POM - Opening the Black Box Simplification et interprétabilité des réseaux de neurones profonds : Partie Image

GUADEBOIS Victor
JAAFAR Joachim
Encadrant : PLANTEVIT Marc

Table des matières

Chapitre 1

Introduction

Le *machine learning* ("apprentissage automatique") est une approche de l'intelligence artificielle qui se base sur l'étude de modèles qui s'améliorent automatiquement pendant un processus d'apprentissage. Il se fonde sur des études statistiques sur des données collectées au préalable pour en extraire des connaissances, afin d'améliorer les performances à résoudre des tâches automatiquement. Le *deep learning* ("apprentissage profond") correspond à un type d'apprentissage automatique se basant sur une structure appelée réseau de neurones.
En traitement d'images, le *deep learning* est régulièrement utilisé dans des domaines où la prédiction porte de lourdes conséquences, tels que le traitement d'images médicales (détecter la présence -ou non- d'un cancer par exemple), ou la reconnaissance de formes (faire en sorte qu'une voiture autonome soit capable de s'arrêter face à un piéton) : une prédiction incorrecte peut être une question de vie ou de mort. On peut aussi citer la recommandation des contenus dans les médias sociaux, qui doivent faire preuve de transparence. Il est donc primordial d'être capable d'expliquer comment et pourquoi une prédiction a été choisie par un modèle.
De nos jours, **les algorithmes de deep learning sont peu *interprétables*** (il est difficile d'identifier et de quantifier l'importance des caractéristiques ou variables qui participent le plus à la décision). Le modèle créée après l'apprentissage n'est pas explicite : c'est ce qu'on appelle une boîte noire.
C'est là qu'entrent en jeu les *explainers*. Un explainer est une technique d'explication qui nous permet de découvrir quels sont les éléments d'une donnée en entrée qui ont été les plus importants pour sa classification, ce qui nous permet donc de connaître l'origine d'une prédiction. Dans le contexte de l'imagerie, un explainer va récupérer l'image, et va mettre en importance les parties de celle-ci qui ont permis de la classifier. Un explainer pionnier dans le domaine est LIME [1], que nous allons d'abord traiter dans ce rapport. Cet explainer est performant, mais très lent dans son exécution. Afin d'expliciter cela, nous allons y apporter une critique qualitative afin de trouver quels sont les défauts à compenser, grâce à deux modèles d'apprentissage automatique, *AlexNet* [2] et *GoogleNet* [3]. Pour pallier à cela, nous allons ensuite parler de comment nous avons implémenté

1. `https://github.com/marcotcr/lime`
2. `https://cs.nju.edu.cn/zhangl/alexnet.pdf`
3. `https://static.googleusercontent.com/media/research.google.com/fr//pubs/archive/43022.pdf`

nos propres explainers et en faire la comparaison avec *LIME*.

Chapitre 2

LIME

LIME ("Local Interpretable Model-agnostic Explanation") est un explainer que nous avons utilisé pendant nos travaux. Il est intéressant car, comme son acronyme l'indique, il est :

— localement fidèle : l'explication d'un modèle ne peut être fidèle que s'il décrit complètement ce modèle. LIME est un explainer qui réplique le comportement du modèle sur l'instance sur laquelle la prédiction est faite.

— modèle-agnostique : un explainer devrait être capable d'expliquer plusieurs, voire n'importe quel modèle. LIME ne fait aucune supposition sur le modèle qu'il doit expliquer.

— interprétable : n'importe qui devrait être capable de comprendre les explications d'un explainer, c'est-à-dire de distinguer les variables importantes du reste. LIME produit un résultat qualitatif et visuellement compréhensible :

(a) Image originale

(b) Explication via GoogleNet

(c) Explication via AlexNet

L'image (b) correspond à l'utilisation de LIME avec *GoogleNet* comme classifieur, et l'image 3 avec *AlexNet*. La partie colorée qui ressort est la zone de l'image qui a permis au classifieur de faire sa prédiction.

Tout au long de ce rapport nous utiliserons la même image pour pouvoir observer les différences entre plusieurs explications.

Comme beaucoup de techniques utilisant des modèles de machine learning ou deep learning, *LIME* a pris environ une minute pour créer ce résultat. Cela peut être dû à son algorithme qui se

base sur une analyse complexe du voisinage de chaque zone de l'image. Un tel temps d'exécution pour une seule image est problématique et nous montre les limites des approches d'interprétabilité existantes. Une partie importante de nos travaux a été de développer de nouveaux explainers.

Chapitre 3

Travail effectué

3.1 Apprentissage et recherches

Durant le premier mois, notre objectif était d'être à niveau pour entamer le projet de façon optimale et de faire un maximum de recherches sur le sujet. Pour cela nous avons décidé de faire une montée en compétence sur le deep learning grâce à des cours en ligne 6.1. Nous avons essayé de faire le plus de semaines de cours possible pour à la fois avoir les compétences pour réaliser ce projet, mais aussi pour avoir une meilleure vision du sujet. Et pour les recherches nous avons lu un grand nombre d'articles conseillés par notre encadrant de POM et d'autres articles trouvés en ligne. 6.2

3.2 Les classifieurs et le jeu de données

Avant d'utiliser LIME, il nous fallait plusieurs modèles de classifieurs à mettre en entrée. Nous avons choisi AlexNet et GoogleNet, deux classifieurs d'images qui ont respectivement gagné le ImageNet Large Scale Visual Recognition Challenge (une compétition entre classifieurs d'images) en 2012 et 2014. Ce sont tous les deux des réseaux de neurones convolutifs.

Après plusieurs tests, nous avons remarqué que ces deux modèles étaient globalement aussi performants l'un que l'autre : sur certaines images, AlexNet prédit mieux que GoogleNet, mais l'inverse est aussi vrai sur d'autres images. Cependant, nous avons soulevé leur différence concernant leur vitesse d'exécution : GoogleNet est globalement plus rapide qu'AlexNet, c'est donc celui-ci que nous avons gardé pour évaluer LIME et nos propres explainers. De plus, GoogleNet a l'air généralement plus précis qu'AlexNet. Cela peut s'expliquer par le fait qu'il soit plus récent, et donc plus optimisé grâce aux avancées exponentielles concernant l'hardware.

Le jeu de données que nous avons utilisé est "Caltech-256 Object Category Dataset" [1], conte-

1. `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`

nant plusieurs milliers d'images de 256 catégories différentes, afin d'obtenir une grande variété de résultats. Nous avons sélectionné aléatoirement une image de chacune de ces catégories.

3.3 Implémentation

Concernant l'implémentation de LIME et de nos explainers, nous nous sommes inspirés d'un exemple de l'auteur de LIME [2]. Les modèles pré-entraînés sont importés grâce à la librairie Python Torchvision.
Avec un de ces modèles, nous récupérons le label prédit avec les plus haut pourcentage de confiance sur l'image grâce à une liste de labels utilisée par ImageNet, une base de données d'images. Enfin, l'explainer va découper l'image de manière à mettre en valeur les pixels de l'image qui ont été importants pour cette prédiction. Le résultat est généré grâce à la librairie de traçage et de visualisation Matplotlib.

3.4 Métriques de comparaison

Pour comparer les résultats obtenus avec GoogleNet et AlexNet appliqués à LIME, nous avons pris plusieurs approches différentes :

— la superposition de couleurs : l'explainer colorie en noir les parties de l'image qu'il considère comme peu utiles à la prédiction. Nous avons donc superposé les résultats obtenus et colorié les parties correspondantes à AlexNet en rouge, GoogleNet en vert, et en jaune si la partie a été expliquée avec les deux modèles.

(a) Superposition de couleurs sur l'image du hibou (GoogleNet/AlexNet)

— la distance de Jaccard, qui est une métrique qui permet d'obtenir le taux de similarité entre plusieurs échantillons, dont la formule est, pour deux ensembles A et B :

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Voici les 5 distances de Jaccard les plus basses, les 5 plus hautes et la moyenne :

2. `https://github.com/marcotcr/lime/blob/master/doc/notebooks/Tutorial%20-%20images%20-%20Pytorch.ipynb`

```
FLOP 5 :
053_0017.jpg 0.2680296296296296
223_0057.jpg 0.3310867293625914
235_0047.jpg 0.3597288017732192
051_0036.jpg 0.3809700176366843
188_0025.jpg 0.38472362231182794


TOP 5 :
014_0065.jpg 0.894495137566288
151_0072.jpg 0.8982970108372038
084_0061.jpg 0.8997735257342764
173_0028.jpg 0.9112738715277777
041_0024.jpg 1.0


MOYENNE : 0.6543285598192666
```

(a) Flop, top et moyenne distances de Jaccard (GoogleNet/AlexNet)

En moyenne, les résultats d'explications des modèles appliqués sur LIME ont donc un taux de similarité de 65%.

En observant les images du flop 5.2 on se rend compte que les raisons de ces grandes différences de prédictions sont liées à plusieurs paramètres.
Ce sont des images qui peuvent être interprétées de plusieurs manières : par exemple, pour une image représentant un mannequin avec un chapeau, LIME couplé à GoogleNet extrait le visage du mannequin, alors qu'avec AlexNet, il extrait le chapeau. Il peut aussi y avoir des images où chaque partie représente la même chose, ou avec des symétries. Dans le cas d'une photo du ciel, chaque classifieur peut détecter le ciel grâce à n'importe quelle partie de celui-ci. Sur une image d'un globe terrestre, il est possible que la partie haute ou la partie basse du globe soit extraite. Enfin, il y a parfois des images où un des explainers fait juste une mauvaise prédiction, et donc le résultat, une fois interprété, peut être totalement différent, ce sont souvent des images difficiles à interpréter.

Pour ce qui est du top, on observe souvent le même type d'image 5.2. On peut y observer quelque chose de très reconnaissable où il y a presque aucune chance que l'un des deux explainers se trompent. Et la plupart du temps, l'objet de l'image est reconnaissable grâce à des caractéristiques uniques. Par exemple, un fusil grâce à sa gâchette, son canon ou sa crosse. On peut aussi prendre l'exemple d'une autruche, puisque ses pattes sont facilement reconnaissables.

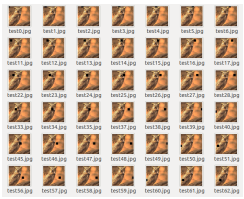
Il ne faut pas oublier que LIME est non-déterministe, ce flop et ce top pourraient donc être totalement différents si on relance LIME sur chaque image, mais c'est globalement le même type d'images qui ressortirait.

3.5 Nos explainers

3.5.1 L'explainer arbitraire

Tout d'abord, l'explainer prend en entrée l'image que l'on veut utiliser. Cette image passe dans un classifieur de notre choix, comme par exemple GoogleNet ou AlexNet. On enregistre la première prédiction du classifieur (c'est-à-dire la prédiction dont le modèle est le plus sûr). L'image est ensuite divisée en plusieurs parties, que nous appellerons "superpixels". Pour l'explainer arbitraire, nous

avons choisi de la découper en superpixels carrés de la taille du plus grand côté de l'image divisée par 10. Après cela nous générons une nouvelle image pour chaque superpixel avec celui-ci colorié en noir.



(a) Images générées pour le hibou

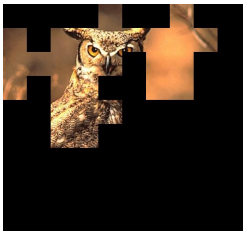
(b) Exemple d'image générée

Nous voyons, sur l'image (a), les images générées pour l'image du hibou avec les superpixels coloriés en noir. Sur l'image (b), on affiche l'une de ces images, pour bien comprendre ce qui est généré.

L'étape d'après consiste à faire passer les images générées dans le classifieur du début et d'enregistrer la prédiction et le pourcentage de confiance attribués à chacune des images. Grâce à cela on peut ordonner les images dans l'ordre de celle qui a la meilleure prédiction à celle qui a la pire (par rapport à la prédiction faite sur l'image de base). Pour cela, toutes les images qui ont la même prédiction que celle de base sont ordonnées selon leur pourcentage de façon décroissante, et les autres (souvent rares) sont mises à la fin de la liste. Nous choisissons ensuite le pourcentage d'image que l'on veut garder dans cette liste. Pour cet explainer, cela est fait de façon arbitraire avec une proportion de 70% d'images gardées. Il suffit ensuite de générer l'image finale en coloriant en noir tous les superpixels des images qui ont été gardé. En faisant cela, les zones qui seront en noir dans l'image sont les zones pour lesquelles la prédiction a été très peu influencée lorsqu'elles étaient en noir, ce sont donc des zones peu utiles pour le classifieur. À l'inverse, les zones qui ne seront pas coloriées en noir sont celles qui ont énormément influencée la prédiction une fois coloriées en noir, ce sont donc des parties de l'image importantes pour la prédiction du classifieur.

Problème principale de ce classifieur : le choix des superpixels à mettre en noir est arbitraire, il est donc possible que l'on en sélectionne trop, ou pas assez selon l'image choisie.

Voici un exemple de ce que l'on peut obtenir :

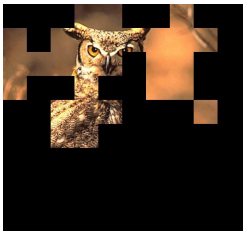


(a) Image généré avec l'explainer arbitraire pour le hibou (GoogleNet)

3.5.2 L'explainer statistique

Pour l'explainer statistique, nous voulions que la sélection des superpixels à mettre en noir soit faite de manière non-arbitraire. Pour cela nous avons pensé à utiliser la moyenne et l'écart type. Nous calculons la moyenne et l'écart type des probabilités associées à la prédiction correcte des images. Les images sélectionnées pour avoir leur superpixel en noir le sont en fonction de leur pourcentage de prédiction. Si ce pourcentage est supérieur à la moyenne calculée moins l'écart type (ou une proportion de l'écart type), ces images-là seront sélectionnées.

Mais là encore, nous pouvons toujours trop en sélectionner, ou pas assez car la proportion de cet écart type est, elle aussi, arbitraire.

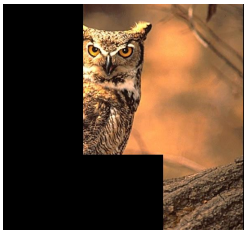


(a) Image générée avec l'explainer statistique pour le hibou (GoogleNet)

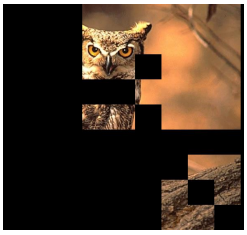
3.5.3 L'explainer de double division

Pour l'explainer de double division, nous avons mis de côté les problèmes de sélection et nous avons voulu tester une nouvelle façon de générer les images pour voir quel impact cela pouvait avoir sur l'image finale, et pour essayer d'améliorer les performances. Pour diminuer le nombre de fois où les images passent par le classifieur, nous avons eu l'idée de d'abord diviser l'image en 9 superpixels et de voir lesquels n'étaient pas utiles pour la prédiction, puis séparer à nouveau en 9 chacun des superpixels jugés utiles, et de ne faire passer dans le classifieur que ces nouveaux superpixels générés. Pour la sélection, nous avons fait le choix d'utiliser la méthode statistique avec la moyenne et l'écart type.

Les résultats n'ont pas été très concluants car le fait de diviser seulement en 9 l'image de base pouvait faire varier la prédiction de base de manière trop conséquente et ainsi garder en couleur trop peu de superpixels.



(a) Phase 1



(b) Phase 2

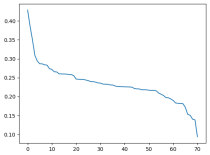
FIGURE 3.6 – Images générées avec l'explainer de double division

L'image (a) est l'image générée dans la phase 1 de l'explainer (avec l'image séparée en 9 parties), l'image (b) est le résultat final.

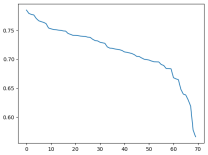
3.5.4 Autre piste possible : méthode du coude

Une autre piste possible serait de sélectionner les superpixels à mettre en noir de manière non arbitraire en traçant la courbe représentant chaque image en fonction de son pourcentage de prédiction, seulement pour les bonnes prédictions.

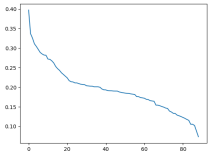
Voici les courbes obtenues pour différentes images 5.3 (avec les images en abscisse et le pourcentage de prédiction en ordonnée).



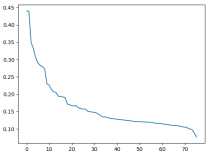
(a) Courbe pour le dauphin



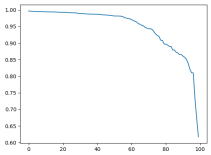
(b) Courbe pour le l'AK47



(c) Courbe pour le hibou



(d) Courbe pour la femme



(e) Courbe pour le dauphin

FIGURE 3.7 – Images générées avec la méthode du coude

On remarque que pour chaque courbe qui représente des images où le classifieur arrive à faire de bonnes prédictions, la proportion à garder est facile à voir (image 2 et 5), car la courbe forme une sorte de "coude". Pour les autres images cela est plus difficile à déterminer car elles ont une

décroissance trop rapide.

L'objectif serait donc d'utiliser un algorithme qui sépare la courbe en deux parties selon ce "coude" afin de minimiser le nombre de superpixels utilisés.

3.6 Validation empirique : comparaison avec LIME

Voici les temps moyens d'exécution obtenus sur les explainers (avec GoogleNet) :

Explainer	LIME	Arbitraire	Statistique	Double Division
Temps moyen	57s	6.6s	6.3s	5.9s

On observe que les temps obtenus par nos explainers sont grandement inférieurs à ceux obtenus par LIME (presque x10). Cela est dû à l'utilisation des superpixels qui permettent de beaucoup diminuer le nombre d'étapes à effectuer par l'algorithme. On peut remarquer que l'explainer de double division a des temps d'exécution légèrement inférieurs aux autres, mais cela au déficit de la qualité du résultat. Lorsque l'on observe visuellement nos résultats,ils ressemblent globalement beaucoup à ceux fait grâce à LIME mais de façon plus grossière. Et en utilisant l'indice de Jaccard, on observe des moyennes aux alentours de 55% (moyenne qui va encore augmenter si on réussit à implémenter l'explainer du coude) entre nos résultats et ceux de LIME. Cela montre que nos résultats sont plutôt proches de ceux de LIME malgré ce temps d'exécution si rapide. Donc, nous pouvons considérer que nos explainers ne sont peut être pas les explainers les plus précis, mais permettent de résoudre les problèmes de lenteur d'exécution qu'avait LIME.

De plus, comme évoqué précédemment, LIME est indéterministe. Ainsi, si on lance plusieurs fois l'explainer sur la même image cela peut donner des résultats différents :

FIGURE 3.8 – Images générées avec LIME

On observe que pour cette image, ce sont deux zones différentes qui sont mis en valeur. Ce n'est pas une différence très grave, même s'il semblerait logique de toujours faire ressortir la même zone : celle qui a le plus grand pourcentage de prédiction. On peut imaginer que sur certains types d'images, cet indéterminisme pourrait faire ressortir des différences plus importantes.

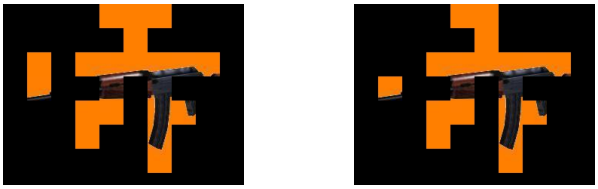


FIGURE 3.9 – Images générées avec l'explainer arbitraire, et l'explainer statistique

L'avantage de nos explainers, en plus de leur vitesse d'exécution, est qu'ils sont déterministes : ils donneront toujours les mêmes résultats. Si une image à deux explications possibles, ils choisiront celle qui est la plus probable.

Chapitre 4

Conclusion

Pour ce rapport, nous avons procédé à l'analyse de LIME. Grâce à cela, et des métriques de comparaison statistiques, nous avons conclu que les explications obtenues via un explainer sont très dépendantes du modèle d'apprentissage automatique qu'il utilise, ce qui peut être un paramètre crucial lors d'une application dans un domaine où les résultats ont des conséquences importantes, tel que l'imagerie médicale.
De plus, les résultats d'un explainer peuvent changer d'une exécution à une autre, qu'il faut différencier de la prédiction du modèle, qui est déterministe.

Nous avons aussi remarqué une vitesse d'exécution très lente. Suite à cette critique, nous avons développé plusieurs explainers qui, bien que moins précis sur le choix des variables importantes pour l'explication, sont bien plus rapides.

Cela montre que les explainers utilisés aujourd'hui peuvent être grandement améliorés, que ce soit par une augmentation de la vitesse d'exécution, ou par un changement de procédé de décision : un explainer modèle-agnostique peut être mauvais si le modèle choisi n'est pas performant.

Chapitre 5

Annexe

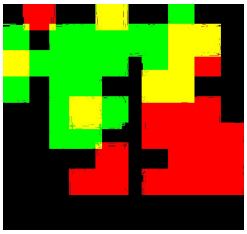
5.1 Influence du flou

L'influence du flou sur les prédictions est un point que nous voulions mettre en valeur : en pratique, il est commun que les données utilisées en apprentissage automatique ne sont pas "propres". Dans le cas des images, il est par exemple possible que certaines d'entre elles soient floues. Nous avons donc aussi lancé notre explainer sur des images floues pour les comparer avec les mêmes images nettoyées.

Voici ici la superposition de couleurs entre une image nette et une image floutée sur l'explainer arbitraire (la partie verte représente l'image nette, le rouge représente l'image floue et la partie commune est en jaune) :

(a) Image nette

(b) Image floue



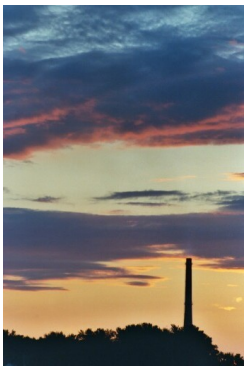
(c) Leur superposition

Pour toutes les images, l'indice de Jaccard moyen est de 52%, ce qui peut paraître faible. Cela peut évidemment être expliqué par le fait que les interprétations sur les images floues ont plus de chance d'être fausses et en particulier sur les images de mauvaise qualité. De plus, pour les images avec des symétries, le flou peut faire ressortir l'autre partie de la symétrie. Ou encore ne faire ressortir qu'une partie de l'image : un avion qui pouvait être reconnu en utilisant le ciel peut, une fois flou, n'être reconnu que par sa forme et non par le ciel qui n'est plus reconnaissable.

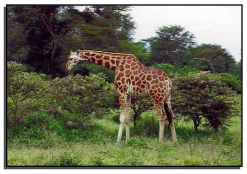
Dans le cas de l'image du hibou, le flou modifie la prédiction qui devient "écureuil", c'est donc pour cela qu'on observe une si grande différence.

5.2 Comparaison Alexnet/GoogleNet

Flop



Top



5.3 Images courbes

Chapitre 6

Références bibliographies

6.1 Cours

https://fr.coursera.org/specializations/deep-learning

https://fr.coursera.org/learn/machine-learning

https://www.academiepro.com/uploads/livres/appr_artif.pdf

6.2 Articles

https://arxiv.org/pdf/1703.04730.pdf

https://arxiv.org/pdf/1802.01933.pdf

https://arxiv.org/pdf/1805.07468.pdf

https://ecmlpkdd2019.org/downloads/paper/572.pdf

https://www.aaai.org/ojs/index.php/AAAI/article/view/5050/4923

https://medium.com/@ODSC/cracking-the-box-interpreting-black-box-machine-learning-models-bc4bdb2b1ed2

https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/

https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5

https://github.com/marcotcr/lime