HANOUZ Akram, JOUVIN Jules, JALAL Zakaria
DIA3

Projet final XAI

Technical Report

Ce projet vise à intégrer **deux systèmes d'explinability IA** (XAI) existants dans **une seule plateforme** interactive capable de traiter à la fois des données audios et des données image.

Table des matières

Conception et intégration spéciales effectuées

L'Application

Tout le code (sauf ce qui concerne Ollama) se trouve dans le fichier `app.py`. Cela comprend l'UI, la prédiction et les méthodes de XAI (la partie LLM est séparée). Les fonctions servant au LLM se trouvent dans `ollama_llm.py`.

Les deux tâches sont implémentées comme des classifieurs d'images.

- **Pour l'audio** :
    - L'utilisateur upload un fichier `.wav` d'un audio, l'audio est ensuite transformé en "mel-spectrogramme" de Mel (utilisant l'échelle des mels qui est une échelle de hauteurs de son). L'image est redimensionnée en 224×224 RGB puis transmise au modèle
    - Si l'utilisateur upload une image de spectrogramme (aucune vérification n'est faite pour vérifier si c'est bien un spectrogramme), elle est simplement redimensionnée comme décrit ci-dessus puis transmise au modèle
- **Pour X-ray** : - L'utilisateur upload une image d'un x-ray thoracique (aucune vérification n'est faite pour vérifier si c'est bien un x-ray), de la même manière, l'image est redimensionnée puis transmise au modèle.

Peu importe la méthode, les deux pipelines renvoient une image redimensionnée, ce qui permet d'avoir un code assez flexible et scallable.

De plus, des modèles spécifiques sont définis pour chaque tâche.

Il est donc assez simple d'ajouter une nouvelle tâche de Computer Vision si elle suit ce schéma.

Classifieur standardisé en 1-logit

Tous les modèles sont des classifieurs binaires et ont pour sortie un seul logit.

Une fonction dédiée (`predict_proba()`) transforme ce logit grâce à l'aide de la fonction sigmoid en probabilités pour l'afficher dans l'UI.

Cette décision « 1 logit au niveau du modèle / 2 classes au niveau de l'API » simplifie l'entraînement tout en prenant en charge les XAI (LIME/SHAP) qui attendent une sortie de probabilité à 2 classes.

Pour l'instant, le code gère uniquement des modèles binaires et ne prend pas en charge la prédiction multi-classe.

Prétraitement

Lors de l'upload, comme évoqué précédemment, l'image (ou le spectrogramme généré) est directement redimensionnée en 224×224.

Nous avons également appliqué la normalisation d'ImageNet. Cette normalisation est devenue un standard en CV. Nous l'avons donc utilisée lors de l'entraînement des modèles et elle doit donc être appliquée lors de l'inférence.

La pipeline pour la génération du spectrogramme audio est comme suivante : **matplotlib → PNG → reload**.

Le rendu visuel (axes, interpolation, palette de couleurs, remplissage) peut fortement dégrader le comportement du modèle ; il doit donc être enlevé en utilisant la pipeline décrite.

Grad-CAM pour les modèles binaires

Nous avons intégré Grad-CAM via la librairie `pytorch-grad-cam`.

1. Un `TwoClassWrapper` transforme le modèle binaire à logit unique en un modèle **2-logit** `[-logit, +logit]` afin que Grad-CAM puisse cibler une classe à l'aide du ciblage multiclasses standard utilisé par Grad-CAM.
2. Les couches cibles sont choisies via `get_gradcam_target_layers()` avec des heuristiques et des fall-back spécifiques au modèle, ce qui réduit les cas de défaillance « blank CAM ».

Performances Streamlit + choix de stabilité

Plusieurs choix d'optimisation ont été pris pour rendre l'utilisation de l'application plus agréable.

- Les poids des modèles sont mis en cache avec `st.cache_resource` (les modèles se chargent une fois par session). Lors de l'upload d'une image, la prédiction est presque instantanée. Cependant, chargé le modèle, même si relativement léger prend quelques secondes. En mettant la fonction de load en cache, cela permet une meilleure expérience utilisateur et moins d'attente.
- Les calculs coûteux et répétés sont mis en cache dans `st.session_state` à l'aide d'une **Clé de hachage d'entrée** :
  - `xai_cache` stocke les images d'explication calculées. Si la page est rechargée cela évite à devoir les re-exécuter (SHAP même si capé ici peut être assez long et donc frustrant)
  - `uri_cache` stocke les URI de données base64 pour le rendu en superposition (voir plus loin).
  - `xai_stats_cache` stocke des statistiques numériques extraites des XAI utilisé pour le LLM
  - `llm_cache` stocke la réponse du LLM qui est la chose la plus longue a généré

- Les fichiers temporaires créés pendant le rendu audio vers spectrogramme sont nettoyés avec `try/finally`.
- Les dépendances facultatives (par exemple, OpenCV) ont un fallback : si `cv2` est manquant, le rendu de superposition Grad-CAM revient à un chemin non OpenCV.

Le cache sauf les poids est supprimé entièrement dès qu'un nouveau fichier est upload.

Explication grâce à un LLM locale facultative (Ollama)

L'application a un panneau **d'explication LLM** facultatif alimenté par un **serveur Ollama local** (par exemple, `gpt-oss:20b`).

Comme très demandant en ressource et nécessitant un set up supplémentaire nous avons décidé de le mettre facultatif.

L'explication est **uniquement textuelle et fondée sur des valeurs numériques**. L'application n'envoie pas d'images/d'audio au modèle.

Elle fournit plutôt :

- Les probabilités prédites + la marge de confiance
- Des **résumés numériques** légers dérivés des sorties XAI (Grad-CAM toujours ; LIME/SHAP réutilisé lorsqu'il a déjà été calculé)

Le llm est aussi chargé de réaliser un petit disclaimer sur l'utilisation des résultats dans des décisions importantes.

Modèle sélectionnés et méthodes XAI

Modèles

Le repo prend en charge plusieurs modèles par tâche, tous récupérés à partir du dossier `./weights/{nom du modèle}.pt`.

Ces modèles sont :

- **DétectiNous avonsudio Deepfake**
  - `audio_vgg16`
  - `audio_resnet50` ○ `audio_mobilenetv2`
- **Tâche pour le cancer du poumon (« opacité pulmonaire »)**
  - `xray_alexnet`
  - `xray_densenet121`

Méthodes XAI

L'interface utilisateur propose quatre méthodes d'XAI pour les deux tâches :

- **Grad-CAM** : carte thermique produite par carte d'activation de classe pondérée par le gradient en utilisant les gradients de la cible de la couche convolutive finale
- **LIME (lime-image)** : explication du choix du modèle mais sur chaque valeur (dit local) - **SHAP (shap.Explainer + Image masker)** : attribution basée sur l'impact positif ou négatif des valeurs.
- **Local Ollama LLM (optional)** : réponse textuelle basé sur la prédiction et le XAI

Remarques supplémentaires sur l'intégration :

- Voir la section précédente pour l'intégration de Grad-Cam.
- LIME et SHAP appellent une fonction de classification qui renvoie des probabilités sous la forme `(N, 2)`.
- La visualisation SHAP utilise le rouge et vert pour distinguer les contributions positives (vert) des contributions négatives (rouge).

Améliorations apportées aux repos originels

Deux projets réunis et une meilleure ergonomie

- Les deux tâches sont réunies dans une seule application.
- Les méthodes XAI sont affichés côte à côte dans l'onglet « Comparer ». Ajout d'une option "overlay" qui permet d'afficher l'image originale à la place de l'image issu d'une méthode XAI. Les informations des méthodes XAI sont donc plus facile à accéder (pas besoin de scroll un seul click suffit) et a comparer.

Meilleure prise en charge d'entrées audio

Le mode audio prend en charge à la fois :

- **Des audios au format `.wav`** (convertis en images spectrogrammes mel)
- **Les images de melspectrogrammes** (`png/jpg/jpeg`).

Lorsqu'une image melspectrogramme est uploadé, l'application essaye de **reconstitué le WAV** pour obtenir un aperçu audible (un avertissement est affiché concernant la fidélité du rendu)

Mise en cache pour une appli plus interactive

Les résultats XAI sont mis en cache, ce qui évite tout recalcul inutile (qui peuvent être long et frustrant) lorsque Streamlit est relancé. Notamment, les données URI mises en cache permettent d'obtenir un comportement de survol stable et réactif sans avoir à recharger les images à plusieurs reprises.

Meilleure ergonomie d'exécution

Même si non recommandé car peut augmenter considérablement le temps d'exécution, si aucun GPU (cuda si disponible) est détecté, sélectionne automatiquement le CPU. Ceci est affiché sur l'UI.

Gestion des fichiers temporaires générés.

Utilisation facultative d'Open-CV supporté.

Explications LLM Locales

Ajout d'une option permettant via un modèle LLM servit sur Ollama.

Voire le dernier paragraphe de la session "Conception et Intégration Spéciale Effectué" pour plus de détail.

Note additionnelle et problèmes rencontrés

- L'application a besoin du dossier `./weights/` contenant les poids des modèles. Ces derniers sont soit à télécharger sur le lien drive partager ou a regénérer via le notebook `training_models.ipynb`.
- PyTorch n'est pas inclus dans le fichier `requirements.txt`. Il est laissé à l'utilisateur le choix de la version de CUDA (ou sans) à téléchargé.
- SHAP méthode est assez lent et nous avons dû caper son nombre d'évaluation pour le garder utilisable facilement pour une démo live.
- L'explication via LLM est optionnelle et demande à l'utilisateur de télécharger et servir un serveur Ollama. L'explication peut également être assez longue à générer dépendant du modèle.

Toutes ces informations sont décrites dans le `README`

Entrainement des modèles

Nous avons décidé pour le projet d'entrainé les modèles puisqu'ils n'étaient pas fournis dans les repos github.

Les deux repos indiquent que les datasets sur lesquels ils ont entrainé leurs modèles sont fornorm (pour deepfakeaudio) et CheXpert (pour Lungcancer). Nous avons donc récupéré fornorm sur le site Yorku (https://bil.eecs.yorku.ca/datasets/) et CheXpert-small sur Kaggle (https://www.kaggle.com/datasets/ashery/chexpert).

Pour CheXpert, nous avons choisi la version small pour gagner en rapidité pour d'entraînement. La version normale est assez lourde et, n'ayant pas pour objectif d'avoir le meilleur modèle en termes de performances, nous nous sommes contentés de ce dernier.

Pour deepfakeaudio, le repo a fourni les notebooks pour convertir les sons en spectrogramme et le code pour train les modèles. L'implémentation des modèles a été fait avec Tensorflow. Or, Tensorflow est cpu-only sur Windows. Nous avons donc converti l'implémentation en pytorch.

Pour Lungcancer detection, rien n'a été fourni concernant l'implémentation des modèles. Le repo mentionne tout de même la création du label « Lung Lesion » à partir des labels « nodule » et « mass ». Cependant, dans le dataset CheXpert , il n'y pas aucun label nodule ou mass, en revanche le label Lung Lesion existe belle et bien. Mais il semble avoir un net problème avec la séparation des classes qu'apparait très déséquilibré ; il n'y a qu'un seul positif pour la partie Test. Il y en a également très peu pour le Train ce qui empêche d'apprécier les performances des modèles. Le plus gros problème étant que les résultats de performance de notre modèle ne sont en aucun cas représentatif de la réelle performance de notre modèle avec un seul cas positif dans le jeu de test. C'est pourquoi nous avons opté pour un autre indicateur : « lung Opacity » qui est bien plus présent et mieux équilibré dans le dataset.

Pour l'implémentation des modèles pour chexpert, nous nous somme notamment aider de ChatGPT et du site geeksforgeeks.

```
Train positives:
Support Devices 116001
Lung Opacity 105581
Pleural Effusion 86187
Edema 52246
Atelectasis 33376
Cardiomegaly 27000
No Finding 22381
Pneumothorax 19448
Consolidation 14783
Enlarged Cardiomediastinum 10798
Lung Lesion 9186
Fracture 9040
Pneumonia 6039
Pleural Other 3523

Test positives:
Support Devices 107
Lung Opacity 126
Pleural Effusion 67
Edema 45
Atelectasis 80
Cardiomegaly 68
No Finding 38
Pneumothorax 8
Consolidation 33
Enlarged Cardiomediastinum 109
Lung Lesion 1
Fracture 0
Pneumonia 8
```

Figure 1 : répartition des labels positif entre le dataset d'entraînement et de test

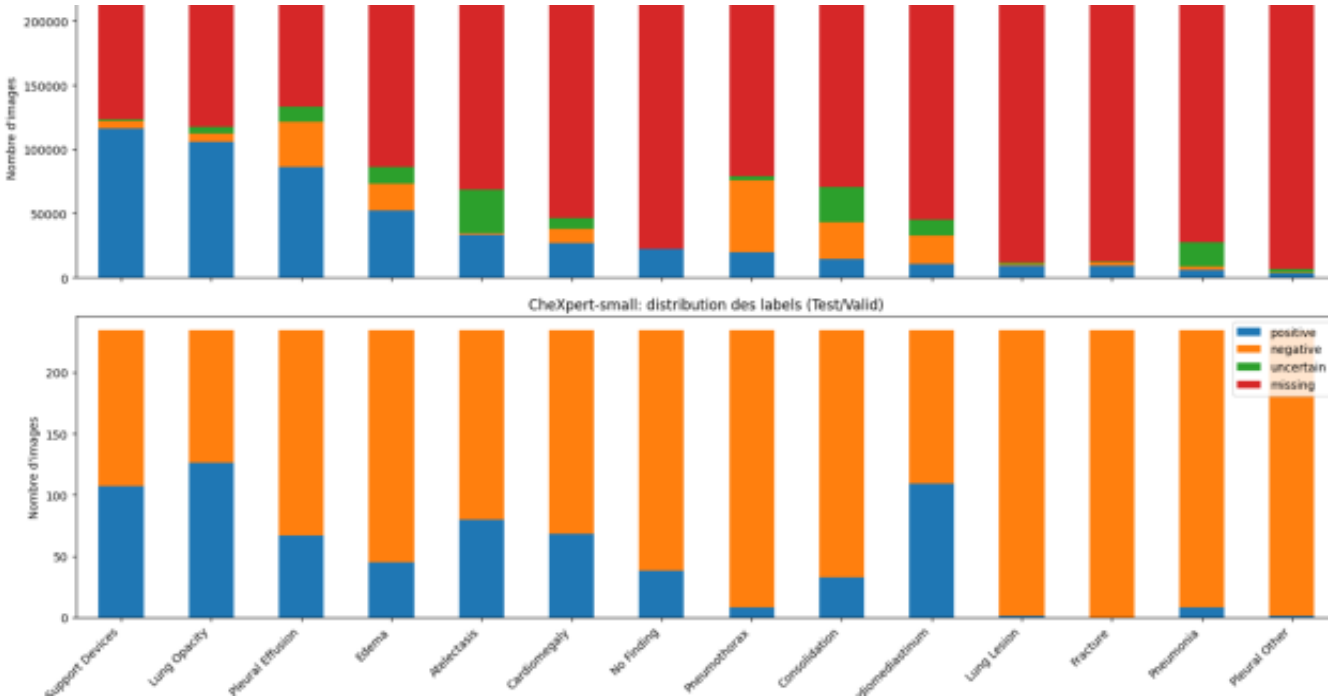

Figure 2 : répartition graphique des labels dans les jeux de données

Les performances de nos modèles ne sont pas terribles notamment comparées à ceux du repo. Cependant, comme dit précédemment, ce n'était pas notre objectif. Ils ont des performances suffisantes pour notre projet orientée Xai et interprétabilité.

Méthodes Xai

Lorsque nous avons analysé les méthodes XAI pour les spectrogrammes, Nous avons été confrontés à une difficulté de compréhension et d'interprétations sur les spectrogrammes. Nous avions du mal à comprendre ce qui était « bien », « pas bien ». Nous avons du mal à savoir si le modèle s'appuyait sur des raisons audios justifiable pour classer les sons en fake ou real.

Nous avons donc cherché sur internet comment interpréter les méthodes Xai pour les spectrogrammes. Nous nous sommes arrêtés sur l'article « Audio Explainable Artificial Intelligence: A Review » (https://spj.science.org) publié dans le journal web *Science Partner Journal*. Cet article aborde la problématique et offre une liste non-exhaustive des méthodes Xai spécifiques à l'audios. Nous avons essayé d'implémenter certaines de ces méthodes mais nous sommes vite rendu compte que cela ne nous avançait finalement pas bien plus. Nous en avons conclu que nous n'avions certainement juste pas les compétences nécessaires et que ces explications ne s'adressent tout simplement pas à nous.

Ainsi cela a mis en évidence que l'interprétation Xai a un aspect personnel : pour un même résultat, deux personnes peuvent avoir des interprétations totalement différentes.

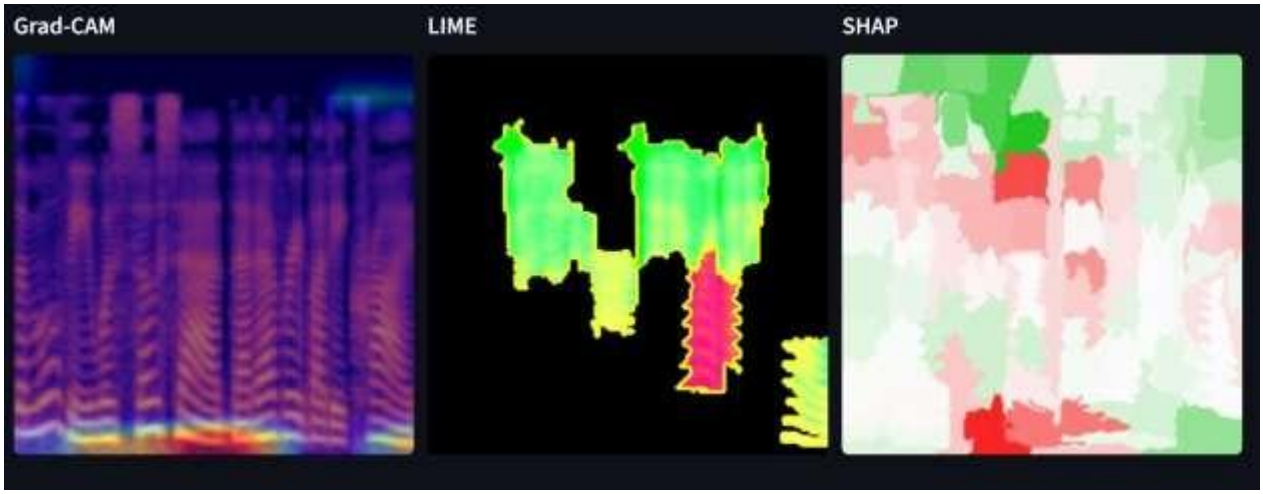

Figure 3 : méthode XAI retenu pour DeepFakeAudio

Pour les images X-ray, ce fut bien plus intuitif pour nous de comprendre et juger ces méthodes.

En effet, nous pouvions voir sur quelles zones de l'image le modèle s'étaient appuyés pour établir sa classification. Or il est clair que si le modèle s'appuie sur des zones en dehors des poumons voire en dehors du corps pour établir la présence de « Lung Opacity ». Alors le modèle a des raisons non médicales pour prédire la classe. Cela permet de prendre du recul sur le modèle : on ne doit pas se fier aveuglément aux prédictions du modèle. Il peut se baser sur des choses surprenantes et non voulu pour faire ses prédictions.

De même, les méthodes Xai offrent une alternative d'utilisation aux modèles.

En indiquant les zones sur lesquels s'est appuyé le modèle pour faire ses prédictions, un médecin peut se concentrer sur ses zones et confirmer s'il y a bien des signes d'une présence d'un « Lung Opacity ». Ainsi Xai permet d'apprécier encore plus l'aide d'une IA dans une décision. XAI permet d'affirmer le rôle d'un modèle comme d'un assistant utile venant en appuie aux décisions d'un professionnel et non comme un décideur « autoritaire » donnant un jugement sans aucune justification. Il permet de faciliter le diagnostic d'un médecin et lui offrir la capacité de valider médicalement la prédiction d'un modèle à usage médicale. En somme, XAI rend les modèles plus « black box » utilisable dans le monde professionnelle.

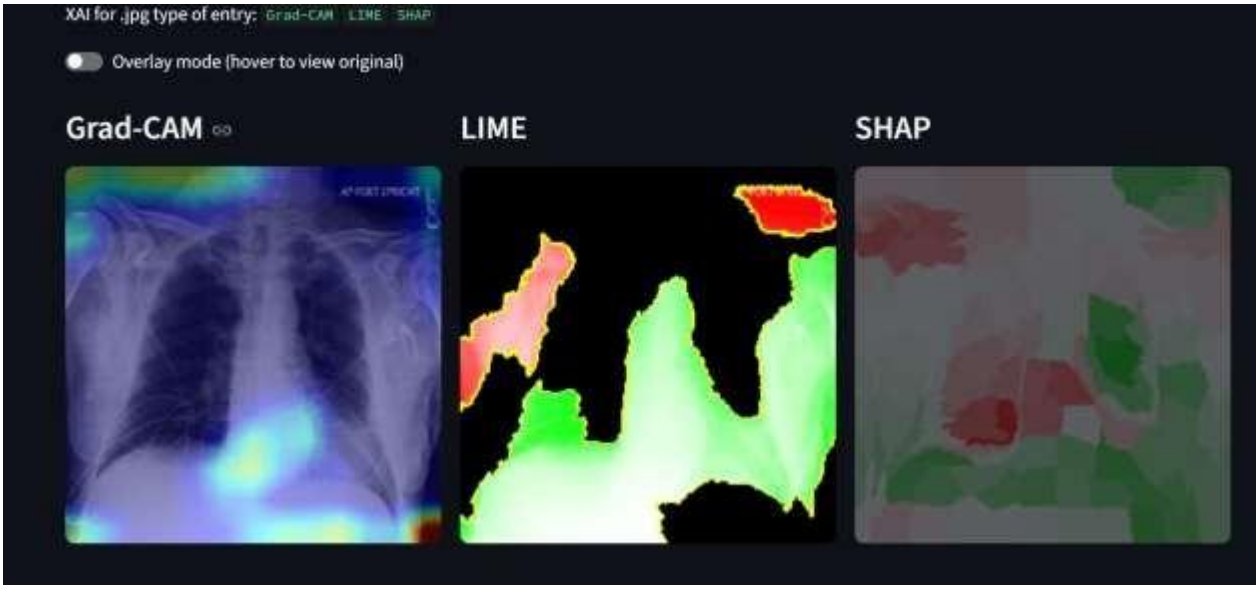

Figure 4 : méthode XAI retenu pour X-ray