Présentation finale
Robust neural networks

Vincent Gouteux
Tiphaine Le Clercq
Maxence Philbert

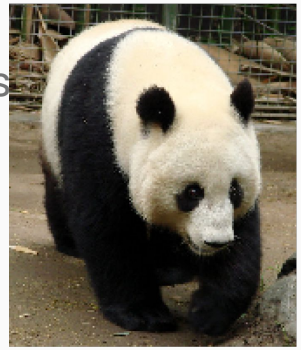
17/12/2019

1) Présentation du problème

- On s'intéresse à la robustesse des réseaux de neurones
- But : implémenter des attaques (**adversarielles puis Black-box)** pour tromper un réseau de neurones de classification d'images puis implémenter un moyen de défense contre celles-ci
- Données utilisées : CIFAR10 (60 000 images de taille 32*32 réparties en 10 classes : avion, voiture, oiseau, chat …)

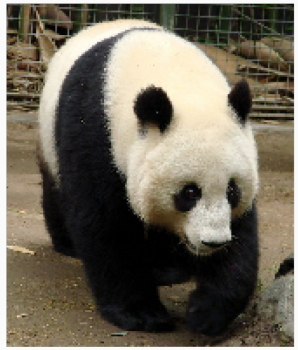
2) Attaques adversarielles
Principe

- But : générer des exemples afin de tromper un réseau de neurones
- En classification d'images : générer des images perturbées de telle sorte que le réseau prédise mal leur classe
- Perturbation faible (non détectable à l'oeil humain) mais assez forte pour que le réseau prédise la mauvaise classe



original image
prediction: giant_panda

the perturbation,
enhanced 127 times



perturbed image
prediction: bucket

2) Attaques adversarielles
FGSM et FGM : Formules

- But : générer une perturbation δ telle que $f(x) = y$ mais $f(x+\delta) \neq y$ sous la contrainte $||\delta|| \leq \epsilon$

$$\delta = \underset{||\delta|| \leq \epsilon}{argmax} \mathcal{L}(f, x+\delta, y)$$

- Attaque FGSM : avec norme L^∞

$$\delta^* = \epsilon sign(\nabla_x \mathcal{L}(f, x, y))$$

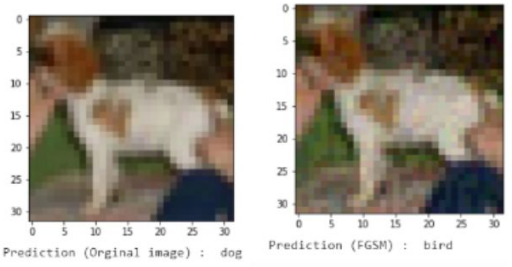
- Attaque FGM : avec norme L^2

$$\delta^* = \epsilon \frac{\nabla_x \mathcal{L}(f, x, y)}{||\nabla_x \mathcal{L}(f, x, y)||_2}$$

2) Attaques adversarielles
FGSM et FGM : Résultats

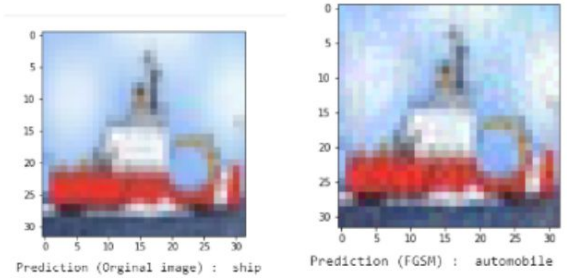
	Images originales	Attaque FGSM $\epsilon = 0.031$	Attaque FGM $\epsilon = 0.4$
Training Accuracy	0.812	0.0988	A 0.0982
Testing Accuracy	0.684	0.0014	0.1303

Table 1: Précisions d'entraînement et de validations obtenues



(a) Chien (b) Oiseau

Figure 5: Mauvaise classification d'une image de chien



(a) Bateau (b) Voiture

Figure 6: Mauvaise classification d'une image de bateau

2) Attaques adversarielles
PGD : Formules

- Version itérative des attaques FG(S)M vues précédemment
- On génère itérativement des perturbations de la manière suivante :

$$\begin{cases} x_0 &= x \\ x_{t+1} &= \pi_{B(0,\epsilon)}(x_t + \underset{||\delta|| \leq \eta}{argmax} \mathcal{L}(f, x_t + \delta, y)) \end{cases}$$

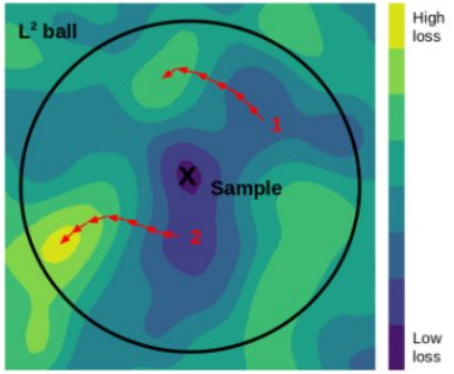


Figure 9: Attaque PGD

- Résultats :

	Images originale	PGD L^∞ ($\eta = 0.031$ et $\epsilon = 0.05$)	PGD L^2 ($\eta = 0.4$ et $\epsilon = 0.05$)
Training Accuracy	0.812	0.0052	0.0044
Testing Accuracy	0.684	0.1465	0.0906

Table 2: Précision d'entraînement et de validations obtenues

3) Attaques Black-box

- On suppose que l'on ne sait rien sur le réseau à part sa prédiction pour une entrée donnée (ici un vecteur de probabilités de taille 10)
- Pas d'accès au gradient de la fonction de perte donc impossible d'utiliser les méthodes vues précédemment.
- Comment trouver la perturbation optimale ?

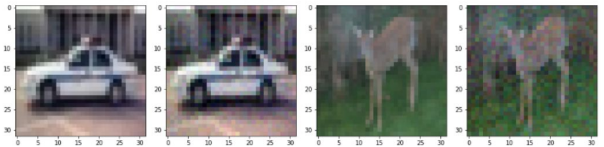
3) Attaques Black-box
Générateur de perturbations aléatoires

Principe :

- On génère une perturbation **aléatoire** très faible
- On regarde ce que prédit le réseau
- Si la proba d'appartenir à la bonne classe diminue alors on conserve la perturbation
- Sinon, on retourne à l'image à l'état précédent
- Conditions d'arrêt : le réseau se trompe + borne sur la norme de la perturbation

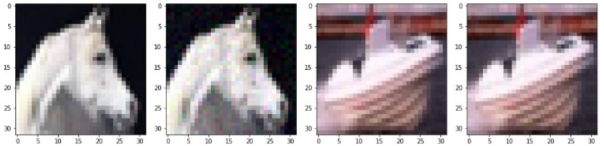
	Images originales	Attaque Random
Testing Accuracy	0.812	0.125

Table 4: Précision du modèle sur les images avec perturbations *Aléatoires*



(a) Voiture (b) Camion (c) Cerf (d) Cheval

Figure 13: Images perturbées *Aléatoirement* et leur prédictions

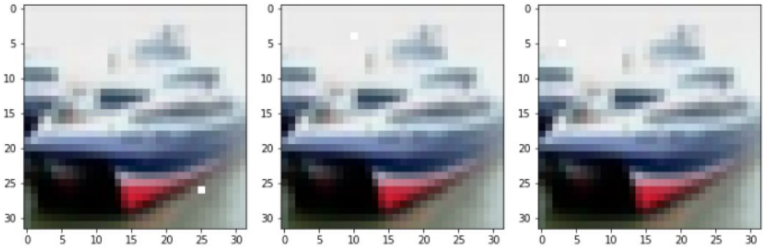


(a) Cheval (b) Oiseau (c) Bateau (d) Chat

Figure 14: Images perturbées *Aléatoirement* et leur prédictions

3) Attaques Black-box
Modification d'un pixel

- Il est possible de tromper le réseau en modifiant un seul pixel d'une image
- Test effectué sur une image :
  - on modifie un par un ses pixels (on met le pixel à 1)
  - on demande au réseau de prédire la classe à chaque fois
  - le réseau se trompe 298 fois sur 1024 (soit 30% de mauvaise classification)

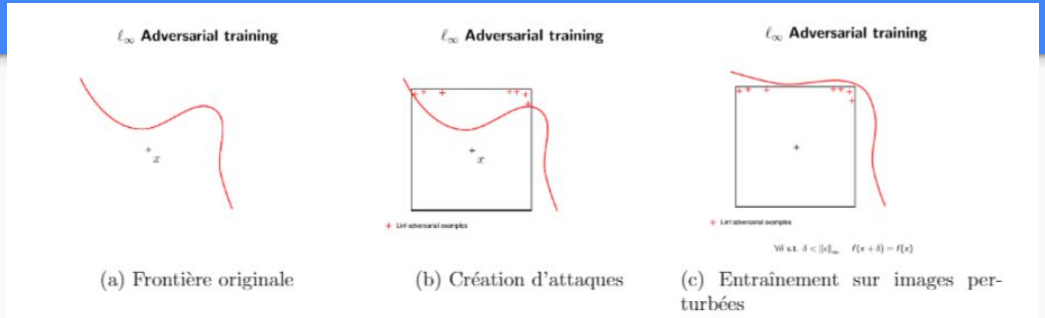


(a) Cheval (b) Oiseau (c) Bateau

Figure 16: Exemples de pixels qui provoquent une mauvaise classification

4) Défense : Adversarial training
Principe

- Principe : On entraîne le réseau avec des images perturbées
- Résultats obtenus :



(a) Frontière originale (b) Création d'attaques (c) Entraînement sur images perturbées

Figure 10: Étapes de *l'Adversarial Training*

		Image originale	FGSM	PGD
Réseau Original	*Training Accuracy*	0.812	0.0988	0.0982
	Testing Accuracy	0.684	0.0014	0.1303
Réseau FGSM	*Training Accuracy*	-	-	-
	Testing Accuracy	0.4321	0.3618	0.3683
Réseau PGD	*Training Accuracy*	-	-	-
	Testing Accuracy	0.5863	0.3823	0.4048

Table 3: Précisions obtenues pour les 3 modèles