

# **robust-neural-nets-476f6c646f72616b-42**

*Joachim DUBLINÉAU & Elie KADOCHE & Thomas PETITEAU*

## **1 Présentation générale**

### **Les réseaux de neurones artificiels, un outil puissant**

Les réseaux de neurones artificiels sont aujourd’hui des outils phares de l’intelligence artificielle. Ils nous permettent de résoudre des problèmes qui étaient jusqu’alors difficiles (reconnaissance d’image, de visage, compréhension de texte, voiture autonome, etc.). Ils sont utilisés par de très nombreuses entreprises et états pour réaliser des tâches plus ou moins importantes.

### **Les réseaux de neurones artificiels, un outil vulnérable**

Il s’avère malheureusement que ces outils sont aussi puissants que vulnérables. En effet, de nombreuses équipes de recherche ont démontré les nombreuses attaques qu’il est possible d’effectuer sur des réseaux de neurones artificiels. L’article le plus connu est celui de Ian J. GOODFELLOW, Jonathon Shlens et Christian SZEGEDY et date de mars 2015. Dans ce papier, il est montré qu’une petite modification d’une image (non reconnaissable à l’œil nu) peut complètement tromper un modèle l’identifiant.

### **Des enjeux importants**

La vulnérabilité de ces outils représentent un enjeu capital dans de nombreux domaines. Par exemple, si dans un aéroport, il est possible de tromper une machine chargée d’identifier les voyageurs et leur passeport, ou bien dans le cadre d’une voiture autonome, s’il est possible de faire reconnaître au système une plante au lieu d’un piéton, et le tout, avec des petites modifications facilement réalisables, soit de l’environnement, soit du réseau, les problèmes que cela peut engendrer semblent assez évidents. La protection des réseaux de neurones artificiels et le développement d’outils robustes est donc un enjeu capital.

### **Deux types d’attaque**

Il existe classiquement deux types d’attaque. Les *white box attack* correspondent aux attaques qui ont accès aux paramètres du réseau, et qui peuvent donc réaliser des attaques précises, dépendant du réseau (repousser les frontières de décision par exemple). Les *black box attack* correspondent aux attaques qui n’ont pas accès au réseau, et qui le voit donc comme une "boîte noire".

### **Notre travail**

Dans le cadre de ce projet nous nous proposons donc d’implémenter et de tester deux types d’attaques, FGSM (Fast Gradient Signed Method) et PGD (Projected Gradient Descent). Les attaques sont réalisées sur un réseau simple sur la base de données CIFAR 10. Nous proposerons enfin des méthodes de protection contre FGSM.

## 2 Les techniques d'attaques

### Fast Gradient Signed Method (FGSM)

La méthode FGSM a été l'une des premières attaques de réseaux de neurones décrites. Il s'agit d'une attaque white box, autrement dit, elle ne fonctionne qu'avec une connaissance complète du réseau à attaquer. Elle consiste à calculer le bruit à partir du gradient de la loss par rapport à l'image en entrée. On peut ainsi définir  $\bar{x}$ , l'image antagoniste de  $x$  telle que :

$$\bar{x} = x + \epsilon * sign(\nabla_x L(\theta, x, y))$$

Le gradient indiquant la plus forte pente, ajouter le gradient à  $x$  aura tendance à le faire passer à travers la frontière de décision séparant sa classe d'une autre. En prenant le signe de ce dernier, la perturbation aura tendance à translater l'image sur une plus grande distance.

Cette méthode n'apporte cependant aucune garantie de fonctionnement, la perturbation est calculée sans procédure itérative pour assurer que l'image perturbée soit classée différemment par le réseau. Cependant, son efficacité est tout à fait surprenante.

Considérant un réseau entraîné sur CIFAR10 qui dépasse les 70% de bonnes prédictions sur les données de test dont les courbes d'entraînement sont présentées en figure 1, on peut facilement diviser par 10 sa précision avec un  $\epsilon$  égal à 0.025. La perturbation est alors très faible visuellement.

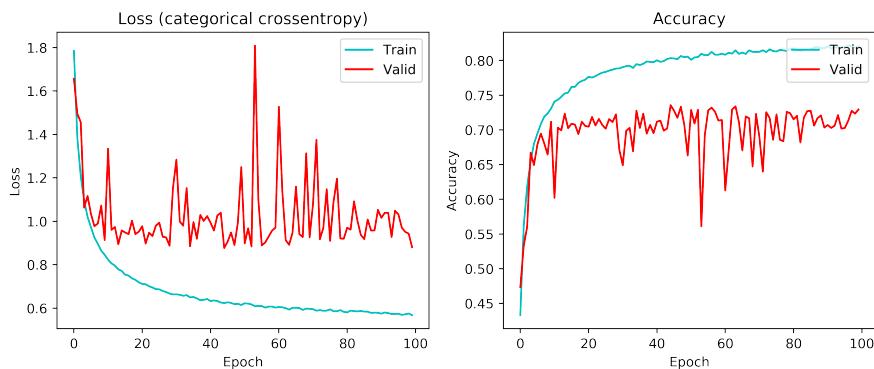


Fig. 1: Courbe d'entraînement d'un réseau de neurones respectant les contraintes du projet sur CIFAR10

Nos expériences montrent que FGSM, malgré sa simplicité apparente, est une arme de destruction massive sur un réseau.

### Projected Gradient Descent (PGD)

L'attaque Projected Gradient Descent ou PGD consiste en quelques sortes à itérer le FGSM. Voici le pseudo-code utilisé. L'algorithme ne fonctionne pas avec un nombre d'itération mais avec convergence à un seuil  $s$ . On dénomme  $I$  l'ensemble des matrices de même dimension que les matrices attaquées,  $\eta$  le pas d'itération et  $\epsilon$  la norme maximale de la perturbation.

L'algorithme précédent a été optimisé pour pouvoir calculer le plus rapidement possible les attaques en vu de pouvoir faire de l'entraînement adversarial. Nous arrivons ainsi à un temps de calcul très raisonnable de 1s par image sur CPU i5 7200. Afin de gagner du temps, l'algorithme ne va attaquer que les images bien prédites par le modèle.

#### Performances :

L'attaque PGD fait diminuer la précision du réseau de manière assez drastique. On voit ainsi sur la figure 6 que pour une perturbation  $\epsilon = 1$ , quasiment invisible à l'oeil nu (cf. Figure 5), on a réussi à diminuer de plus de 30% l'accuracy du réseau avec la norme  $l_2$ .

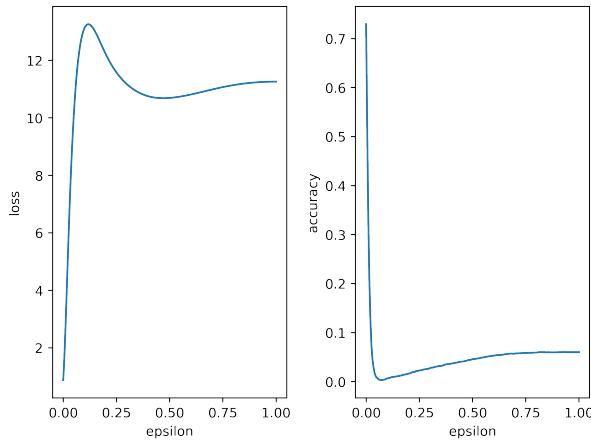


Fig. 2: Graphique présentant l'évolution de la précision et de la loss du réseau présenté en figure 1 en fonction de epsilon sur le jeu de test perturbée par une attaque FGSM

```

1  $d \leftarrow s + 1;$ 
2  $g \leftarrow \text{sign}(\nabla L(\theta, x, y));$ 
3  $a \leftarrow \text{random\_attack};$ 
4 tant que  $d > s$  faire
5    $a_{\text{prev}} \leftarrow a;$ 
6    $a \leftarrow a + \eta \times g;$ 
7    $a = \Pi_{B(x,\epsilon)}(a);$ 
8    $d = \|a - a_{\text{prev}}\|;$ 
9 fin
10  $a;$ 
```

Nous avons également tester les performances de notre réseau sous des attaques avec une norme infinie. Les résultats obtenus sont présentés en figure 7.

### One-Pixel Attack

Nous avons également implémenter l'attaque One Pixel qui consiste simplement à mettre un pixel à 0. Cette technique d'attaque est dite white box. Elle est beaucoup moins efficace et ne fait que peu baisser l'accuracy. Nous avons obtenu ainsi, une baisse de l'accuracy sur un échantillon de 500 images de seulement 2%. Cela peut s'expliquer par le fait que le modèle utilise des convolution avec un kernel de taille 3x3 qui vient atténuer l'impact d'un pixel sur le réseau. Pour améliorer l'impact de cette attaque, il faudrait utiliser une multi-pixel attaque.

## 3 Les techniques de défense

### Les réseaux antagonistes génératifs

Les réseaux antagonistes génératifs ou GAN (pour *Generative Adversarial Networks*) fonctionnent ainsi : un réseau créateur A a pour but de créer des images à partir de vecteurs aléatoires et un réseau discriminant B a pour objectif de dire si une image vient du véritable ensemble de données ou si elle a été créée par le réseau A (si l'image est artificielle). Les deux réseaux A et B s'entraînent ensemble, s'améliorant progressivement l'un et l'autre.

Nous avons eu l'intuition qu'en entraînant simplement un GAN, on aurait donc un réseau discriminant capable de dire, pour n'importe quelle image, si elle a été artificiellement créée ou pas. Naturellement, nous nous sommes dit que combiner un tel réseau avec un autre classifiant les images, pourrait donner un modèle global résistant aux attaques : si l'image n'est pas modifiée, prédire normalement, et si l'image a été modifiée, faire autre chose.

En pratique, nous n'avons pas eu de résultats satisfaisants avec cette méthode : une image légèrement modifiée peut beaucoup tromper le réseau et n'est pas identifiée comme une image artificielle. Il faudrait donc exploiter d'autres types de GAN, plus spécifiques contre des attaques précises.

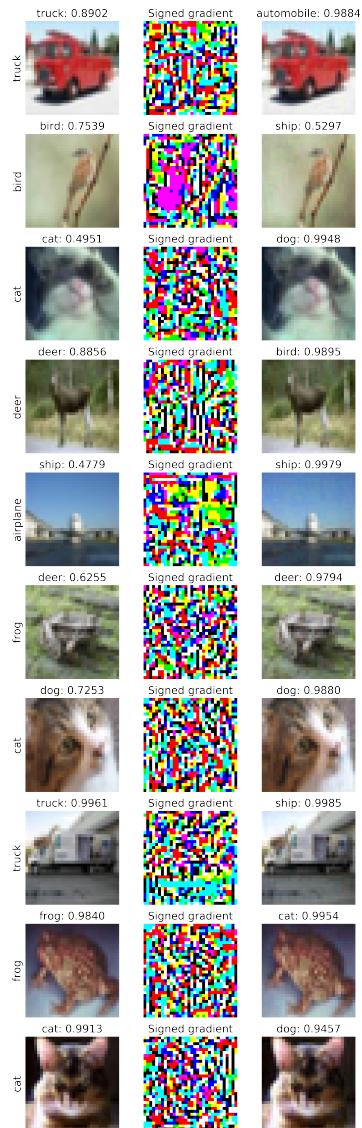


Fig. 3: Exemples d'images utilisées en figure 2. Sur chaque ligne est présentée l'image originelle, la perturbation et l'image perturbée. Pour chaque image, on retrouve son étiquette à gauche et la prédiction du réseau au-dessus.

Fig. 4: Illustration de l'attaque PGD

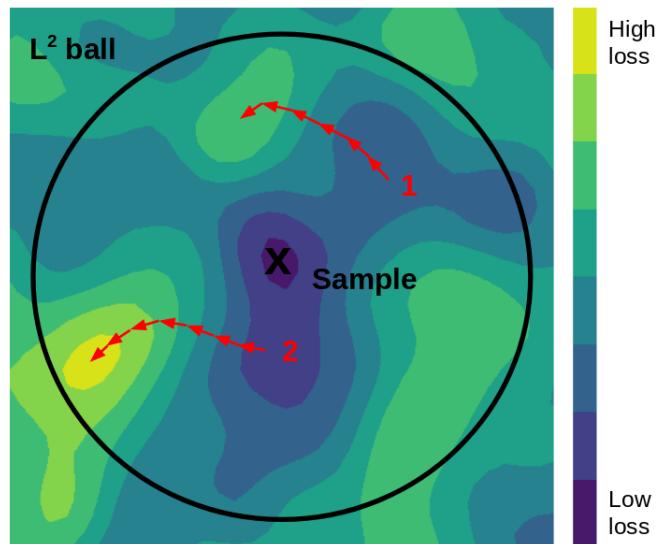
Fig. 5: Évolution des images avec epsilon en  $l_2$ 

Image source vs Image attaquée (eps=1)

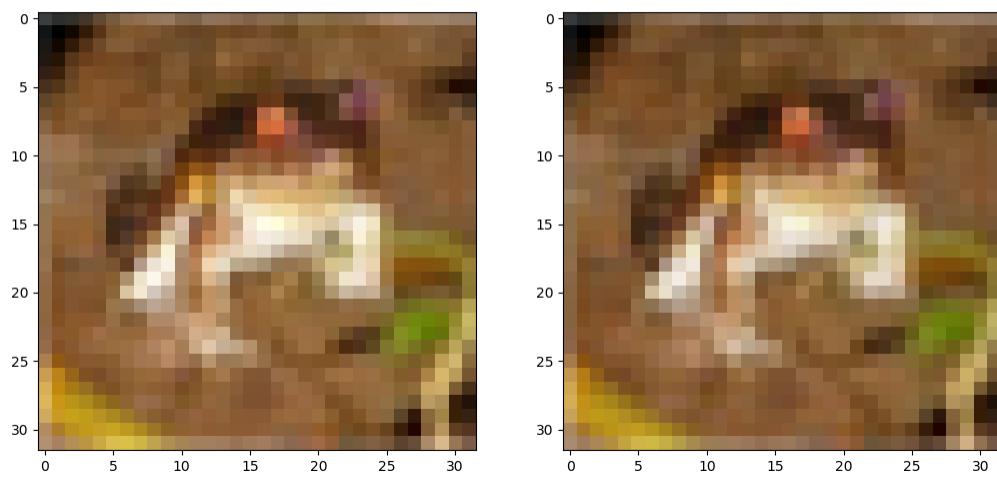


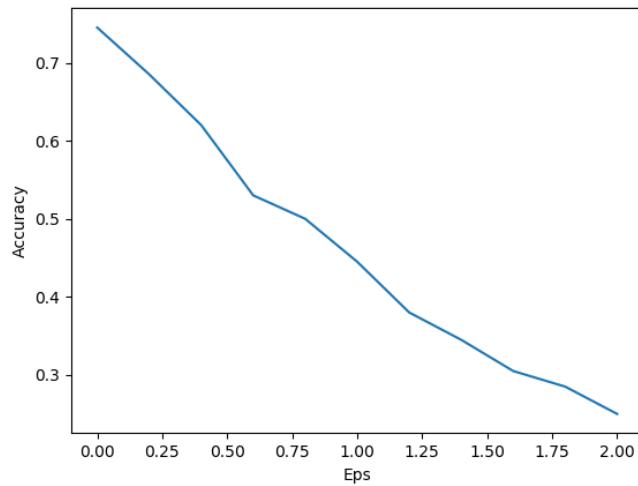
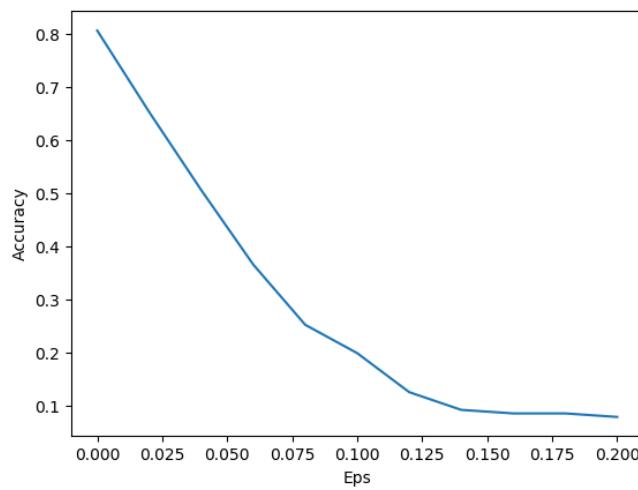
Fig. 6: Performances attaque PGD en  $\|2$ 

Fig. 7: Performances attaque PGD en norme infinie



## Défense FGSM

Nous avons tenté de trouver une stratégie pour parer aux attaques FGSM. Nous avons d'abord expérimenter rapidement plusieurs approches :

- Entraîner un réseau seulement sur des images perturbées.
- Entraîner un réseau préalablement entraîné sur des images perturbées.
- Intégrer des images perturbées dans le jeu de données d'entraînement à chaque epoch.

Ces différentes pistes se sont révélées peu fructueuses et nous les avons laissées de côté. Nous avons cherchés à rentrer les images perturbées dans l'entraînement pour forcer le réseau à les considérer et s'entraîner à les classifier.

Une façon simple de les inclure est de les faire entrer en considération dans la loss servant à l'entraînement. On peut alors sommer la loss sur l'image et celle sur l'image perturbée pour constituer la loss totale. On ajoute un paramètre  $\alpha$  permettant de maîtriser la part attribuée à chacune des loss dans la loss totale.

$$L(\theta, x, y) = \alpha \times l(\theta, x, y) + (1 - \alpha) \times l(\theta, x + \epsilon * sign(\nabla_x l(x, y)), y)$$

Nous avons ensuite conduit un certain nombre d'expériences pour essayer différentes hyperparamètres. Nous avons joué sur trois paramètres de l'apprentissage

- *alpha* : nous l'avons fait varié entre trois valeurs : 0.25, 0.5 et 0.75. Ces valeurs sont choisies de manière purement arbitraire.
- *epsilon* : égale à 0.025 et 0.07. La première valeur donne des images très peu perturbées et nous la considérons donc comme faibles. La seconde valeur est tirée de nos expériences sur FGSM, il s'agit en effet de la valeur minimisant la précision de notre réseau en figure 2.
- croissance de  $\epsilon$  : Suite à nos premières expérimentations sur cette stratégie, nous avons eu idée de faire croître  $\epsilon$  au cours de l'entraînement. Nous avons fixé 0.005 comme valeur ajoutée à chaque epoch dans nos expériences utilisant ce paramètre.

Les différents résultats sont présentées en figure 8. Une première constatation qu'on peut faire est que cet entraînement entraîne une perte de précision sur tous les hyperparamètres testés. Cette perte est accentuée avec la croissance de  $\epsilon$ . Cependant, sur toutes les courbes, on observe une résistance accrue face aux attaques FGSM avec les réseaux (d) présentant la meilleure résistance. Ces derniers sont intéressants puisqu'ils développent une résistance aux plus fortes perturbations mais restent plus faible aux petites perturbations. Le fait d'augmenter le paramètre  $\epsilon$  au cours de l'entraînement apporte une contrebalance à ce phénomène et réduit le creux observé dans les courbes des réseaux (c).

### 3.0.1 S'assurer que le code fonctionne

A la vue de nos premiers résultats sur les défense FGSM, nous avons soupçonné l'existence d'un bug dans notre code. Afin de vérifier cette hypothèse, nous avons comparer un réseau entraîné simplement et un réseau entraîné avec un paramètre *a* égal à 1. La figure 9 présente les courbes d'entraînements de ces deux réseaux. Malgré un nombre d'epoch différent des deux côtés, les courbes sont suffisamment similaires pour nous assurer que les méthodes d'entraînements sont proches. La figure 10 montre finalement qu'une attaque FGSM réalisées par un code indépendant produit deux courbes très similaires dont la différence est mise sur le dos d'une non égalité parfaite des deux réseaux.

En conclusion, nous avons réussi avec cet entraînement particulier à endurcir le modèle fixé dans les modalités du projet face aux attaques FGSM au prix d'une perte allant de 0 à près de 20% sur le taux de bonnes réponses sur le jeu de test. L'entraînement avec une faible perturbation (réseaux (b)) permet de passer l'epsilon minimisant la précision du réseau de 0.07 à 0.25 qui engendre des images bien trop perturbées pour qu'on attende d'un réseau qu'il sache les classifier correctement.

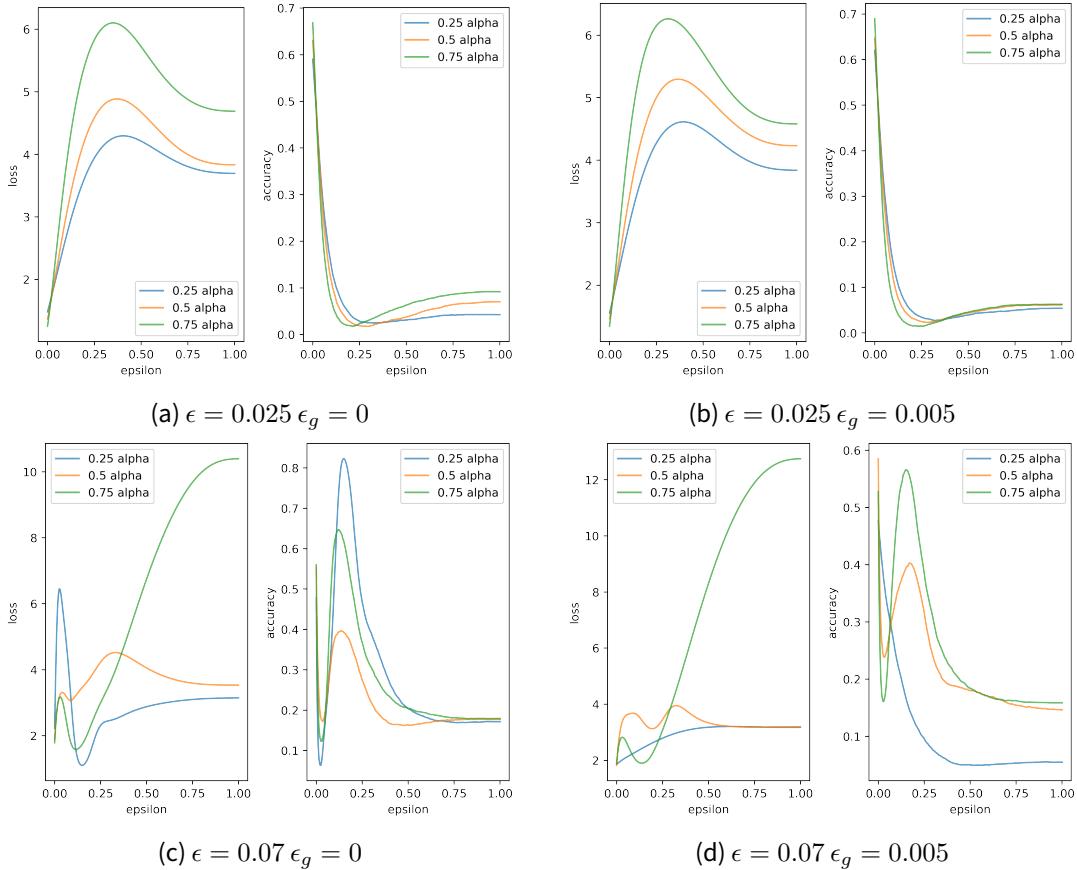
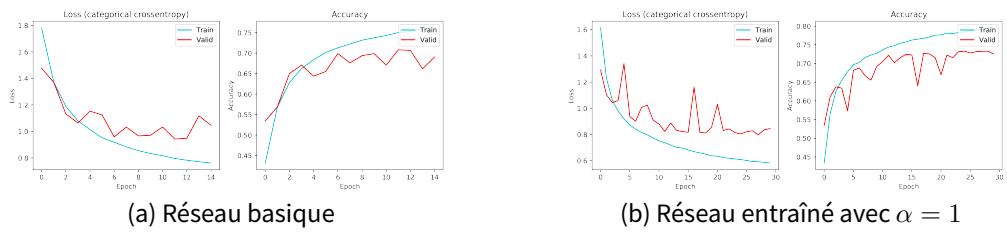


Fig. 8

Fig. 9: Comparaisons de courbes d'entraînement de deux réseaux entraînés respectivement par un code classique et un code de défense FGSM avec  $\alpha$  égal à 1

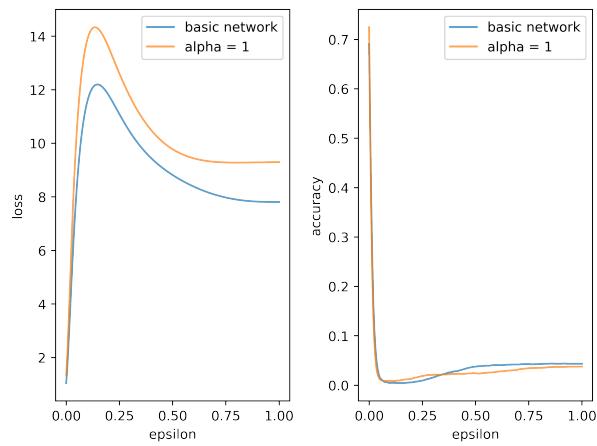


Fig. 10: Graphique présentant l'évolution de la précision et de la loss du réseau présenté en figure 9 en fonction de epsilon sur le jeu de test perturbée par une attaque FGSM