RESEAU DE NEURONES CONVOLUTIONELS INTERPRETABLES (XAI)

Document final projet 7

Remi CAZELLES

OpenClassRooms | Octobre 2023

Table des matières

1. Contexte

Etudiant dans le parcours Ingénieur MachineLearning, le stage présenté dans ce rapport rend compte d'algorithmes permettant d'expliquer la prise de décision de modèles de type black-box tel que les réseaux de neurones convolutifs pour la catégorisation d'images.

J'ai effectué ce stage en m'appuyant sur le projet 6 du parcours « Classez des images à l'aide d'algorithmes de Deep Learning » qui vise à attribuer une race de chien pour chacune des photos du dataset StandfordDogsDataset.

J'ai choisi d'étudier les algorithmes XAI en adéquation avec mon projet professionnel visant à employer des méthodes de machine learning pour la santé, notamment pour l'interprétation des prises de décisions à partir d'imageries médicales.

L'objectif de la démarche est de comparer différents types d'algorithme d'explainability afin de d'adpater au mieux es hyperparamètres permettant de classifier les différentes données.

2. Problématique

Comment rendre interprétable par un humain le processus décisionnel d'un algorithme de réseau de neurone profond afin d'améliorer ses performances ?

3. Missions confiées

Le but de la mission est de comparer les algorithmes d'interpretation de modèle black box tels que les réseaux de neurone convolutifs

Mission 1 : entrainer deux algorithmes de classification d'images

Les premiers réseaux de neurones utilisés pour la classification d'images ont été LeNet-5 (développé par Yann LeCun et ses collegues dans les années 1990s) et AlexNet (dévellopé par Alex Krizhevsky en 2012) qui étaient composé chacun de 3 et 5 couches convolutionelles, respectivement. Le second introduisant la fonction d'activation ReLu et la régularisation Dropout permettant de généraliser son application. Les premiers réseaux de neurone profonds sont les architectures VGG (introduit par l'université d'Oxford en 2014) et Inception (introduit par google en 2014). Celles-ci se démarquent de par la profondeur du réseau (jusqu'à 19 couches pour VGG19) et la complexité (Inception utilise des filtres de convolution de tailles différentes en parallèle).

Le développement de ces réseau à conduit à l'émergence de réseaux encore plus profond tels que ResNet (Kaiming He et al. in 2015 utilise 512 couches) et encore plus complexes comme les versions v3 et v4 du modèle développé par google (Inception). Finalement de nouvelles architecture permettent d'améliorer les performances de calcul comme MobileNet (google en 2017) DenseNet (Gao Huang et al. in 2017) et Efficient Net (Tan and Quoc V. Le in 2019).

Mission 1.1 Effectuer un transfert Learning vers des réseaux pré entrainés

Pour effectuer la comparaison des algoithme d'interpretabilité des boites noires, j'ai choisi d'utiliser VGG16 et Inception-V3 qui diffèrent de par leur architecture (photo)

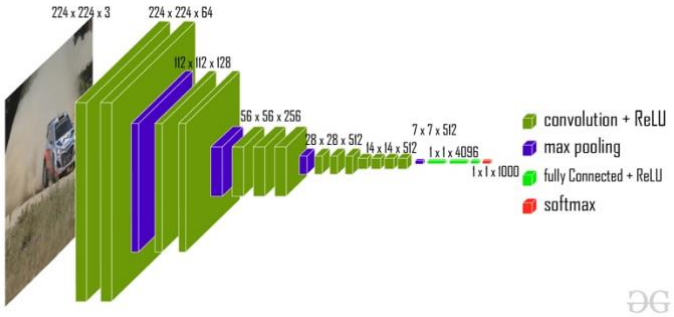


Figure 1 : architecture du réseau de neurones VGG16

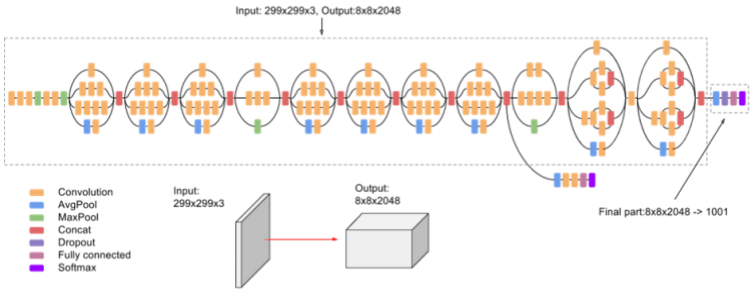


Figure 2 : architecture du réseau de neurones Inception-v3

Mission 1.2 Optimiser les paramètres

Plusieurs algorithmes ont été développé pour l'optimisation des hyperparamètres des réseaux de neurones convolutifs, KerasTuner, HyperOpt ou Optuna en sont 3 exemples.

HyperOpt (Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. TProc. of the 30th International Conference on Machine Learning (ICML 2013), June 2013, pp. I-115 to I-23.) est une alternative au GridSearchCV de scikit-learn. Optuna qui est développé en 2019 permet d'accéder facilement à des visualisation grâce au Dashboard de son API (arXiv:1907.10902). Keras tuner[1] que nous utiliserons ici, développé en 2019, il permet d'utiliser 3 algorithmes de recherche des meilleures valeurs d'hyperparamètres (optimisation bayésienne, hyperbande et aléatoire), il est facilement utilisable pour les modèles de deep learning.

[1] O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & others. (2019). Keras Tuner. .

```python
from tensorflow.keras.applications.inception_v3 import InceptionV3
from keras.layers import Dense, Flatten, Dropout
from keras import Model, regularizers
# Charger modèle pré-entraîné sur ImageNet et sans les couches fully-connected
def model_builder(hp):
    _model = InceptionV3( input_shape=(299, 299, 3), weights='imagenet', include_top=False)
    # Définir les paramètres à optimiser
    hp_units1 = hp.Int('units1', min_value=128, max_value=512, step=256)
    hp_units2 = hp.Int('units2', min_value=128, max_value=512, step=256)
    hp_learning_rate = hp.Choice('learning_rate', values=[1e-3, 1e-4, 1e-5])
    hp_dropout_1 = hp.Choice('dropout1', values=[0.0, 1e-1, 2e-1])
    hp_dropout_2 = hp.Choice('dropout2', values=[0.0, 1e-1, 2e-1])
    hp_dense1_l1 = hp.Choice('dense1_l1', values=[1e-3,1e-4,1e-5])
    hp_dense1_l2 = hp.Choice('dense1_l2', values=[1e-3,1e-4,1e-5])
    hp_dense2_l1 = hp.Choice('dense2_l1', values=[1e-3,1e-4,1e-5])
    hp_dense2_l2 = hp.Choice('dense2_l2', values=[1e-3,1e-4,1e-5])
    # Récupérer la sortie de ce réseau
    x = _model.output
    x = Flatten(name="flatten")(x)
    x = Dense(units=hp_units1, activation='relu',
              kernel_regularizer=regularizers.L1L2(l1=hp_dense1_l1, l2=1e-3),
              kernel_initializer = tf.keras.initializers.GlorotNormal())(x)
    x = Dropout(hp_dropout_1)(x)
    x = Dense(units=hp_units1, activation='relu',
              kernel_regularizer=regularizers.L1L2(l1=hp_dense2_l1, l2=1e-3))(x)
    x = Dropout(hp_dropout_2)(x)
    # Ajouter la nouvelle couche fully-connected pour la classification à 10 classes
    x = Dense(10, activation='softmax')(x)
    # Définir le nouveau modèle
    _model = Model(inputs=_model.input, outputs=x)

    for layer in _model.layers[0:-5]:
        layer.trainable = False

    _model.compile(optimizer=keras.optimizers.Adam(learning_rate=hp_learning_rate),
                loss=keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                metrics=['accuracy'])
    return _model
```

Figure 3 : extrait de code montrant la création du modèle CNN incluant la variable hp qui permet d'optimiser ses hyperparamètres

L'optimisation se déroule en 3 étapes : (1), Écrire une fonction qui crée et renvoie un modèle Keras (2) Utiliser l'argument hp pour définir les hyperparamètres lors de la création du modèle. (3) Initialiser un tuner (Hyperband) en spécifiant l'objectif de sélection des meilleurs modèles. (4) Lancer la recherche.

Pour notre application nous définissons l'objectif comme étant la différence entre les classes prédites et les classes réelles pour le jeu de données de validation (val_loss).

Tous les hyperparamètres peuvent être entrainés, le nombre de couche Dense avec un hyperparamètre entier avec hp.Int('units', min_value=32, max_value=512, step=32), l'utilisation d'une couche Dropout avec hp.Boolean(), le type de fonction d'activation à utiliser avec hp.Choice(), mais aussi le taux d'apprentissage de l'optimiseur avec hp.Float().

Mission 1.3 évaluer les algorithmes de classification

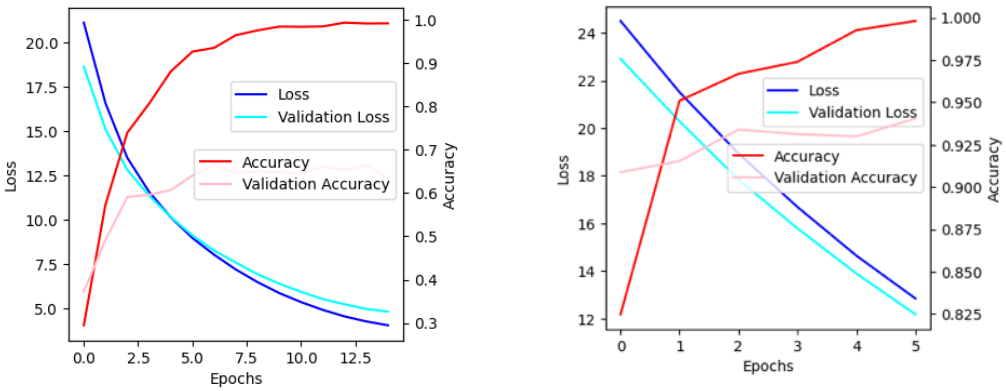


Figure 4 : Historique des valeurs de pertes (Loss, validation Loss) et de précision (Accuracy, val_accruary) lors de l'entrainement par transfert learning du vgg16 (à gauche) et du inception-v3 (à droite).

Les algorithmes dont les paramètres sont optimisés avec la librairie KerasTuner sont ensuite entrainés sur un jeu de données de 210 photos dont chacune est transformée 3 fois. Nous utilisons les rotations, des translations, des zooms et des étirements aléatoires pour enrichir la base de données d'entrainement et le critère d'optimisation choisi est le val_loss

Mission 2 : Utiliser 2 algorithmes XAI pour comprendre le processus décisionnel de VGG16 et Inception-v3

Explicatifs pour les boîtes noires classant les données d'images, triés par type d'explication : Cartes d'attention (saliency map, SM), Attributions de concepts (CA), Contrefactuels (CF) et Prototypes (PR).

- La carte de saillance (e.g. prépondérance) ; C'est une image dans laquelle la luminosité d'un pixel représente le degré de saillance de ce pixel. Formellement, une carte de saillance est modélisée comme une matrice S, dont les dimensions sont les tailles de l'image pour laquelle nous voulons expliquer une décision, et les valeurs sij sont les valeurs de saillance des pixels ij. Plus la valeur de sij est grande, plus la saillance de ce pixel est importante.
- L'attribution de concept ; Contrairement aux modèles carte de saillance qui sont conçus pour fonctionner sur des caractéristiques de bas niveau comme les bords et les lignes d'une image, les méthodes d'attribution de concepts (AC) quantifient, par exemple, la contribution du concept "rayures" à la prédiction de la classe "zèbre". L'explication est basée sur des concepts définis par l'homme. Par exemple étant donné un ensemble d'images appartenant à un concept $[x(1),x(2),...,x(i)]$ avec $x(i) \in C$, les méthodes de CA peuvent être considérées comme une fonction $f:(b,[x(i)]) \rightarrow e$ qui attribue un score e au concept C sur la base des prédictions et des valeurs de la boîte noire b sur l'ensemble $[x(i)]$.
- Les TCAV, Testing with Concept Activation Vectors (Kim et al. 2018) est un explicateur global post-hoc agnostique du modèle pour les classificateurs d'images qui fournit une explication quantitative de l'importance d'un concept pour la prédiction. Dans tcav, chaque concept est représenté par un vecteur particulier appelé Concept Activation Vectors (CAVs) créé par l'interprétation d'un état interne d'un réseau neuronal en termes de concepts adaptés aux humains. ConceptSHAP (Yeh et al. 2020) est une évolution de SHAP qui tente de définir un score d'importance pour chaque

concept découvert de façon automatique. Il trouve l'importance de chaque concept individuel à partir d'un ensemble de m vecteurs de concepts $Cs=\{c1,c2,...,cm\}$ en utilisant les valeurs de Shapley. Le concepthap vise à regrouper les concepts dans certaines régions spatiales cohérentes.

- ICNN, Shen et al. (2021), Images uniquement, Intrinsèque, Global, Spécifique ; Un CNN interprétables dans lequel chaque filtre d'une couche convolutive élevée représente une partie spécifique de l'objet. Les CNN interprétables utilisent les mêmes données d'apprentissage que les CNN ordinaires sans annotations supplémentaires des parties d'objet ou des textures pour la supervision. Le CNN interprétable affecte automatiquement chaque filtre d'une couche convolutionnelle élevée à une partie d'objet au cours du processus d'apprentissage. La connaissance explicite des CNN interprétables peut aider les gens à comprendre leur logique, c'est-à-dire les modèles mémorisés par le CNN pour la prédiction.

Mission 2.1 algorithme LIME

LIME : Ribeiro et al. (2016), Tout type de données, Post-hoc, Local, Agnostique.

Lime est un explicateur local post-hoc agnostique de modèle, il renvoie des saliency map (SM) basés sur la segmentation où chaque segment est appelé superpixel. Après la segmentation de l'image d'entrée, lime adopte une représentation vectorielle à un coup de l'image où l'image d'entrée est un vecteur composé de m uns si m est le nombre de segments identifiés. Ensuite, il crée le voisinage synthétique en remplaçant aléatoirement les superpixels par une couleur uniforme, éventuellement neutre, et en stockant également les représentations à un coup. Le voisinage des images synthétiques est ensuite introduit dans la boîte noire, et la représentation à un instant du voisinage, ainsi que la prédiction de la boîte noire, sont utilisées pour former un modèle linéaire clairsemé. Enfin, les coefficients du modèle linéaire sont utilisés pour déterminer l'importance des superpixels. Des exemples d'explications renvoyées par lime sont présentés dans la deuxième ligne de la figure suivante. Un aspect critique pour obtenir une bonne explication avec cette approche est le choix de l'algorithme de segmentation et de ses hyperparamètres. **Le parametre « model_regressor » de la méthode explain instance permet de choisir n'importe quel regresseur lineaire du module Sklearn.**

Mission 2.2 algorithme GradCAM

Grad-cam, Selvaraju et al. (2020), Image uniquement, Post-hoc, Local, Spécifique.

GRAD-CAM est un explicateur local post-hoc spécifique au modèle pour les données d'image. Il utilise les informations de gradient qui circulent dans la dernière couche convolutive d'un CNN pour attribuer des valeurs de saillance à chaque neurone pour une décision particulière.

Les couches convolutives retiennent naturellement les informations spatiales dans les couches entièrement connectées, de sorte que l'on suppose que les dernières couches convolutives présentent le meilleur compromis entre la sémantique de haut niveau et les informations spatiales détaillées. Pour créer la carte de saillance, grad-cam prend les cartes de caractéristiques créées à la dernière couche du réseau convolutif. Ensuite, il calcule le gradient d'une sortie d'une classe particulière yc pour chaque activation de carte de caractéristiques k, c'est-à-dire $\partial yc/\partial ak$. Cette équation renvoie un tenseur de dimensions [k, v, u] où k est le nombre de cartes de caractéristiques et u et v sont la hauteur et la largeur de l'image. grad-cam calcule la valeur de saillance pour chaque carte de caractéristiques en regroupant les dimensions de l'image. La carte thermique finale est calculée comme une somme pondérée de ces valeurs. Il en résulte une carte thermique grossière de la même taille que les cartes de caractéristiques convolutives. Une technique d'échantillonnage ascendant est appliquée au résultat final pour produire une carte de la dimension initiale de l'image. La figure suivante montre clairement ce style de carte thermique qui mettent en évidence des parties très différentes de l'image par rapport aux autres méthodes.

Mission 3 interprétation des résultats pour un décision correcte des algorithmes

Mission 3.1 LIME

Les algorithmes VGG16 et Inception-v3 sont utilisés avec LIME pour comprendre quels superpixels contribuent le plus à la prise de décision des algorithmes pour l'attribution à une classe. Les prédictions pour la classe Afghan hound sont les meilleures pour les deux algorithmes. Alors que VGG16 peine parfois à identifier la bonne classe (probabilités < 60%), inception-v3 attribuent avec une grande confiance les images à la bonne classe (probabilités > 99%).

Figure 5 : résultat de la classification d'images de la classe Afghan hound pour VGG16 (à gauche) et inception-v3 (à droite)

Les superpixels identifiés pour chacun des algorithmes sont très similaires avec LIME, on peut remarquer que lorsque VGG16 peine à identifier la classe avec précision les superpixels n'englobent pas les parties spécifiques de l'animal (tête, oeils) alors que inception-v3 y parvient.

Une étude plus approfondie des superpixels contribuant négativement à l'attribution de la bonne classe montre que les détails de l'environnement tendent à impacter négativement la prise de décision lorsqu'on utilise VGG16.

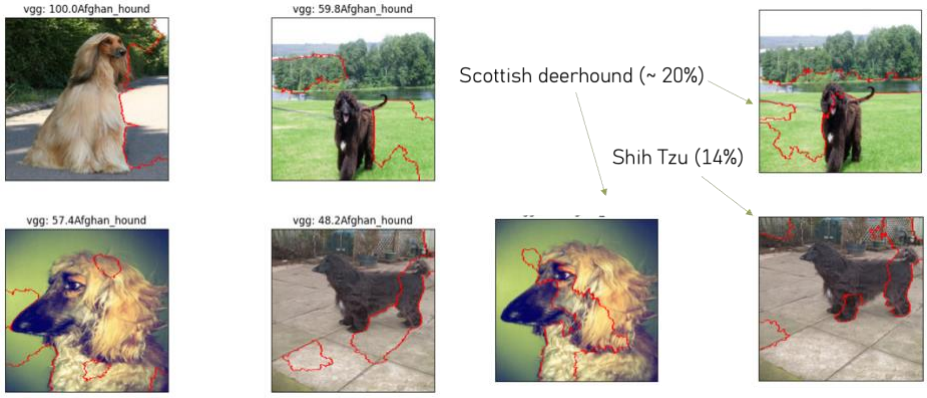


Figure 6 : Superpixels impactant négativement la prise de décision pour l'attribution de la bonne classe (afghan hound, à gauche). Superpixels impactant positivement la prise de décision pour la mauvaise classe (à droite).

Avec l'algorithme VGG16, les images de la classe Afghan hound qui auraient pu être attribuées à une classe différente identifient avec précision les détails morphologiques mais les attribuent à la mauvaise classe.

Mission 3.2 GradCam

Comme GradCam utilise les gradients des différentes couche de convolution, il peut être utilisé pour voir ou se porte l'attention du réseau de neurones au fur et a mesure de l'extraction des caractéristiques. Grad-CAM calcule les gradients de la prédiction de classe par rapport aux cartes de caractéristiques de la dernière couche convolutive, calcule la moyenne pondérée des gradients positifs. La carte thermique obtenue peut alors être utilisée pour visualiser les régions de l'image qui sont les plus importantes pour la prédiction de la classification. Dans l'ensemble, Grad-CAM+ est une méthode d'explicabilité plus puissante et plus polyvalente que Grad-CAM. Elle est particulièrement utile pour les applications où une localisation précise est importante, comme la détection et la segmentation d'objets.

Figure 7 : Utilisation de algorithme Grad-CAM (en haut) et Grad-CAM+ (en bas) pour visualiser l'attention des modèles VGG16 (à gauche) et Inception-v3 (à droite) lors de l'attribution de la classe Afghan Hound.

Les réseaux VGG16 et Inceptoin-v3 diffèrent beaucoup de par leur architecture, nous choisissons d'analyser les dernières couche convolutives pour chaque block de VGG16 et les dernières couches de convolution du réseau de neurone inception-v3 (voir annexe 1)

Dans le cas du réseau de neurones VGG16, il est très clair pour Grad-CAM que l'attention commence à être porté sur des détails de bas niveaux (bords) avant d'identifier les caracteristiques morphologiques (oreille). Grad-CAM+ à lui une attention plus diffuse sur l'ensemble de l'image avant de recentrer son attention sur des caractéristiques plus nombreuses (museaux + oreille + pelage).

Figure 8 : evolution de l'attention du réseau VGG16 pour l'attribution de la classe Afghan Hound à une image représentant un Afghan Hound. « Layer » représente la dernière couche de convolution de chacun des 5 block du VGG16.

Dans le cas du réseau de neurones Inception-v3, il semble qu'en entrée du réseau aucune features n'a pu être clairement identifiée. La visualisation de l'attention sue les derniers couches

du réseau montre que l'étape de MaxPooling à pour effet de changer l'attention du réseau sur différente parties de l'image. L'algorithme Grad-CAM+ peine à identifer des régions d'importance ou bien celle-ci sont extrement petites et localisée (pas de visuel de la carte de saillence)

Mission 3.2 analyse des erreurs avec LIME et Grad-CAM

Les algorithme d'explainability de réseau de neuronnes peuvent servir à identifier les biais de décisions afin d'optimiser le processus de décision en enrichissant la base de donnée avec des

Pour VGG16: Prédiction classe Scottish deerhoud (45.2%) alors que classe réelle est Irish wolfhound (45%). L'algorithme peine à trancher entre deux classes. LIME permet de mettre en évidence que le modèle à mal catégorisé le pelage du Scottish Deerhound alors qu'il a bien identifié les jambes du Irish et s'est servi d'element du contexte (arrière-plan). GradCAM montre l'erreur d'interprétation de la gueule et des jambes attribuant ainsi la mauvaise classe avec certitude alors que l'attention porté au pelage pour la bonne classe n'a pas eu un poid assez important dans la prise de décision.

Pour Inception-v3: le modèle prédit avec certitude la mauvaise classe Scottish deerhoud (96.5%) alors que la classe réelle Irish wolfhound n'est que très peu identifiée (3.5%). LIME montre la mauvaise catégorisation de détails morphologiques (l'oeil/oreille) et identifie tout comme pour VGG16 ls jambes à la bonne classe. GradCAM montre une attention très générale pouvant expliquer l'erreur de prédiction. Aucun élément du vrai target n'a été identifié car l'attention est portée au coin en bas à gauche de l'image

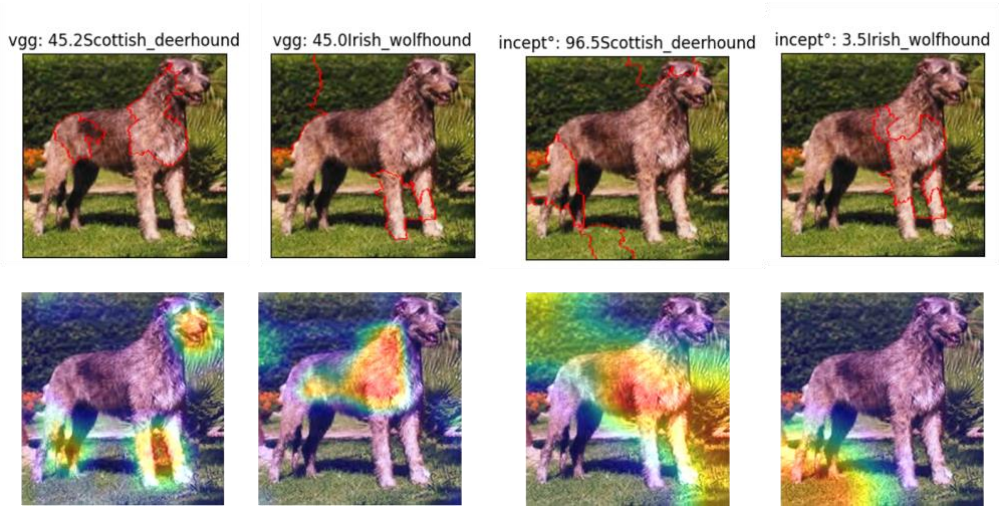


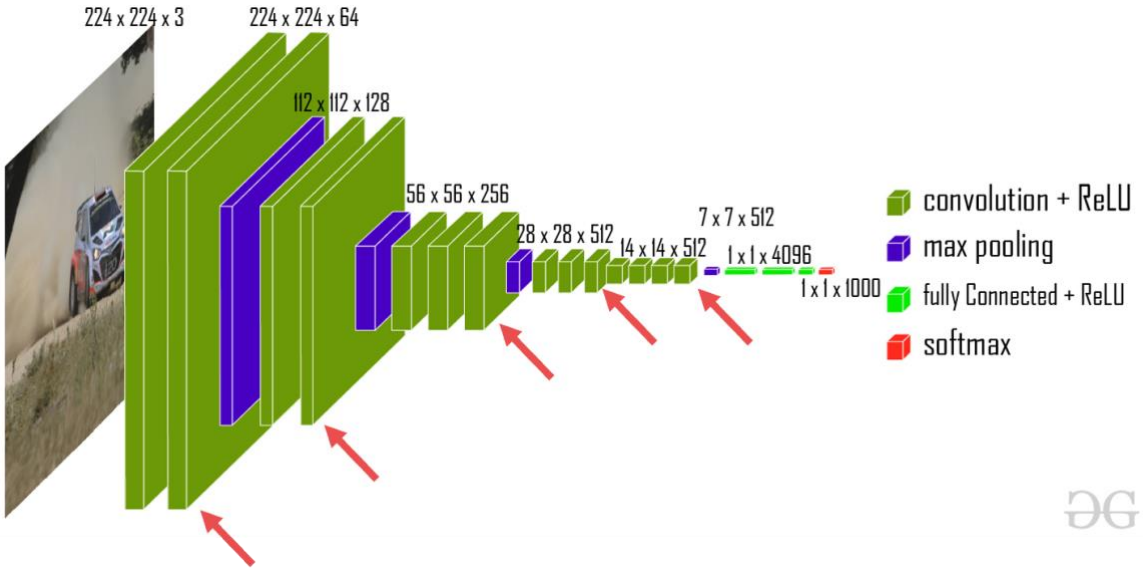
Figure 9 : Analyse des cartes de saillance avec l'algorithme LIME (en haut) et l'algorithme Grad-CAM (en bas) pour un réseau de neurones VGG16 (à gauche) et Inception-v3 (à droite) dans la prise de décision d'attribution de la classe Scottish deerhound et Irish wolfhound.

4. Bilan

En conclusion, l'analyse de la prise de décision par les réseaux de neurones, en particulier les CNN, est une tâche complexe qui nécessite des outils appropriés pour obtenir des ‘insights’ précieux. La carte de saillance se révèle être un instrument très pratique pour cette fin, en offrant une vue d'ensemble des zones d'intérêt dans l'image. L'algorithme LIME, de son côté, permet de révéler les détails structuraux qui sont pris en compte par le réseau de neurones pour effectuer ses prédictions, tandis que l'algorithme Grad-CAM peut nous éclairer sur la trajectoire de l'attention du réseau.

Cependant, il est important de noter que chaque algorithme d'explicabilité a ses propres forces et limites. C'est pourquoi, pour une explication complète de la prise de décision d'un réseau de neurones formé pour une tâche de classification, la combinaison de plusieurs méthodes d'explication semble être la meilleure option. En agrégeant les informations fournies par différents algorithmes, on obtient une perspective plus riche et complète de la façon dont le réseau de neurones effectue ses prédictions. Cette approche permet d'améliorer la transparence des modèles d'apprentissage profond et de renforcer la confiance dans leurs décisions, tout en aidant les chercheurs et les praticiens à mieux comprendre le fonctionnement interne de ces systèmes complexes.

Annexe 1
Couches de convolution analysée avec les algorithmes Grad_CAM et Grad-CAM+ pour l'architecture VGG16



Annexe 2
Couches de convolution analysée avec les algorithmes Grad_CAM et Grad-CAM+ pour l'architecture Inception V3