

Adversarial machine learning

Haowei WANG p1911278
Quentin REJAUMONT p1609656
Ahmed Amine BENSLIMANE p2212225

janvier 2023

1 Introduction

1.1 Adversarial machine learning

”Adversarial machine learning” est un sous-domaine de ”machine learning” qui se concentre sur le développement d’algorithmes et de techniques pour se défendre contre les attaques malveillantes sur les modèles d’apprentissage automatique. Il implique la création d’exemples adverses, qui sont des entrées pour un modèle qui ont été spécifiquement conçues pour tromper le modèle en faisant des prédictions incorrectes. Ces exemples peuvent être utilisés pour tester la robustesse d’un modèle et identifier les vulnérabilités qui pourraient être exploitées par un attaquant. L’apprentissage par renforcement inclut également des techniques pour détecter et atténuer de tels attaques, ainsi que des méthodes pour former des modèles à être plus robustes contre eux.




1.2 Les modalités d’attaque

Dans notre projet, nous avons utilisé les algorithmes d’attaque FGSM et PGD. FGSM (Fast Gradient Sign Method) et PGD (Projected Gradient Descent) appartiennent à la méthode d’attaque d’exemples adverses.

1.2.1 FGSM

FGSM est une méthode d’attaque d’exemples adverses basée sur les gradients qui perturbe légèrement les données d’entrée pour faire une classification erronée par le modèle. Plus précisément, FGSM calcule les gradients de la classe cible pour les données d’entrée courante, puis perturbe légèrement les données d’entrée pour faire une classification erronée par le modèle.

Exemple FGSM :

	$+ .007 \times$		$=$	
x		$\text{sign}(\nabla_x J(\theta, x, y))$		$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
“panda”		“nematode”		“gibbon”
57.7% confidence		8.2% confidence		99.3 % confidence

1.2.2 Targeted FGSM

Targeted FGSM et FGSM sont tous deux des variantes de la méthode d'attaque Fast Gradient Sign Method (FGSM), mais ils diffèrent par leurs objectifs et la façon dont ils modifient les données d'entrée.

- FGSM est une méthode d'attaque non ciblée, ce qui signifie qu'elle vise à faire classer une entrée par un modèle dans n'importe quelle classe incorrecte. Il fonctionne en calculant le gradient de la fonction de perte par rapport aux données d'entrée, puis en appliquant une petite perturbation dans la direction du gradient qui maximise la perte, entraînant une mauvaise classification de l'entrée par le modèle.
- Targeted FGSM est une méthode d'attaque ciblée, ce qui signifie qu'elle vise à faire classer une entrée par un modèle dans une classe spécifique ciblée. L'attaquant spécifie la classe ciblée et l'algorithme modifie les entrées de telle sorte que le modèle soit plus susceptible de les classer dans la classe ciblée. Targeted FGSM fonctionne de manière similaire à FGSM, mais au lieu de maximiser la perte, il modifie l'entrée pour minimiser la perte pour la classe ciblée et la maximiser pour les autres classes.

En résumé, Targeted FGSM est une variation de FGSM qui est plus discrète et difficile à détecter, car la sortie du modèle peut ne pas être manifestement incorrecte.

1.2.3 PGD

PGD est une méthode d'attaque d'exemples adverses étendue, similaire à FGSM, mais il perturbe les données d'entrée plusieurs fois jusqu'à ce que le modèle classe incorrectement. PGD utilise l'algorithme de descente de gradient pour mettre à jour les perturbations et limite l'intervalle de perturbations pour éviter des perturbations trop importantes.

2 Notre travail

Notre projet comprend deux parties.

2.1 Partie 1 : MNIST with FGSM

La première partie est l'utilisation de FGSM pour tromper un modèle sur les données MNIST.

2.1.1 Processus

Notre processus pour ce projet est :

1. Définir la valeur de perturbation et utiliser le modèle MNIST pré-entraîné.
2. Construire un modèle attaqué.
3. Définir la fonction d'attaque FGSM.
4. Définir la fonction de test.
5. Exécuter l'attaque.
6. Afficher les résultats.

2.1.2 La fonction d'attaque FGSM

Maintenant, nous pouvons définir la fonction qui crée les exemples adverses en perturbant les entrées originales. La fonction `fgsm_attack` prend trois entrées, l'*image* est l'image originale propre(x), *epsilon* est la quantité de perturbation par pixel (ϵ), et *data_grad* est le gradient de la perte par rapport à l'image d'entrée ($\nabla_x J(\theta, \mathbf{x}, y)$).

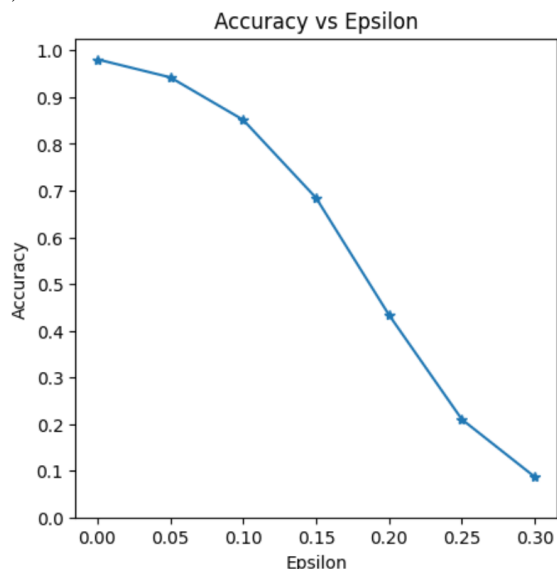
La fonction crée alors une image perturbée sous la forme

$$perturbed_image = image + epsilon * sign(data_grad) = x + \epsilon * sign(\nabla_x J(\theta, \mathbf{x}, y))$$

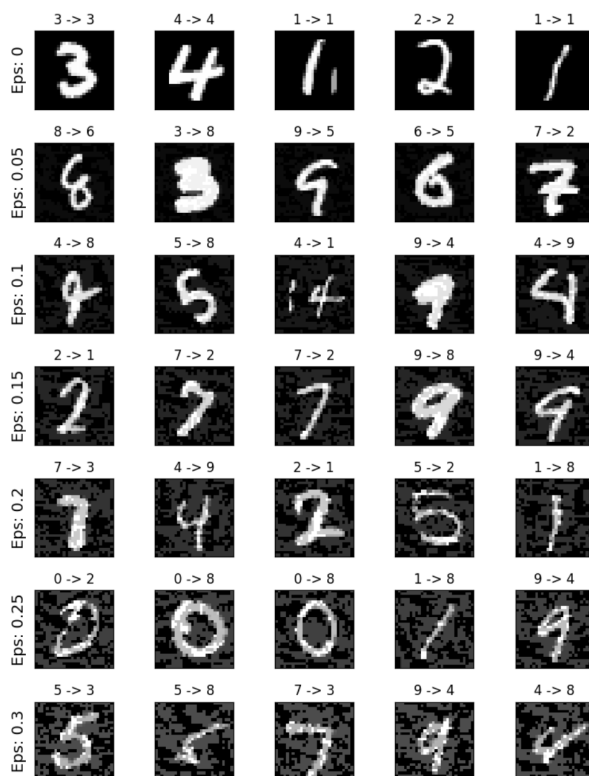
Enfin, afin de conserver la plage originale des données, l'image perturbée est échantillonnée à la plage $[0, 1]$.

2.1.3 Accuracy vs Epsilon

Le premier résultat est le graphique de la précision en fonction d'épsilon. Comme nous l'avons vu précédemment, lorsque epsilon augmente, la précision du test devrait diminuer. Cela s'explique par le fait que plus l'épsilon est grand, plus nous faisons un pas dans la direction qui maximise la perte. Remarquez que la tendance de la courbe n'est pas linéaire, même si les valeurs d'épsilon sont linéairement espacées. Par exemple, la précision à $(\epsilon)=0,05$ n'est inférieure que d'environ 4% à celle de $(\epsilon)=0$, mais la précision à $(\epsilon)=0,2$ est inférieure de 25% à celle de $(\epsilon)=0,15$. Aussi, remarquez que la précision du modèle frappe la précision aléatoire pour un classificateur à 10 classes entre $(\epsilon)=0,25$ et $(\epsilon)=0,3$.



2.1.4 Final result



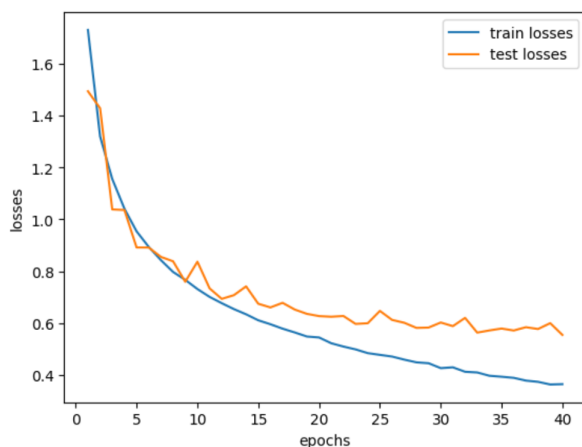
2.2 Partie 2 : Adversarial Training on CIFAR-10 with FGSM and PGD attacks

Dans ce notebook, nous lançons des attaques ciblées et non ciblées à l'aide des méthodes FGSM et PGD sur le modèle Resnet18, qui a été entraîné sur le jeu de données CIFAR-10. Nous mettrons également en place des mécanismes d'entraînement adversaires pour se défendre contre ces attaques et construire des modèles plus robustes contre elles.

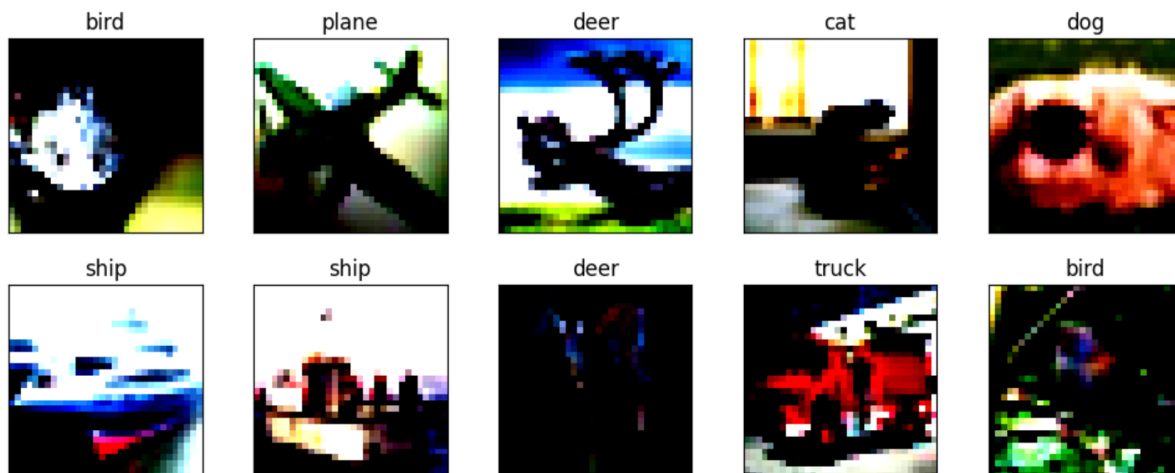
2.2.1 Processus

- **Preparing train and test data and building Resnet model**
 - Training function
 - Test function
 - Trainig and test loss of Resnet18 model on Cifar-10
- **FGSM**
 - Visualizing 10 selected samples from the dataset
 - FGSM attack function
 - Creating adversarial examples from samples with the FGSM attack and $\text{eps} = 1/255$
 - Adversarial Training with FGSM
 - Comparing naturally-trained and adversarially-trained models
 - Evaluating the adversarially-trained model with FGSM against FGSM attack on test data
 - Targeted FGSM
- **PGD**
 - PGD attack function
 - Adversarial Training with PGD
 - Evaluating the adversarially-trained model with PGD against PGD attack on test data

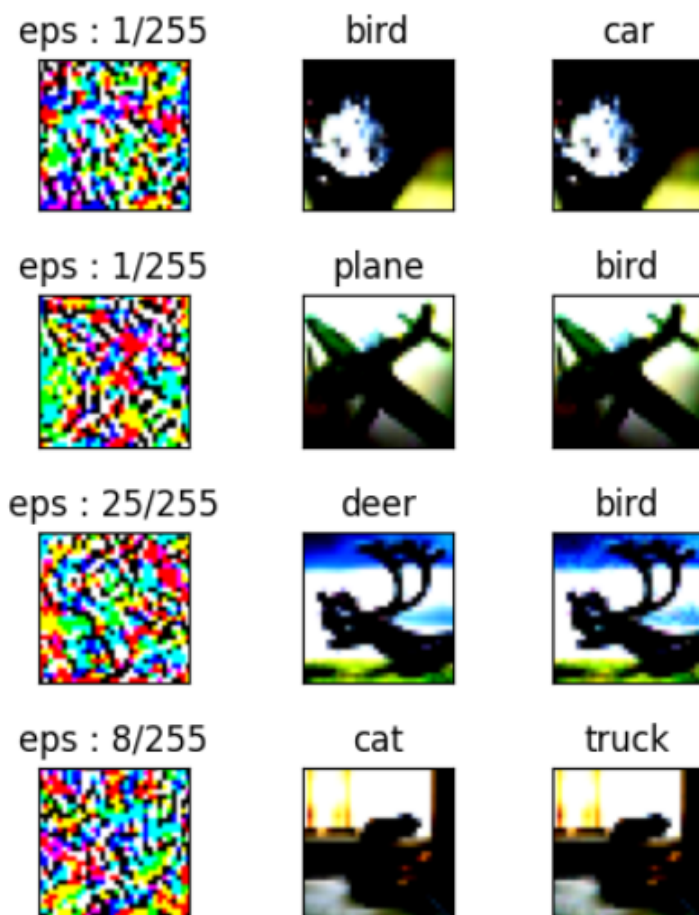
2.2.2 Training and test loss of Resnet18 model on Cifar-10



2.2.3 Visualizing 10 selected samples from the dataset

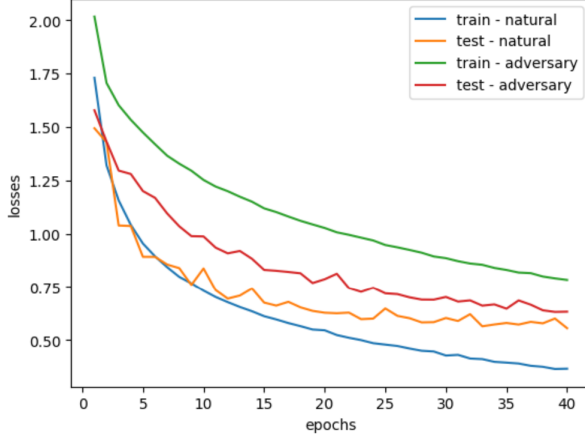


2.2.4 Visualizing selected samples with FGSM



Pour l'œil humain, les exemples original et adversarial sont extrêmement similaires.

2.2.5 Comparing naturally-trained and adversarially-trained models



Les pertes de formation du modèle formé de manière contradictoire sont plus élevées que les pertes de formation du modèle formé naturellement, ce qui est intuitif puisque le modèle formé de manière contradictoire est formé contre des exemples contradictoires, ce qui rend plus difficile pour le modèle d'étiqueter correctement ces entrées perturbées et entraîne des erreurs plus élevées.

La perte du modèle formé naturellement sur les données de test est plus élevée que la perte de formation, puisque les données de test ne sont pas vues par le modèle, ce qui entraîne une erreur de classification plus élevée.

Cependant, la perte du modèle entraîné de façon adversariale sur les données de test est inférieure à la perte d'entraînement correspondante. Cela est probablement dû au fait que les instances de test ne sont pas contradictoires (contrairement aux données de formation) et que le modèle a appris à extraire des caractéristiques importantes et utiles, ce qui lui permet d'obtenir de meilleures performances sur les images de test.

2.2.6 Evaluating the adversarially-trained model with FGSM against FGSM attack on test data

epsilon : 0.011267605633802818 accuracy : 77.64

epsilon : 0.03137254901960784 accuracy : 76.36

epsilon : 0.047058823529411764 accuracy : 75.11

Nous constatons qu'à mesure que epsilon augmente, la précision diminue, car pour des epsilon plus grands, des perturbations plus importantes sont autorisées, et il devient donc plus difficile pour le modèle d'étiqueter correctement ces exemples perturbés.

2.2.7 Targeted FGSM

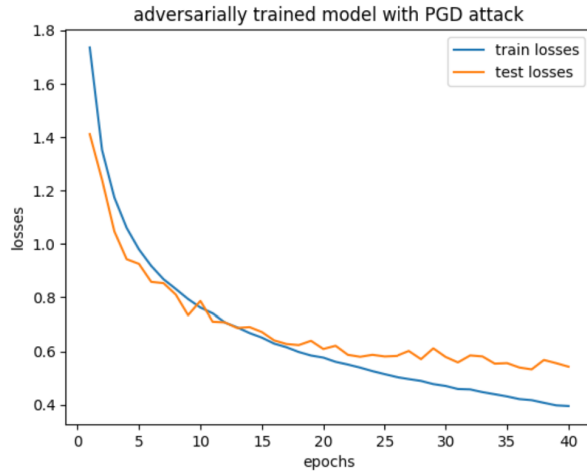
Dans les attaques ciblées, nous voulons que le modèle classe mal son entrée dans la classe cible donnée. Par conséquent, au lieu de simplement maximiser la perte de la véritable étiquette, nous maximisons la perte de la véritable étiquette et minimisons la perte de l'étiquette alternative.

Nous constatons que, dans les deux cas, la précision du modèle formé par l'adversaire est beaucoup plus élevée que celle du modèle formé naturellement, car le modèle formé par l'adversaire est devenu plus robuste et a donc une meilleure précision face aux exemples adverses.

2.2.8 PGD attack function

we repeat $\delta := \mathcal{P}(\delta + \alpha \nabla_{\delta} l(\theta, x, y))$ for t iterations

2.2.9 adversarially trained model with PGD attack



2.2.10 adversarially trained model with PGD attack

Train - natural : training loss of the naturally-trained model

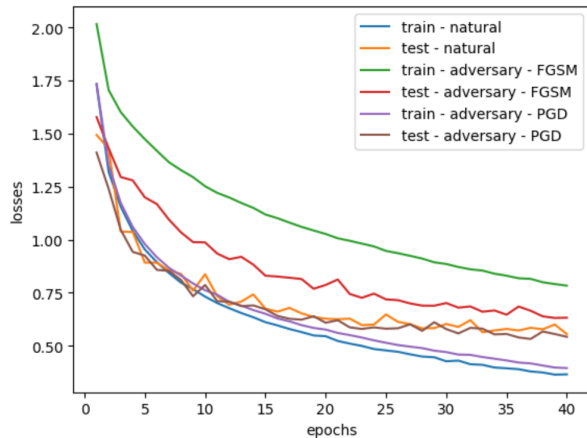
Train - adversary - FGSM : training loss of the adversarially-trained model (on examples generated with FGSM)

Train - adversary - PGD : training loss of the adversarially-trained model (on examples generated with PGD)

Test - natural : loss of the naturally-trained model on original (unperturbed) test images

Test - adversary - FGSM : loss of the adversarially-trained model (on examples generated with FGSM) on original (unperturbed) test images

Test - adversary - PGD : loss of the adversarially-trained model (on examples generated with PGD) on original (unperturbed) test images



2.2.11 Evaluating the adversarially-trained model with PGD against PGD attack on test data

accuracy of net_pgd against PGD attack with iters= 3 : 82.33

accuracy of net_pgd against PGD attack with iters= 7 : 81.88

accuracy of net_pgd against PGD attack with iters= 12 : 81.28

Nous constatons qu'à mesure que les itérations augmentent, la précision diminue. La raison en est qu'à mesure que le nombre d'itérations augmente, la fonction PGD produit des attaques plus fortes qui rendent plus difficile pour le modèle de les étiqueter correctement. Par conséquent, la précision du modèle diminue.

De plus, comme net_pgd a été entraîné sur des exemples d'adversaires produits à l'aide de l'attaque PGD avec 3 itérations, il a une bonne performance (82,33%) sur des exemples similaires, c'est-à-dire

pour $\text{iter}=3$. Cependant, sa précision diminue contre les attaques PGD avec des itérations plus élevées (7 et 12). Cela est probablement dû au fait que `net_pgd` a été entraîné avec PGD-3 ($\text{iter}=3$) et qu'il lui est plus difficile de se défendre contre des PGD avec 2 ou 4 fois plus d'itérations.

Références

- [1] [ADVERSARIAL EXAMPLE GENERATION.](#)
- [2] [Adversarial Training on CIFAR-10 with FGSM and PGD attacks](#)