1 en lui attribuant la classe « anemone fish » (poisson clown) avec des probabilités très élevées. Il reconnait cependant l'image comme étant un poisson et il est certain que les classes « goldfish » et « anemone fish » contiennent des images aux caractéristiques proches.

Sur la **Figure 3 B**, ResNet50 reconnait correctement l'image pour des valeurs de ε inférieures à 3 mais avec des probabilités qui diminuent lorsque ε augmente. A partir de 3, l'image est considérée comme un « lorikeet » (loriquet) et donc plus comme un poisson mais un type de perroquet (il n'est pas impossible qu'une image de perroquet ait des caractéristiques proches d'une image de poisson rouge). Dans cet exemple, les probabilités données par ResNet50 sont souvent bien inférieures à celles données par VGG19, que le modèle se trompe ou non

La **Figure 4** montre que, pour tout le jeu de données, les performances globales de ResNet50 et VGG19 diminuent lorsque ε augmente. Cependant, le taux de bonnes prédiction décroit plus vite pour ResNet50 que pour VGG19 lorsque ε augmente jusqu'à environ 2,3 pour la Top-1 Accuracy (**Figure 4 A**) et 1,6 pour la Top-5 Accuracy (**Figure 4 B**). Lorsque ε est supérieur à ces valeurs, ResNet50 obtient de bien meilleurs résultats que VGG19. Ainsi, pour ε = 7 , ResNet50 obtient des scores de 12,54% pour la Top-1 Accuracy et 32,76% pour la Top-5 Accuracy, contre 7,67% et 21,97% chez VGG19.

Ces observations montrent que ResNet50 semble légèrement plus sensible aux petites perturbations que VGG19, tandis que pour des perturbations plus importantes, ResNet50 est nettement plus résistant que VGG19. Les relativement bonnes performances de VGG19 pour de petites perturbations peuvent en partie s'expliquer par le fait que le gradient passe obligatoirement par toutes les couches de convolution, lesquelles peuvent atténuer ces perturbations. Avec ResNet50, le gradient peut sauter certaines de ces couches et passer presque sans atténuation grâce aux connections résiduelles. Cependant, ces connections résiduelles permettent de garder un chemin « identité » et de mieux saisir les caractéristiques globales de l'images. Lorsque la perturbation est importante, les filtres de convolutions de VGG19 ne peuvent plus suffisamment l'atténuer et cette dernière se diffuse dans tout le réseau ; les performances du modèle s'effondrent tandis que ResNet50 arrive mieux à distinguer la forme de l'objet du bruit produit par la perturbation.

Le cas de InceptionV3 dénote avec les précédents dans la mesure où l'espace d'entrée est normalisé en [-1 ;1] alors que les valeurs se situent autour de [-100 ;100] pour les deux autres modèles. De ce fait, une petite perturbation tel que ε = 0.1 représente 10% de toute la dynamique. Des valeur de ε supérieures à 1 donnent une image entièrement bruitée ou la forme de l'objet n'est plus apparente. Ainsi, ses performances sur l'ensemble du jeu de données ont été présentées à part sur la **Figure 5**.

Etonnement, l'exemple de **Figure 3 C** montre que de InceptionV3 ne classe pas correctement l'image pour de ε = 0.05, lui attribuant la classe « rock beauty » (poisson ange) à seulement 7,2%, mais donne une réponse correcte pour des valeurs à partir de ε = 0,1 avec des probabilités qui augmentent jusqu'à 82.6% pour ε = 0,4. Ce surprenant résultat pourrait s'expliquer par le fait que, lorsque ε devient trop grand, beaucoup de pixels sont projetés à -1 ou 1 dans l'espace des caractéristiques, saturant le réseau. Les prédictions deviennent plus stables mais comme seulement certaines classes deviennent dominantes, les prédictions sont souvent fausses. Les bons résultats pour cette image lorsque ε relèvent plus de la chance que d'une réelle robustesse du modèle.

La **Figure 5** montre un autre résultat étonnant. Bien que les performances globales d'InceptionV3 soient grandement affectées par les perturbations, les résultats sont légèrement meilleurs pour ε = 0,1 que pour ε = 0,05 (Top-1 et Top-5 Accuracy respectivement à 25,80% et 53,66%, contre 27,43% et 54,80%). Cela pourrait s'expliquer par le fait que, quand ε prend certaine valeurs, la perturbation est suffisamment grande pour pénétrer jusqu'aux activations clé d'InceptionV3, mais suffisamment petite pour être partiellement brisée et devenir aléatoire, ce qui peut faire tomber l'image dans une classe plausible. Ce phénomène disparaît vite et les résultats se détériorent vite quand ε devient plus grand que 0,2 et que le bruit perturbe grandement le modèle.

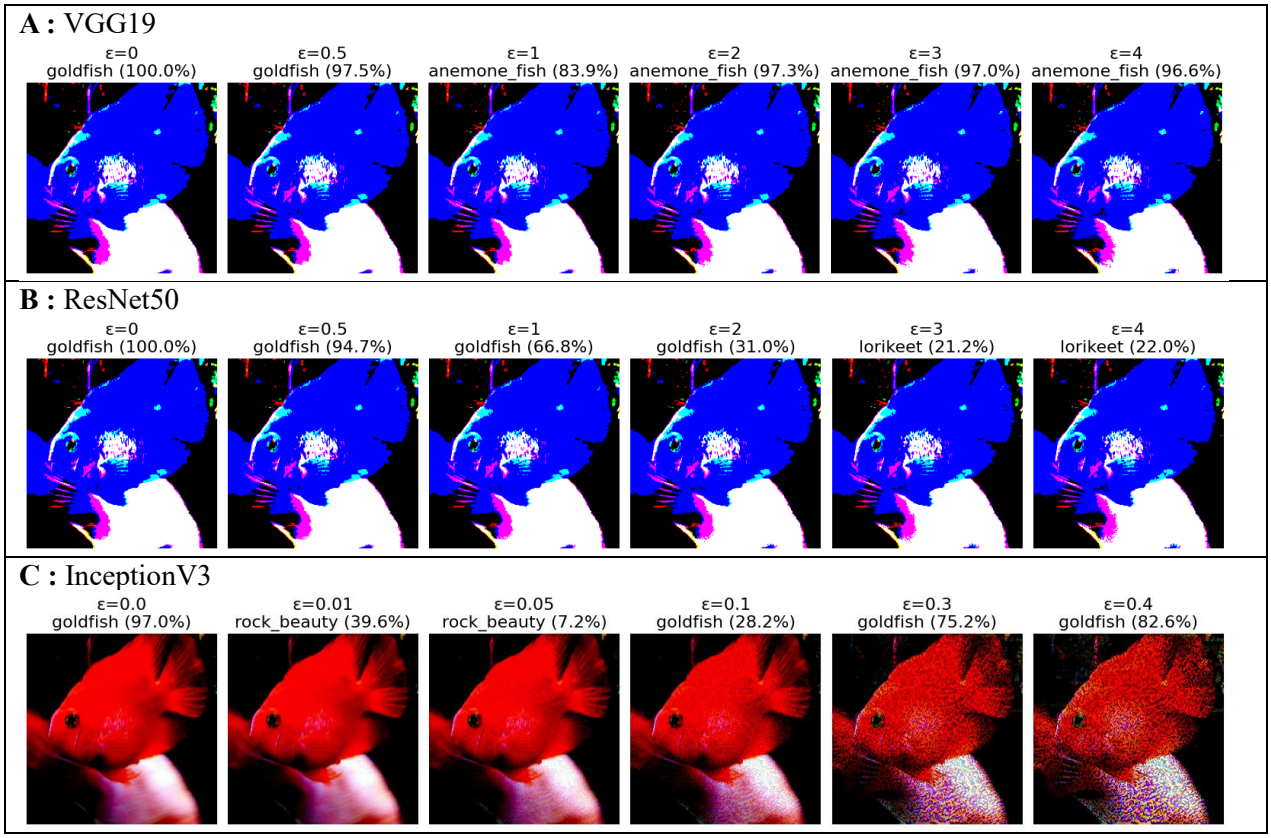


Figure 3 : Visualisation, pour chaque modèle, d'une attaque adverse FGSM. Au dessus de chaque image sont présentées la valeur de ε ainsi que la classe et la probabilité attribués par le modèle. Les images pour lesquelles ε = 0.0 correspondent aux images prétraitées initiales.

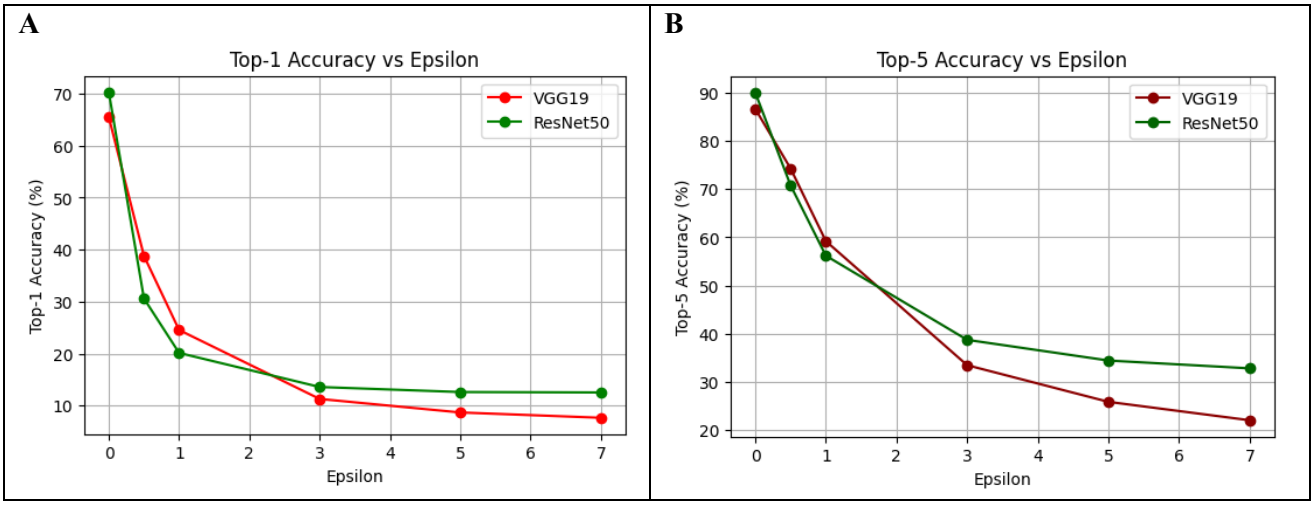


Figure 4 : Top-1 Accuracy (**A**) et Top-5 Accuracy (**B**) pour VGG19 et ResNet50 en fonction de ε sur tout jeu de données. Fonction de perte : *cross entropy*

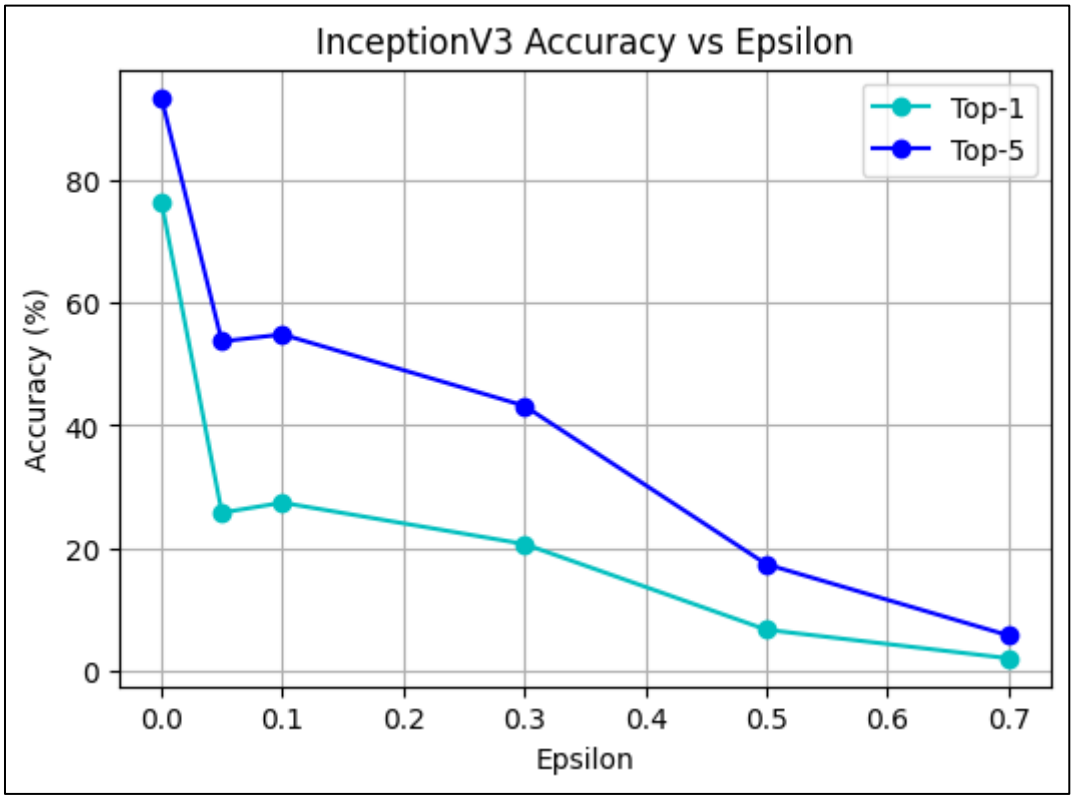


Figure 5 : Top-1 et Top-5 Accyracy pour InceptionV3 en fonction de ε sur tout jeu de données. Fonction de perte : *cross entropy*

II) Attaque FGSM ciblée

Il existe plusieurs variantes d'attaque FGSM. La plus commune est l'attaque ciblée. L'objectif ici n'est plus de perturber le modèle, mais de forcer la prédiction d'une classe précise. Ainsi la formule du filtre de l'attaque adverse peut s'écrire selon l'équation suivante :

$$advX = X - \varepsilon \cdot sign(\nabla_X J(\theta, X, Ycible)) \quad (2)$$

où les paramètres sont les même que pour (1) sauf :

- *Ycible* : Classe cible.

Les petites modifications apportées à l'image vont donc dans le sens opposé au gradient de la fonction de perte entre l'image et la classe que l'on souhaite prédire.

L'efficacité de ce type d'attaque dépend beaucoup des caractéristiques communes entre l'image d'entrée et les images types de la classe dont on veut forcer la prédiction. Pour l'image présentée en exemple, la classe « teddy » (nounours) est une classe cible efficace avec VGG19 (**Figure 6 A**), la classe « lorikeet » l'est avec ResNet50 (**Figure 7 A**) tandis que la classe « rock beauty » l'est avec InceptionV3 (**Figure 8 A**). En revanche, prendre une classe cible trop éloignée de celle de l'image d'entrée aura seulement pour effet de perturber le modèle de manière moins efficace qu'une attaque classique et n'aboutira pas à la prédiction espérée (**Figures 6 B**, **7 B** et **8 B**). En outre, pour que ce type d'attaque soit réellement efficace sur tout le jeu de données, il faudrait au moins une attaque spécifique par classe. Ce travail n'a pas été fait ici.

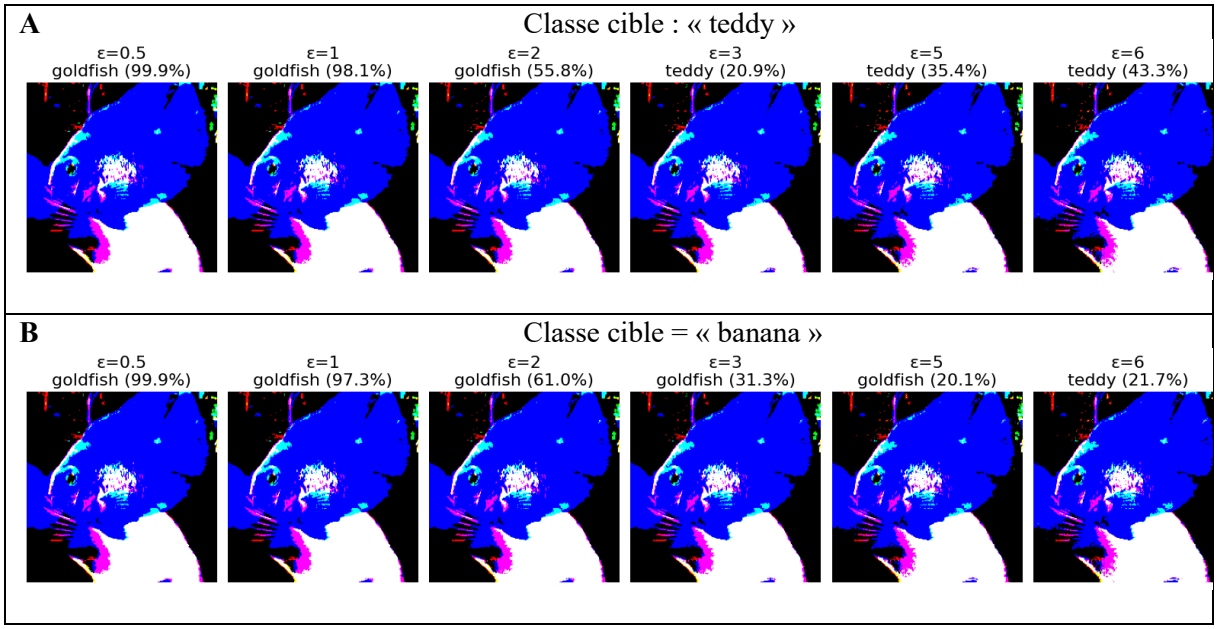
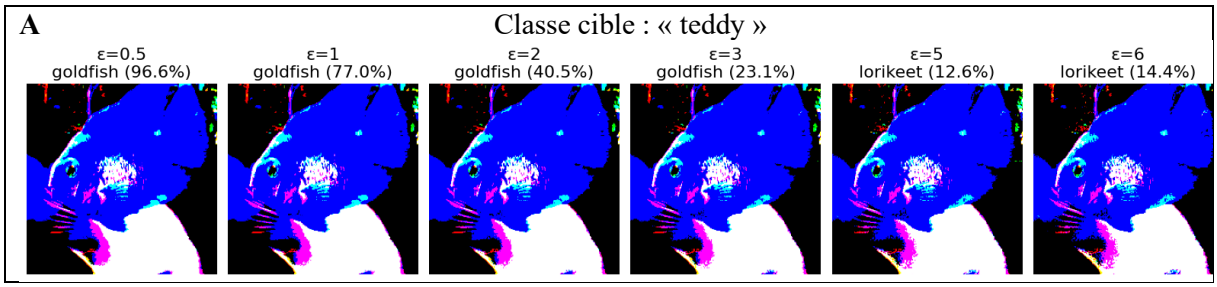


Figure 6 : Avec VGG19, exemples de classe cible efficace (**A**) et peu efficace (**B**) pour réaliser une attaque FGSM ciblée sur l'image prise en exemple. Fonction de perte : *cross entropy.*



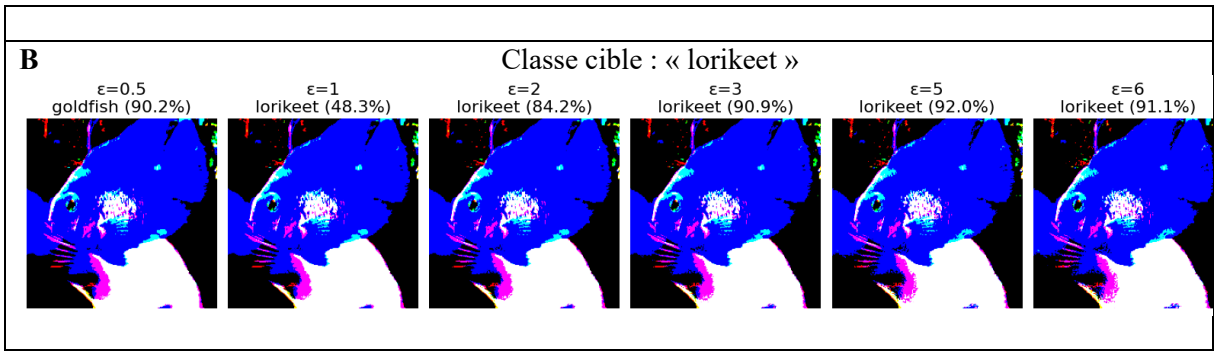


Figure 7 : Avec ResNet50, exemples de classe cible efficace (**A**) et peu efficace (**B**) pour réaliser une attaque FGSM ciblée sur l'image prise en exemple. Fonction de perte : *cross entropy.*

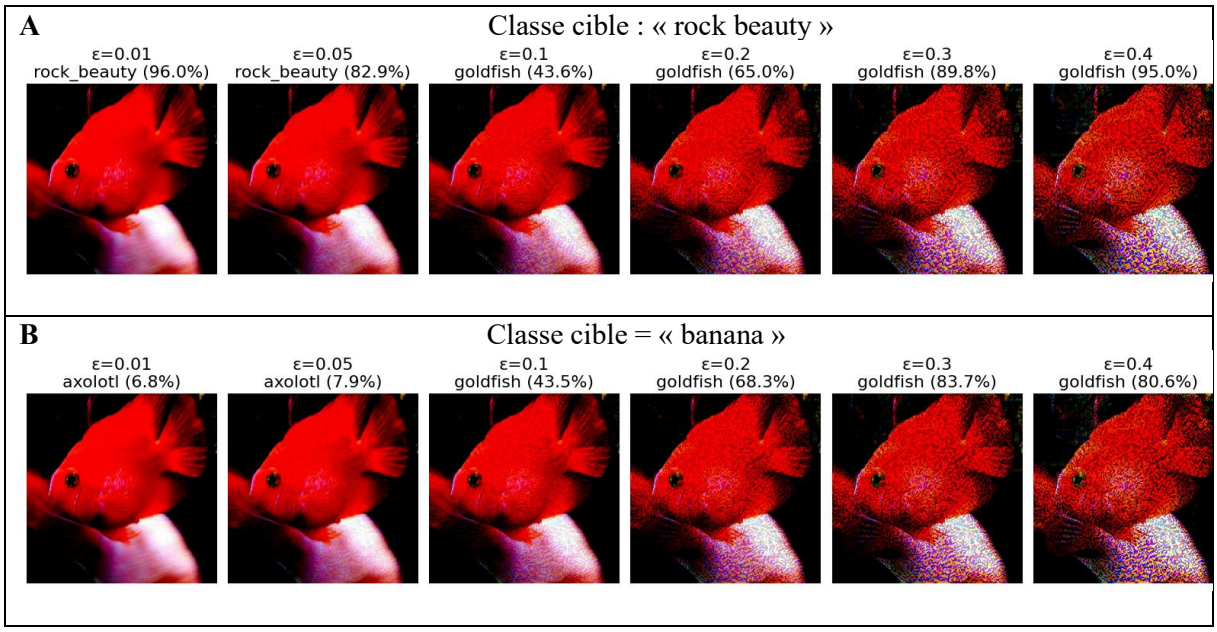


Figure 8 : Avec InceptionV3, exemples de classe cible efficace (**A**) et peu efficace (**B**) pour réaliser une attaque FGSM ciblée sur l'image prise en exemple. Fonction de perte : *cross entropy.*

III) Attaque FGSM itérative (I-FGSM)

Une FGSM itérative est une attaque où l'on applique plusieurs petites perturbations successives au lieu d'une seule grande comme vu précédemment. On a donc l'équation suivante :

$$X_{t+1} = Clip_{X,\varepsilon}(X_t + \alpha \cdot sign(\nabla_X J(\theta, X_t, Ypred)) \quad (3)$$

où

- t : itération
- X_t: Image à l'itération *t*.
- ε *:* Perturbation maximale autorisée
- $Clip_{X,\varepsilon}$:empêche la perturbation de sortir de $[-\varepsilon, \varepsilon]$
- α :pas de mise à jour $(0 < \alpha < 1)$
- J: Fonction de perte

- ∇_X : Gradient
- θ : Paramètres du modèle.
- $Ypred$: Classe d'entrée d'origine.

La taille de la perturbation peut augmenter avec le nombre d'itération, le pas de mise à jour et le cliping. Cette attaque est non ciblée car l'on suit le sens du gradient.

La I-FGSM a été testé sur plusieurs images du jeu de données en faisant varier plusieurs paramètres mais la prédiction n'a été altérée pour aucun des trois modèles (résultats non présentés ici). Il semblerait que pousser progressivement le gradient loin de la prédiction mais sans classe cible précise peut amener celui-ci dans une zone peu confiante ou plate, ce qui fait que la prédiction reste plus stable qu'avec une attaque FGSM non ciblée classique si l'unique pas est suffisamment grand pour perturber le réseau.

La I-FGSM peut être employée pour une attaque ciblée où la les petites perturbations vont dans le sens opposé au gradient entre l'image et une classe cible. L'équation devient :

$$X_{t+1} = Clip_{X,\varepsilon}(X_t - \alpha \cdot sign(\nabla_X J(\theta, X_t, Ycible)) \quad (4)$$

où les paramètres sont les même que pour (3) sauf :

- $Ycible$: Classe cible.

La I-FGSM ciblée s'est révélée extrêmement efficace sur le jeu de données. La **Figure 9** montre que, avec pour classe cible « banana » (banane), 10 itérations et un pas de 1/10, les trois modèles sont facilement dupés et renvoient tous « banana » à partir d'une certaine valeur de perturbation maximale autorisée ε, alors même que cette classe possède des caractéristiques bien différente de la classe « goldfish ». InceptionV3 est de loin le plus sensible (**Figure 9 C**) avec aucune bonne prédiction. ResNet50 se trompe à partir de ε = 1 et prédit la classe cible à plus de 90% à partir de ε = 2 et 100% pour ε = 3 (**Figure 9 B**). VGG19 est le plus résistant, se trompant à partir de ε = 2 et prédisant la classe cible à plus de 90% à partir de ε = 3 (**Figure 9 A**). La I-FGSM apporte des petites perturbations locales et nous avons vu précédemment que ce dernier modèle y était moins sensible que les deux autres.

Il est important de noter que la génération d'images modifiées par I-FGSM prend beaucoup plus de temps que par simple FGSM. A cause d'un manque de puissance de calcul, seul 100 classes sur les 1000 ont été aléatoirement sélectionnées pour l'évaluation globale des modèles présentée sur les **Figures 10** et **11**. Les résultats qui y sont présentés, donnent cependant un aperçu des performances des trois modèles en fonction de ε bien, bien qu'étant fortement dépendants de la sélection. La classe cible est encore « banana » pour toute les images.

On remarque sur la **Figure 10** que le taux de bonnes prédiction de VGG19 décroit surprenaient lentement avec des valeurs minimales de Top-1 et Top-5 Accuracy respectivement à 55,26% et 76,32% pour ε = 7. En comparaison, ces mêmes scores pour ResNet50 s'effondrent à 0% respectivement pour ε = 3 et ε = 5. Cette résistance de VGG19 à la I-FGSM n'est pas si surprenante car ce type d'attaque apporte de petites perturbations locales et nous avons vu précédemment que ce dernier modèle y était moins sensible que les deux autres.

Les scores obtenus par InceptionV3 présentés sur la **Figure 11** sont étonnement bons pour de petites valeurs de ε (Top-1 et Top-5 Accuracy respectivement à 81,58% et 97,37% pour ε = 0,1). Peut-être est-ce parce que pour les classes testées, les petites perturbations ne passent pas dans certaines couches clé du modèle. Pour des valeurs de ε entre 0,1 et 0,7 , la décroissance des scores semble régulière pour atteindre des valeurs de Top-1 et Top-5 Accuracy respectivement à 15,79% et 23,68% pour ε = 0,7.

La puissance d'une attaque I-FGSM ciblée vient du fait que le gradient est recalculé à chaque pas, ce qui fait que sa direction est recalculée en continu contrairement à une FGSM ciblée celle-ci est figée.

De plus, l'accumulation de petits pas permet à la perturbation de traverser plusieurs couches linéaires (fonction ReLU, BatchNormalisation, pooling…) alors qu'une FGSM ciblée ne peut en traverser qu'une seule.

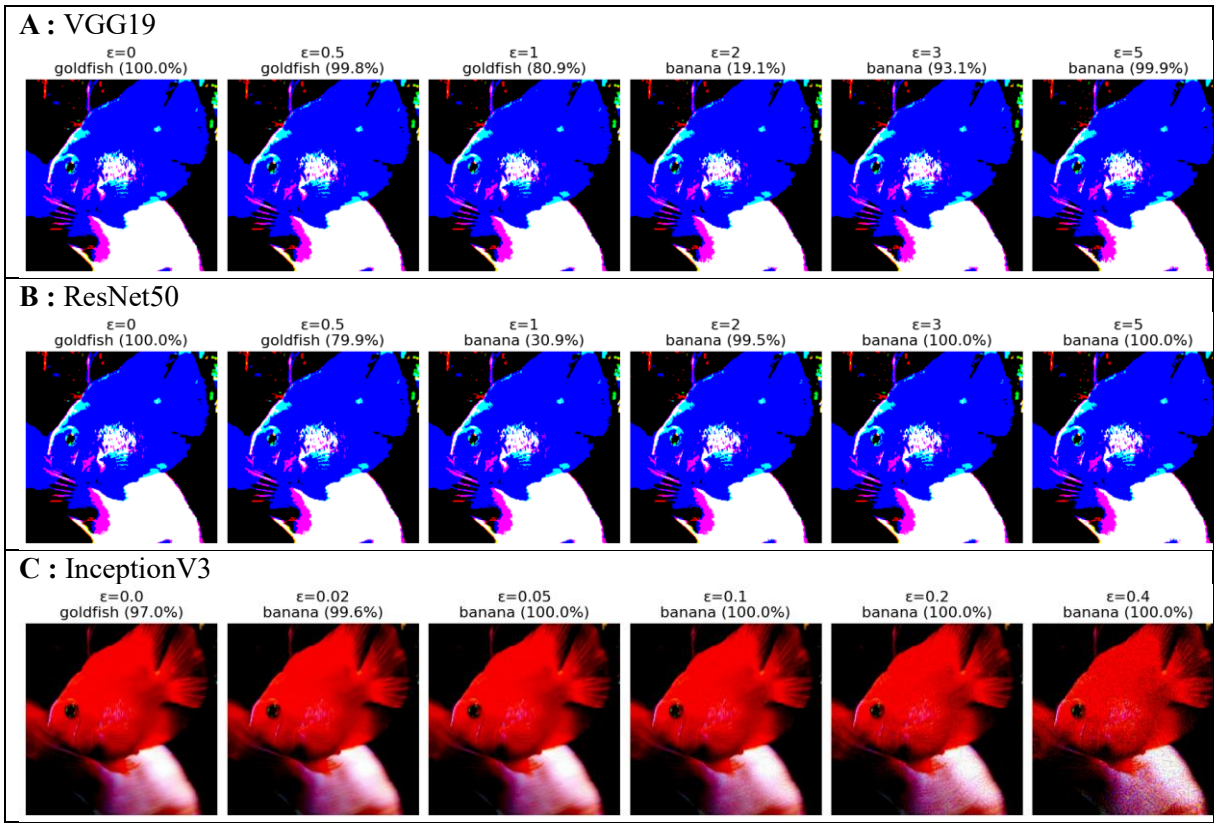


Figure 9 :Visualisation, pour chaque modèle, d'une attaque adverse i-FGSM ciblée avec pour classe cible : « banana ». Les classes et les probabilités attribués par le modèle pour différentes valeurs de perturbation maximale autorisée ε sont affichées au dessus des images. Pas d'apprentissage α=0,1. Fonction de perte : *cross entropy.*

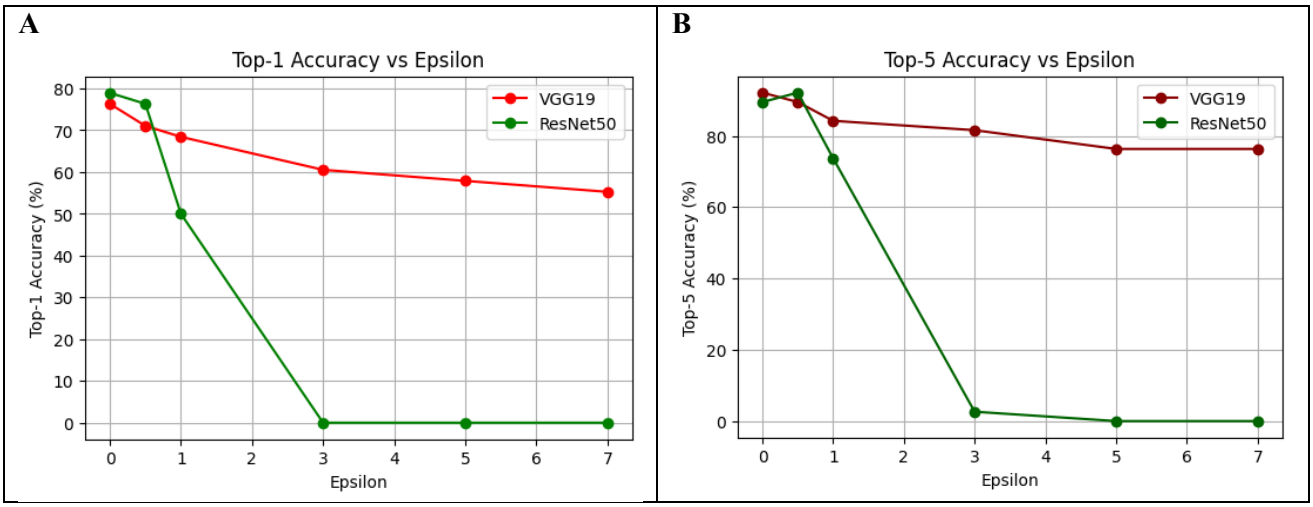


Figure 10 : Top-1 Accuracy (**A**) et Top-5 Accuracy (**B**) pour VGG19 et ResNet50 en fonction de ε sur toutes les images de 100 classes tirées aléatoirement. Pas d'apprentissage α=0,1. Fonction de perte : *cross entropy.*

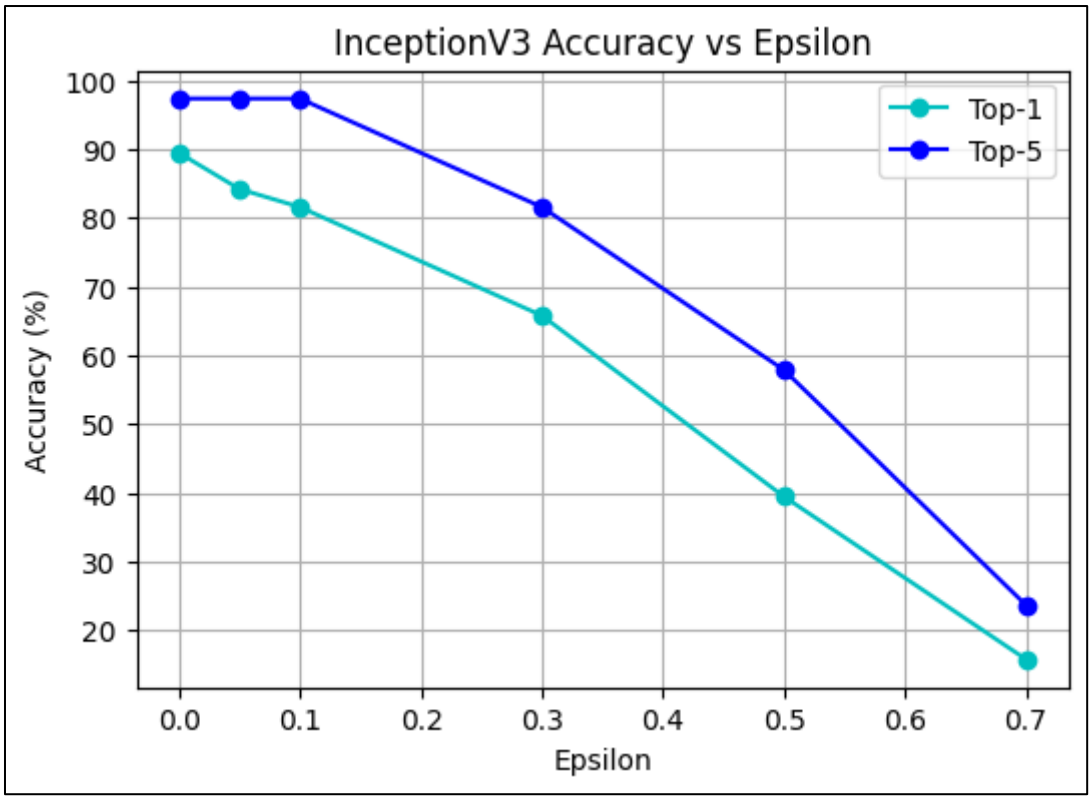


Figure 11 : Top-1 et Top-5 Accyracy pour InceptionV3 en fonction de ε sur toutes les images de 100 classes tirées aléatoirement. Pas d'apprentissage α=0,1. Fonction de perte : *cross entropy*

Discussion

Tous ces résultats montrent que les différents modèles réagissent aux attaques de manière bien différente.

Chez VGG19, le gradient passe par un unique chemin. La perturbation y est d'autant plus efficace que l'écart avec le gradient est grand. Cependant, la longueur du réseau et le nombre élevé de couches convolutives pourrait expliquer sa relative résistance face aux attaques itératives.

ResNet50 possède un réseau moins profond avec plusieurs chemins possibles pour le gradient grâce aux connections résiduelles. Pour les grande perturbations, il est le modèle le plus robuste, vraisemblablement grâce à ces dernières qui gardent une mémoire de forme. Il reste cependant très vulnérable aux attaques itératives ciblés où les petites mises à jours de la perturbation arrivent à duper ces connections résiduelles.

IneptionV3 doit être traité différemment des deux autres car beaucoup complexe avec un réseau vaste et plusieurs chemins en parallèle pour le gradient. Ses résultats sont parfois difficilement explicables et il est certain qu'ils puissent être très variables d'une image à l'autre ou d'une classe à l'autre. Le modèle est vulnérables aux fortes perturbations qui bruitent fortement l'image. De manière différente, les très petites perturbation de quelques centièmes de points d'écart sur le gradient l'affecte tout autant. Or ces dernière si les plus redoutables car imperceptibles à l'œil nu et probablement difficilement détectables par un système de pré-traitement.

Le choix des candidats a été fait pour essayer de tester au mieux les différentes architectures, mais d'autres modèles complexes pré-entrainés sur ImageNet auraient pu être testés : GoogLeNet, Xception, ShuffleNet…

De manière générale, les résultats présentés ici ont été grandement impactés par une limitation en puissance de calculs, le plus long étant la génération d'images piégées pour tout le jeu de données.

Les attaque adverses de type FGSM sont connues et bien documentées et il existe différents moyens de s'en protéger, à commencer par inclure des images modifiées par cette méthode dans l'apprentissage.

Il est documenté que les noyaux RBF (Radial Basis Function) y résiste bien [1]. Pour ce travail, un modèle hybride incluant ResNet50 pré-entrainé sur ImageNet suivit d'une couche RBF a été tenté mais n'a pas donnée de résultats concluants (ils sont cependant présentés à la fin du carnet Jupyter ci-joint).

Le point faible des attaques FGSM, comme toutes les autres attaque « en boîte blanche », est la nécessité de connaître exactement les paramètres du modèle que l'on veut tromper pour avoir un effet optimal. Dans la pratique, l'attaquant n'en a souvent aucune connaissance et ne peut qu'observer les entrées et les sorties. Les hackers utilisent souvent un « modèle de substitution » pour générer des exemples adversaires qui se transfèrent efficacement vers le système cible , une propriété connue sous le nom de transférabilité.

Sources :

[1] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[4] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[5] Sen, J. (2024, January). The FGSM attack on image classification models and distillation as its defense. In *International Conference on Advances in Distributed Computing and Machine Learning* (pp. 347-360). Singapore: Springer Nature Singapore.