Dauphine | PSL
UNIVERSITÉ PARIS

Rapport

Adversarial attacks in Deep Learning

Élèves :
Matteo Barbieri
Lea Amar
Hassan El mansouri Khoudari
Salma Elghourbal

Enseignant :
Benjamin NEGREVERGNE

20 décembre 2020

Table des matières

1 Introduction

Les travaux récents ont démontré que les réseaux de neurones profonds sont vulnérables à la perturbation d'exemples d'entrée qui sont presque impossible à distinguer des données naturelles et pourtant classées incorrectement par le réseau. En fait, certaines des dernières conclusions suggèrent que l'existence de ces attaques puissent être une faiblesse inhérente aux modèles d'apprentissage profond due à leur linéarité.

Pour résoudre ce problème, nous étudions la robustesse des réseaux de neurones en trois temps :

— Exploration des méthodes d'attaques (FGSM et PGD) pour mieux cerner les variables en jeu et mieux aborder les mécanismes de défense.
— Implémentation de défenses contre ces attaques adversariales et étude de leurs effets sur le jeu de données d'origine.
— Ouverture sur de nouvelles méthodes d'attaques et de défenses non usuelles.

2 Modèle de classification de base

Dans ce projet, nous travaillons sur la base de données CIFAR-10 contenant des images de taille (32,32,3) pouvant appartenir à 10 classes. Nous avons construit un réseau de neurones convolutionel pour la tâche de classification des images de CIFAR-10. L'architecture du réseau est la suivante : 3×(2 Conv2D + 1 Maxpool) + 2 Dense. Nous avons aussi utilisé du Dropout et la Batch normalization après les couches convolutives. Ainsi, ce modèle atteint une accuracy de 93% sur l'ensemble d'entraînement et de 85% sur l'ensemble de test.

3 Attaques adversariales

Dans cette section, nous présentons les différentes attaques adversariales que nous avons étudiées et implémentées.

3.1 Fast gradient sign method - FGSM

3.1.1 Rappel

Goodfellow et al.[2] ont proposé une attaque efficace et non ciblée pour générer des images adversariales. Cette attaque est appelée FGSM "Fast Gradient Sign Method". Elle effectue un seul pas dans la direction (i.e. le signe) du gradient de la fonction de coût par rapport à l'entrée dans le but d'augmenter la fonction de coût en exploitant la direction de la pente la plus forte (Voir la formulation de FGSM dans l'équation 1).

$$x' = x + \epsilon \cdot sign(\nabla_x(J(x, \theta, y))) \tag{1}$$

Le paramètre ϵ représente le rayon de la boule infinie toléré autour d'une image normale x. L'attaque FGSM fonctionne si l'hypothèse d'approximation linéaire de la fonction de coût au voisinage d'un point x est valable (c.f. [2]).

3.1.2 Expériences

Nous avons généré un ensemble d'images adversariales test à partir des images de test de CIFAR-10 et de la fonction de coût du modèle de base implémenté en 2. Ensuite, nous avons évalué ce dernier sur ces exemples adversariaux. Voici un tableau récapitulant les résultats pour $\epsilon = 0.01$ et $\epsilon = 0.03$ sur une échelle de 0 à 1 ce qui équivaut à des perturbations de 2.55 et 8 sur une échelle de 0 à 255 :

	Images test normales	attaques avec $\epsilon = 0.01$	attaques avec $\epsilon = 0.03$
Accuracy	85%	31%	8%

TABLE 1 – Performance du modèle de base sur l'ensemble de test sous attaque FGSM

Nous affichons dans la figure suivante quelques perturbations sur une image d'un camion de CIFAR-10. Les images adversariales sont bien imperceptibles à l'oeil humain.Le modèle de base prédit correctement la classe de l'image normale du camion avec une confiance élevée de 100%. En revanche, quand on augmente la perturbation (0.01 puis 0.02), on observe que la confiance de la prédiction diminue. Enfin, avec un epsilon de 0.03 le modèle se trompe et prédit "chat" au lieu de "camion" avec une confiance de 16.9%

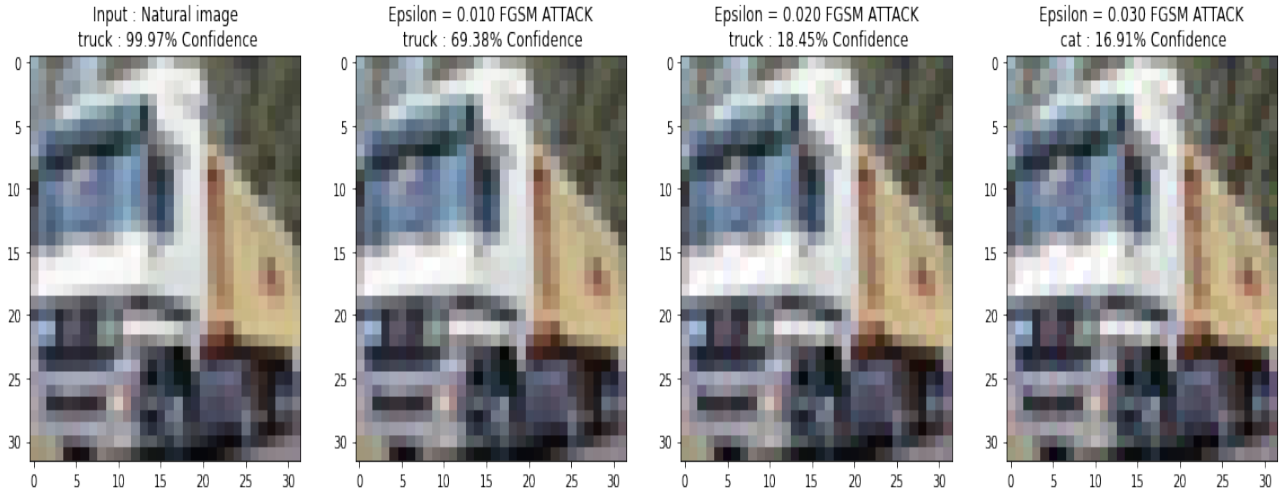

FIGURE 1 – Visualisations d'attaques avec les prédictions et confiances du modèle

Par la suite, nous avons exploré l'impact de ϵ sur l'accuracy du modèle de base. Ceci a abouti à la figure 2. Avec des valeurs de ϵ allant de 0.005 à 0.03, l'accuracy du modèle de base diminue de 80% jusqu'à atteindre 8%.

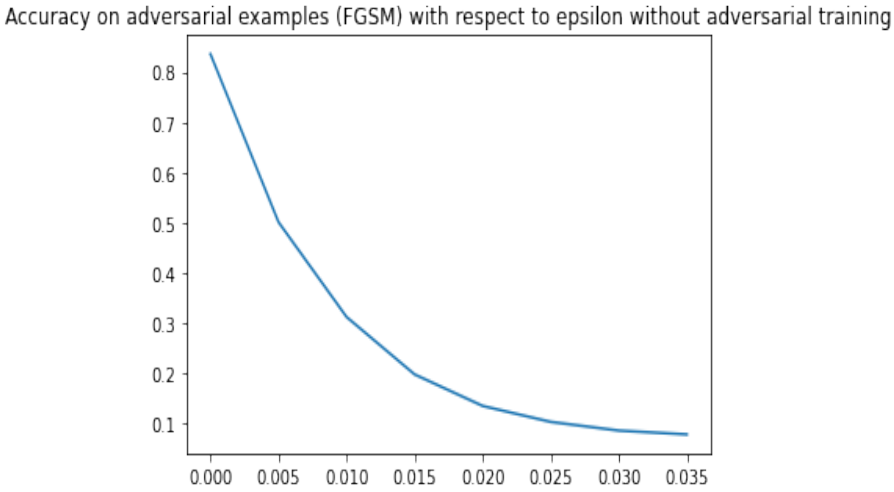

FIGURE 2 – Accuracy du modèle de base sur les attaques FGSM en fonction de epsilon

3.1.3 Apprentissage adversarial contre les attaques FGSM

Quand on veut construire des exemples afin de tromper un réseau, nous cherchons une solution du problème suivant :

$$\max_{||\delta||<\epsilon} \ell(h_\theta(x+\delta), y) \tag{2}$$

Où x est un exemple appartenant à la classe y, ℓ étant la fonction de coût, h_θ le modèle à paramètres θ, δ est une perturbation dont la norme doit être inférieure à ϵ
Soit un ensemble de paires d'entraînement S. Afin d'entraîner un modèle pour qu'il soit robuste contre une attaque, nous essayons de résoudre le problème d'optimisation suivant :

$$\min_\theta \frac{1}{|S|} \sum_{(x,y)\in S} \max_{||\delta||<\epsilon} \ell(h_\theta(x+\delta), y) \tag{3}$$

1- Méthode de base

Une stratégie simple et intuitive permettant d'approcher un h_θ qui soit robuste contre les attaques est de créer ces attaques et de les incorporer dans le processus d'entraînement.

Pour FGSM en particulier, à chaque "epoch" de l'entraînement nous générons un ensemble d'exemples FGSM à partir du gradient de la fonction de coût du modèle courant, nous entraînons le modèle sur ces exemples ainsi que sur des exemples normaux de la base d'origine, et nous répétons. Nous avons entraîné le modèle avec des exemples adversariaux générés avec $\epsilon = 0.03$

Nous résumons dans le tableau de la figure 4 les résultats de l'adversarial training avec des exemples FGSM sur CIFAR-10 :

	Images test normales	attaques avec $eps = 0.01$	attaques avec $eps = 0.03$
Accuracy	77%	33%	21%

TABLE 2 – Performance du modèle robuste sur une attaque FGSM

Par ailleurs, nous visualisons l'évolution de l'accuracy en fonction de ϵ dans la figure 3.

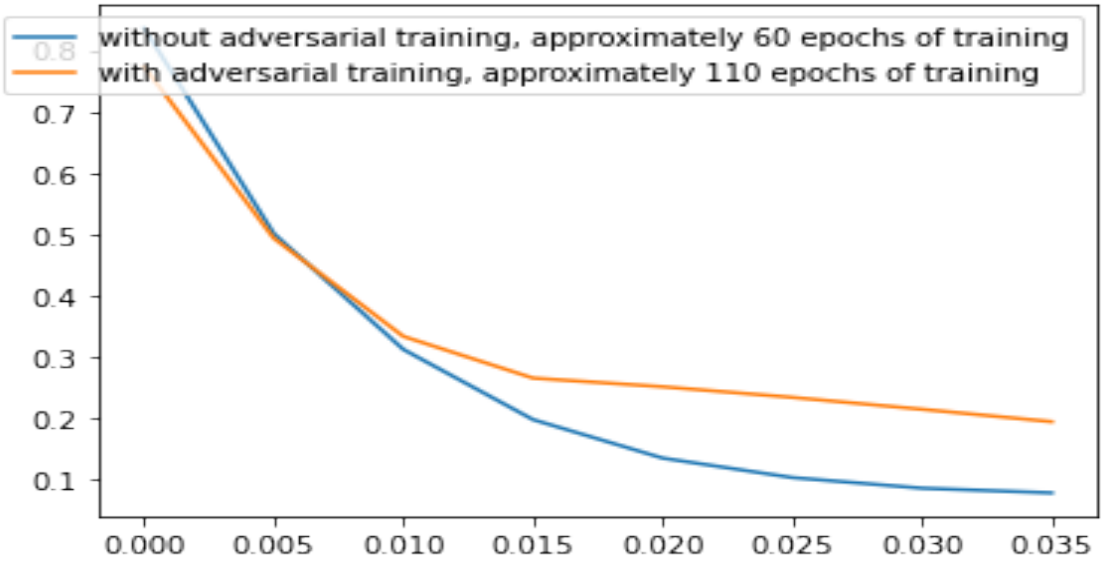

FIGURE 3 – Performances des modèles sur les attaques FGSM en fonction de epsilon

Le tableau 4 montre que le modèle "robuste" est un peu moins performant que le modèle de base sur les images test naturelles (77% contre 85% pour le modèle de base). La figure 3 illustre le fait que le modèle "robuste" devient plus performant que le modèle d'origine sur les attaques ayant une perturbation en norme infinie supérieure à 0.01 . Pour $epsilon = 0.03$, le modèle robuste a une accuracy de 21% contre seulement 8% pour notre CNN basique.

2- Une autre technique pour augmenter la robustesse d'un modèle

Des chercheurs de Facebook AI Research [3] ont proposé une technique pour rendre un modèle plus robuste aux attaques adversariales, cette technique est appelée "Feature Denoising for Improving Adversarial Robustness". Leurs travaux sur ImageNet et sur des réseaux de type ResNet montrent qu'en rajoutant des perturbations, même si très faibles, sur des images naturelles pour créer des exemples adversariaux, l'impact est le plus important sur les espaces latents. C'est à dire que les espaces latents d'une image adversariale passée dans un réseau sont très bruités par rapport à ceux d'une image normale. Ainsi, il est intéressant de supprimer le bruit de ces espaces latents en effectuant des opérations de débruitage comme est expliqué dans le papier de recherche. La méthode consiste à ajouter des blocs de débruitage à la sortie de chaque bloc résiduel du modèle. La figure 4 illustre l'architecture d'un "denoising block".

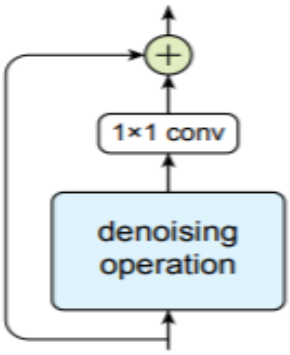

Figure 4. A generic denoising block. It wraps the denoising operation (*e.g.*, non-local means, bilateral, mean, median filters) with a 1×1 convolution and an identity skip connection [9].

FIGURE 4 – Architecture d'un bloc de débruitage, capture prise du papier de recherche ([3])

Nous avons implémenté un bloc de débruitage après la première couche convolutive du modèle CNN de base, puis nous avons effectué un adversarial training comme décrit dans la section 3.1.3. En choisissant des valeurs de perturbation très faibles dans le processus de génération des exemples adversariaux pour l'entraînement, le modèle de débruitage ne donne pas de résultats concluants. En revanche, il est plus robuste qu'un modèle sans bloc de débruitage face aux attaques de norme plus élevée , par exemple pour $\epsilon = 0.1$. La figure 5 représente une comparaison des performances d'un modèle (1 bloc de débruitage + adversarial training) et d'un modèle de base avec adversarial training.

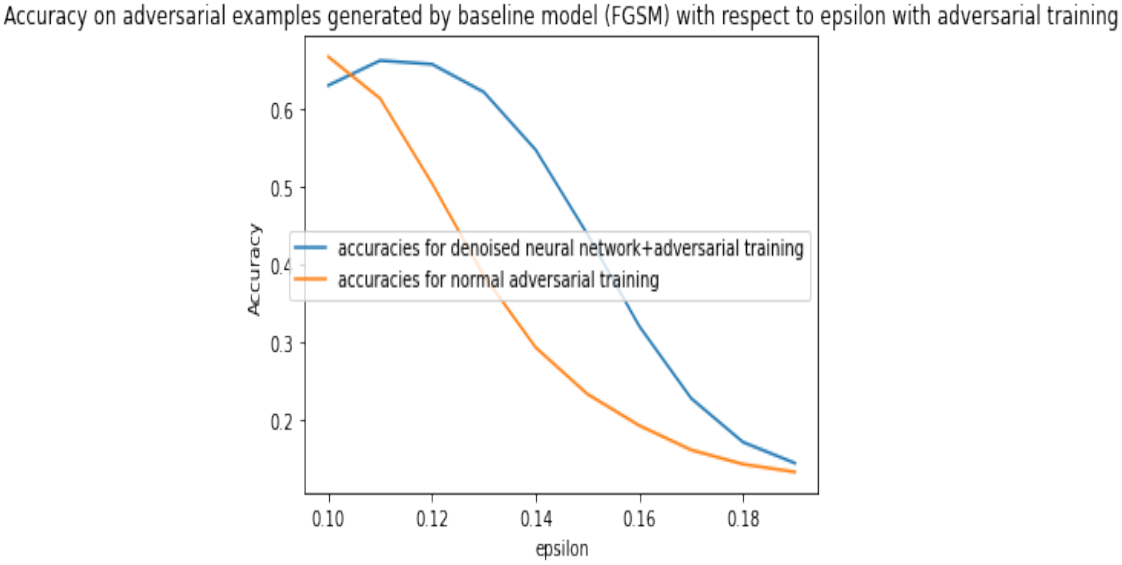

FIGURE 5 – Denoised model vs normal model

3.2 Projected Gradient Descent (PGD)

Le projected gradient descent est une extension du principe du FGSM dans le sens où l'on fait plusieurs pas (et non pas un seul) vers la direction de la maximisation du gradient de notre modèle.

Soit x un exemple qu'on veut attaquer, alors on définit la suite x_t selon la récursion suivante :

$$\begin{cases} x_0 = x \\ x_{t+1} = \Pi_{B(x,\delta)}[x_t + \varepsilon sgn(\nabla J(x_t, y, \theta))] \end{cases}$$

Où $\Pi_{B(x,\delta)}$ est la projecteur sur la boule de centre x et de rayon δ.

Le pas de chaque étape est représenté par ε. La projection sur la boule de centre x, et de rayon δ par rapport à la norme infinie s'assure que l'on ne dépasse pas x d'un certain seuil, ceci assure que le changement reste imperceptible à l'oeil humain.

3.2.1 Attaque et résultats :

Choix des paramètres :

Pour assener l'attaque la plus puissante à notre modèle, on commence par la recherche des hyperparamètres ε et δ optimaux.

On effectue donc le calcul de la précision du modèle sur un ensemble de validation pour des valeurs de ε allant de 0.005 à 0.001. Par ailleurs, comme dans la littérature on trouve que le δ devrait être au maximum égal à 0.03, on fait notre expérience avec des valeurs allant de à 0.01 à 0.03.

On obtient la heatmap suivante :

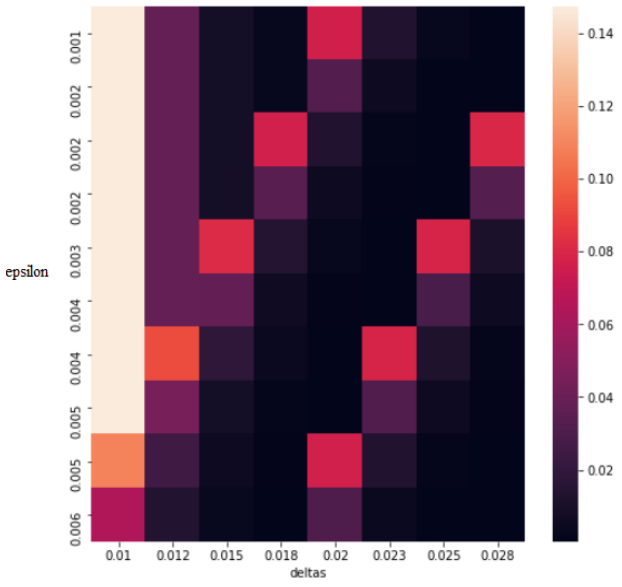

FIGURE 6 – Heatmap de la précision en validation en fonction des hyperparamètres ϵ et δ

En gardant en tête que l'on a droit à un $\delta \leq 0.03$, on choisit cette borne supérieure pour avoir plus de marge de manoeuvre avec un epsilon relativement faible. Avec de tels paramètres, couplés à un possibilité de faire plusieurs itérations de PGD, on a réussi à avoir de bons résultats qu'on présente dans la prochaine section.

Résultats :

La précision du modèle chute abruptement après seulement 3 itérations pour atteindre 3% alors que le modèle de base a une performance de 85%. Ensuite, si l'on continue avec davantage d'itérations, on obtient des valeurs avoisinant les 0%.

	Images test normales	Attaques avec $iter = 3$	Attaques avec $iter > 5$
Accuracy	83.58%	3.27%	<0.12%

TABLE 3 – Performance du modèle de base

Les performances de notre modèle contre des attaques de type PGD en fonction du nombre d'itérations sont mieux représentées dans le graphe suivant qui montre la nette réduction de précision en fonction du nombre d'itérations,

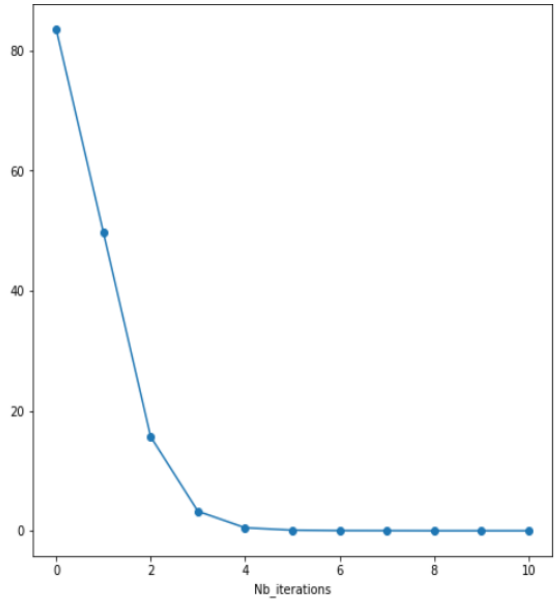

FIGURE 7 – Performance du modèle de base face à des attaques PGD en fonction du nombre d'itérations

3.2.2 Défense et résultats :

Dans cette section, on essaie de se défendre contre des attaques PGD. Pour ce faire, on utilise une fonction de perte modifiée :

$$\tilde{J}(x, y, \theta) = \alpha \cdot J(x, y, \theta) + (1 - \alpha) \cdot J(PGD(x), y, \theta)$$

Cette fonction va pondérer l'apprentissage de notre réseau, pour donner de l'importance aux exemples réels mais aussi les exemples attaqués. Pour les attaques, on utilise notre paramétrisation la plus puissante et donc $\alpha = 0.006$, $\delta = 0.03$ et $nb - iter = 10$.

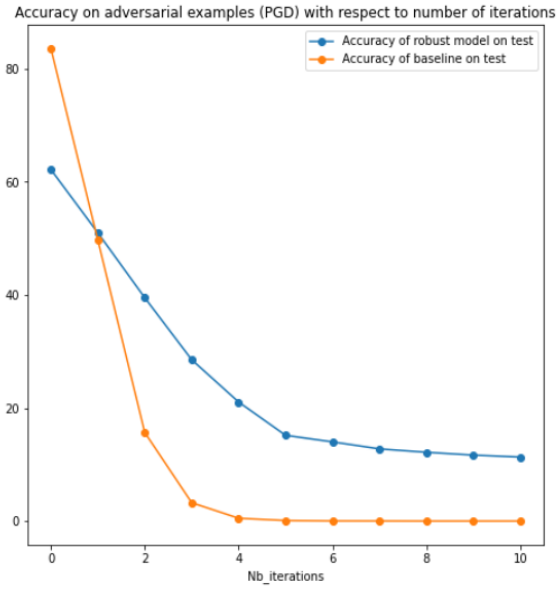

FIGURE 8 – Comparaison des performances du modèle de base et du modèle robuste

On présente également le tableau suivant où on résume les résultats :

	Images test normales	Attaques avec $iter = 5$	Attaques FGSM $\epsilon = 0.03$
Accuracy	62%	15%	30%

TABLE 4 – Performance du modèle robuste

Remarques et conclusions :
- La précision sur les images originales passe de 80% à 60%, ce qui est normal dans ce cas (nous n'avons pas pu entraîner le modèle aussi longtemps que le modèle de base, donc la différence pourrait en découler)
- Le modèle robuste fonctionne mieux sur les images attaquées que son homologue régulier stagnant à environ 15% pour les itérations$>$ 5, tandis que le modèle de base chute à 0% en précision.
- Le modèle robuste se défend mieux contre les attaques FGSM avec une précision de 30% contre 21% pour la défense utilisant FGSM.

3.3 Generetive Adversarial Networks

Dans cette partie, nous essayons de de proposer une méthode qui ne requiert pas d'avoir accès au gradient de la fonction de coût du modèle de départ pour produire les attaques. Nous nous tournons pour cela vers les GANs.

3.3.1 Principe de fonctionnement

Comme l'expliquent Xiao *et al.* [1], l'idée de fonctionnement est de construire deux réseaux : un générateur et un discriminant. Le générateur est en charge de produire les perturbations qui seront ajoutées aux images de la base de données avec l'objectif de tromper le réseau de neurone original. En face, le discriminateur a la fonction d'apprendre

à distinguer les images réélles de simages adversariales, puis de pénaliser le générateur lorsque les images qu'il produit sont visuellement trop différentes des images réélles.

Plus précisément, la fonction de perte du générateur est constituée de deux termes. Le premier terme, qui provient de l'évaluation du résultat par le discriminant $\mathcal{L}_D$, se rapporte à sa capacité à produire des images réalistes, tandis que le second $\mathcal{L}_{adv}$, provenant de l'analyse des erreurs du modèle de base, permet d'établir si la perturbation proposée permet effectivement d'induire le modèle de base en erreur. L'architecture générale du modèle peut être vue sur le schéma suivant (figure 9).

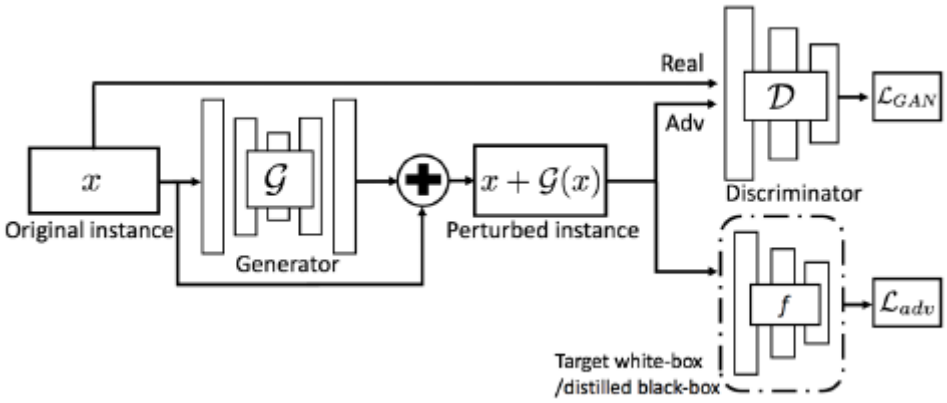

FIGURE 9 – Schéma représentant le fonctionnement du réseau AdvGAN.

Ainsi, nous avons pensé qu'il serait intéressant d'analyser la manière dont il est le plus prudent de procéder dans la phase d'aggrégation des différentes fonctions de perte du modèle générateur. Nous avons alors proposé de modifer légèrement la fonction de perte du générateur en pondérant les deux pertes citées précédemment.

$$\mathcal{L}_G = \lambda_D \mathcal{L}_D + \lambda_{adv} \mathcal{L}_{adv}$$

Les heatmaps suivantes montrent alors l'évolution de la valeur de la fonction de perte et de la précision du modèle de base ainsi que la taille de la perturbation. Pour ce faire, nous avons entraîné le GAN sur une portion de l'ensemble de test et supervisé l'entraînaement en surveillant l'évolution de sa performance à tromper le réseau de base sur un ensemble de validation indépendant. Par la suite, nous avons repris le code présent à cette adresse (`https://github.com/mathcbc/advGAN_pytorch`), afin d'étudier l'effet des paramètres λ_{adv} et λ_D. Nous rapportons dans la figure 10 les résultats obtenus sur l'ensemble de test.

On remar

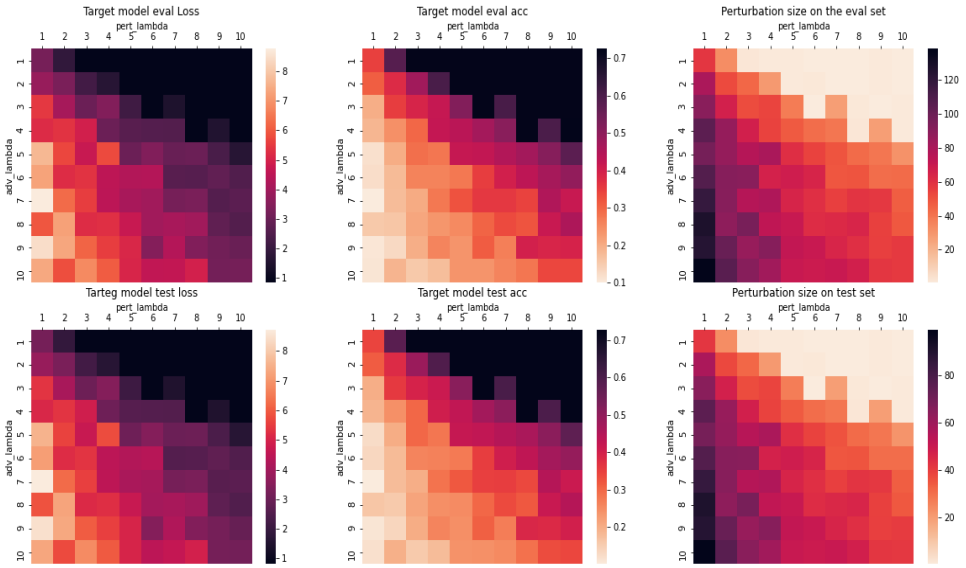

FIGURE 10 – Comparaison des performances du modèle de base et de la taille de la perturbation en fonction des poids associés aux pertes de perturbation et de classification.

4 Conclusion

Nos résultats prouvent que les réseaux de neurones profonds peuvent être rendus plus ou moins résistants aux attaques adversariales mais non sans effet négatif sur la performance pour les données d'origine. Effectuer l'apprentissage en résolvant le problème d'optimisation pondéré entre la fonction de perte initiale et la fonction de perte sur les attaques, reste un moyen fiable de rendre les réseaux plus robustes à de telles attaques.

Les résultats obtenus ne sont pas parfaits et nécessitent certainement plus de raffinement, on peut aussi se poser la question de la dépendance par rapport à la base de donnée choisie et si on pourrait espérer obtenir de meilleurs résultats avec des données qui ne sont pas aussi floutés que celles de CIFAR10. Ceci aurait par exemple pu rendre les blocs de denoising utilisés présentés en section 3.1.3 beaucoup plus impactant.

Références

[1] J-Y. Zhu W. He M. Liu D. Song C. Xiao, B. Li. Generating adversarial examples with adversarial networks, 2018.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[3] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.