Projet sur les exemples adversaires

ENSTA 2022-2023
Pol Labarbarie, Adrien Chan-Hon-Tong

1 Introduction

1.1 Définition

Les exemples adversaires ont été brièvement évoqués en cours mais un rappel s'impose.

On considère un réseau de neurones profond (paramétré par des poids w) f_w qui prend en entrée des images RGB (qu'on va supposer de taille fixe de 224×224 pixels) et qui produit pour chaque image un score d'appartenance à chacune des 1000 classes d'Imagenet (donc f_w prend en entrée $x \in \mathbf{R}^{3\times224\times224}$ et $f_w(x) \in \mathbf{R}^{1000}$).

Considérons maintenant une image χ donc la classe $y(\chi) \in \{0, ..., 999\}$ est connue et qui est correctement traitée par le réseau f_w (c'est à dire que $\underset{c\in\{0,...,999\}}{\arg\max}\ f_w(\chi)_c = y(\chi)$).

On dit alors que f_w admet un exemple adversaire $\delta \in \mathbf{R}^{3\times224\times224}$ de norme ε en χ si $\underset{c\in\{0,...,999\}}{\arg\max}\ f_w(\chi+\delta)_c \neq y(\chi)$ et $||\delta||_\infty \leq \varepsilon$.

Autrement dit, on a un exemple adversaire en x, si en bougeant un tout petit peu la valeur de chaque pixel de l'image x, on est capable de changer la valeur pourtant correcte de $f_w(x)$. En quelque sorte, il s'agit de point de pseudo discontinuité de f_w.

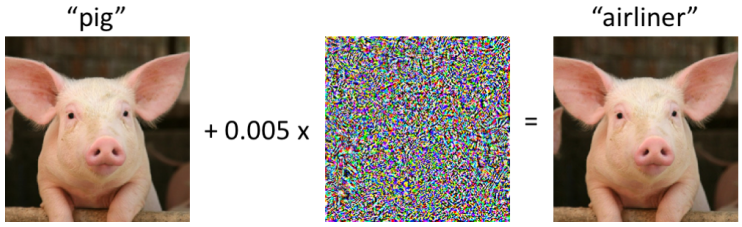

FIGURE 1 – Illustration de la perturbation d'une image par un exemple adverse. Sur l'image de gauche, le réseau classifie correctement l'image avec un haut niveau de confiance. Sur l'image de droite, après perturbation adverse, le réseau classe mal l'image avec un très haut niveau de confiance. La perturbation causée par la perturbation adverse est imperceptible pour l'oeil humain.

1.2 Calcul pratique

En pratique, les poids du réseau w ont été optimisé par descente de gradient stochastique pour minimiser une fonction de perte $L(f_w(x), y(x))$ (généralement

L est la cross entropy). Cela suppose qu'on sache calculer le gradient de la perte par rapport à w qu'on peut noter $\frac{\partial L(f_w(x),y(x))}{\partial w}$ ou $\nabla_w L(f_w(x), y(x))$ (c'est pytorch qui le fait).

Ainsi, en modifiant très légèrement le code, on peut non plus optimiser w mais δ de sorte à maximiser $L(f_w(x+\delta), y(x))$ (via le calcul de $\frac{\partial L(f_w(x+\delta),y(x))}{\partial \delta}$ ou $\nabla_\delta L(f_w(x+\delta), y(x))$).

Attention toutefois à ne pas oublier qu'on impose que $||\delta||_\infty \leq \varepsilon$ (il conviendra par exemple de remettre les valeur de δ qui seraient trop grandes dans l'intervalle $[-\varepsilon, \varepsilon]$).

2 Format et objectif du projet

Dans ce projet, 50 images Imagenet seront fournies ici attention c'est un lien temporaire. Ces images auront été choisies de sorte qu'elles soient correctement traitées par la plupart des réseaux préexistant sur torchvision notamment elles sont bien classées par Resnet50 et Resnet18 de torchvision.

Le projet devra commencer par une partie imposée qui guidera vers la construction d'exemples adversaires simples (avec la méthode FGSM). Puis, le projet devra se poursuivre via l'exploration autonome d'une (ou plusieurs) question parmi 3 proposées :

- Comment construire des attaques plus puissantes et moins détectables (méthode PGD et contrainte d'attaque entière etc) ?
- Que ce passe-t-il lors de l'attaque ? Cette question invitera à regarder l'impact de l'attaque sur les distributions de score, à regarder si les erreurs sont minimes (un chien pris pour un loup) ou étonnantes (un chien pris pour un tracteur) et à regarder si l'attaque a une trace dans les neurones internes du réseau.
- Comment réaliser une telle attaque si on ne dispose pas réellement de la fonction f_w (par exemple, on optimisera δ pour qu'il marche sur un autre réseau g_ω et on regardera si δ produit un effet sur f_w, notamment avec des attaques au niveau des couches internes).

3 Partie imposée

Dans cette première partie, vous allez implémenter une attaque très simple dites FGSM pour *Fast Gradient Sign Method* (vous pourrez trouver de nombreux blogs sur cette approche sur internet).

Cette méthode consiste simplement en $\delta \leftarrow \varepsilon \times \text{sign}\left(\nabla_\delta L(f_w(x+\delta), y(x))\right)$.

Dit autrement, on calcule pour chaque pixel i, j, la dérivée de $L(f_w(x+\delta), y(x))$ par rapport à ce pixel i, j. Si cette dérivée est positive, alors $\delta_{i,j} = \varepsilon$, sinon, $\delta_{i,j} = -\varepsilon$. L'idée sous-jacente c'est que par construction on aura bien $||\delta||_\infty \leq \varepsilon$.

Vérifier que les images fournies sont bien classées par Resnet50 et Resnet18 de torchvision. Implémenter cette attaque sur ces images

contre ces deux modèles et reporter les performances avec et sans attaques pour ces deux modèles.

On prendra $\varepsilon = 18$ (si on considère la valeur des pixels entre 0 et 255) ou $\varepsilon = 0.07$ si on considère leur valeur entre 0 et 1.

N'hésitez pas à poser des questions en cas de doute. Cette partie est sensée être guidée !

4 Parties en autonomie

4.1 De FGSM à PGD

Considérer $\delta \leftarrow \varepsilon \times \text{sign}\left(\nabla_\delta L(f_w(x+\delta), y(x))\right)$ est évidemment améliorable car on n'optimise pas vraiment δ : celui-ci est défini en un seul pas de gradient.

Cependant, une approche bien plus performante a été développé après : la méthode PGD dont vous trouverez une description (en bas de la page 4) de cet article (Towards Deep Learning Models Resistant to Adversarial Attacks).

Globalement, il s'agit juste d'optimiser δ via une multitude d'itération (plutôt que 1) tout en maintenant les contraintes de norme.

Implémenter cette attaque PGD sur les images fournies avec un Resnet50 et un Resnet18 et reporter les performances. Vous devriez atteindre une performance proche de 0 ! Diminuer la norme pour observer à partir de quelle norme vous arrivez à maintenir une performance proche de 0. Dans le même temps, on rajoute la contrainte que les valeurs de l'image modifiée doivent rester entre 0 et 255 et être des entiers (ce qui est le cas normalement). Reportez les performances avec ces contraintes et différente amplitudes (norme) d'attaque.

4.2 Le mécanisme de l'attaque

L'idée de cette question est d'analyser plus précisément l'effet de l'attaque basique.

Pour commencer, il faudra aller chercher les classes d'Imagenet (que veulent dire la classe 10 et la classe 42 ?). Puis, vous analyserez précisément les exemples adversaires produits : quelle classe au départ et quelle classe à l'arrivée ? Vous analyserez aussi si la confusion est faible ou forte (est ce que le score de confiance envers la nouvelle classe est fort ou pas) ? Utilisez les scores avant attaque pour voir quelle devrait être la dynamique attendue. Enfin essayez de voir si certains neurones du réseau donnent des valeurs très différentes pour les images modifiées que pour les images originales.

4.3 Attaque universelle

L'objectif est ici d'être capable d'attaquer le resnet18 avec un bruit optimisé sur le resnet50 ! Il faut donc que le bruit cherche à capturer quelque chose qui devrait tromper pas seulement le réseau observé mais également d'autres.

Un papier a apporté une rupture sur ce point : (Feature space perturbations yield more transferable adversarial examples).

Implémentez cette attaque.

5 Annexe : Utilisation des réseaux de torchvision

Pour utiliser correctement un réseau torchvision, il faut le faire comme ça :

```
import torch
import torchvision

net = torchvision.models.resnet18(weights="IMAGENET1K_V1")
moyenne = torch.Tensor([0.485, 0.456, 0.406])
variance = torch.Tensor([0.229, 0.224, 0.225])
normalisation = torchvision.transforms.Normalize(moyenne, variance)

x = torchvision.io.read_image("projet/img0.png")
x = x/255
x = x.resize(1,3,224,224)
x = normalisation(x)
score = net(x)
```

Par ailleurs vous pouvez utiliser wget dans google colab pour rappatrier des données exemple

```
!wget lien_vers_archive/Imagenet_images.tar.xz

!tar -xf Imagenet_images.tar.xz

!ls
```

devrait télécharger l'archive, la dézipper et afficher les dossiers locaux en l'occurrence un dossier "projet" avec des images dedans (il faut évidemment remplacer lien_vers_archive par le lien vers l'archive...).