

UNIVERSITÉ NATIONALE DU VIETNAM, HANOI
ÉCOLE INTERNATIONALE

=====



ALASSANI Abdoul Saib Gaston

**EXPLICATION DE PRÉDICTIONS
D'UN RÉSEAU DE NEURONES**

**GIẢI THÍCH DỰ ĐOÁN
CỦA MẠNG NƠ-RON**

Programme : Systèmes Intelligents et Multimédia

Code : 8480201.02

Encadrant : Pr. Arnaud Sallaberry

MASTER SYSTÈMES INTELLIGENTS ET MULTIMÉDIA

Hanoi – 2026

Attestation sur l'honneur

Je déclare par la présente que ce mémoire a été réalisé par moi-même sous la direction et la supervision du Pr. Arnaud Sallaberry ; et que les travaux et les résultats qu'il contient sont conformes à la réalité de l'auteur et respectent l'éthique de la recherche. Les données et figures présentées dans ce mémoire sont issues de mes propres analyses, commentaires et évaluations, issues de diverses sources et sont dûment mentionnées dans la partie référence.

De plus, les autres commentaires, analyses et données utilisés par d'autres auteurs et organisations ont été explicitement mentionnés.

J'assume l'entière responsabilité de toute fraude détectée dans mon mémoire.

Hanoi, le 27/05/2026

(Auteur)



ALASSANI Abdoul Saib Gaston

Approuvé par

ENCADRANT

(Signature et nom complet)



Pr. Arnaud Sallaberry

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué au bon déroulement de ce stage et à la réalisation de ce rapport.

Tout d’abord, je remercie chaleureusement Luc Etienne Pommé, postdoctorant au LIRMM et membre de l’équipe ADVANSE, pour son encadrement quotidien, sa patience et sa rigueur scientifique. Ses réunions hebdomadaires, ses retours détaillés et son expertise en intelligence artificielle explicable, notamment en tant qu’auteur de la méthode H²O, ont été déterminants pour la réussite de ce projet.

Je remercie Arnaud Sallaberry, Professeur des Universités en Informatique au LIRMM (Université de Montpellier, CNRS), mon tuteur de stage, pour la confiance qu’il m’a accordée, ses orientations stratégiques et ses connaissances approfondies en visualisation d’information.

Je remercie également Ho Tuong Vinh, mon tuteur académique à l’Institut Franco-phone International (IFI) de l’Université Nationale du Vietnam, Hanoï, pour son suivi régulier et ses recommandations pertinentes qui ont enrichi ma réflexion.

Mes remerciements vont aussi à l’ensemble de l’équipe ADVANSE du LIRMM pour leur accueil chaleureux et leur collaboration.

Enfin, je remercie ma famille et mes proches pour leur soutien constant tout au long de mes études et de ce stage.

Résumé

Mots-clés : Intelligence artificielle explicable, vision par ordinateur, stratégies de masquage, benchmark, LIME, RISE, H2O, HiPe, évaluation

L’explicabilité des modèles d’apprentissage profond est devenue un enjeu central, tant pour la confiance des utilisateurs que pour la conformité réglementaire (RGPD, AI Act européen). En vision par ordinateur, les méthodes d’explication par occultation produisent des cartes de saillance en masquant des régions de l’image et en observant la variation de la prédiction du modèle. Or, masquer suppose de remplacer les pixels par d’autres valeurs : ce choix, appelé stratégie de masquage, est rarement explicité et diffère d’une méthode à l’autre. Les comparaisons publiées dans la littérature confondent ainsi l’effet de la méthode et celui de la stratégie, sans possibilité de les dissocier.

Ce stage, réalisé au sein de l’équipe ADVANSE du LIRMM, avait pour objectif d’isoler et de quantifier l’impact de la stratégie de masquage sur les performances des méthodes par occultation. Pour cela, nous avons conçu un framework de benchmark unifié intégrant quatre méthodes de référence (LIME, H2O, RISE et HiPe) sous une interface commune, auxquelles nous avons appliqué sept stratégies de masquage standardisées, dont `kmeans_colors`, proposée dans le cadre de ce travail. L’évaluation repose sur huit métriques couvrant la fidélité, la corrélation, la localisation et la complexité des explications, calculées sur le dataset PASCAL VOC 2007 avec un modèle ResNet-18, soit trente combinaisons méthode-stratégie.

Les résultats montrent que le choix de la stratégie influence fortement les cartes produites : pour une même méthode, le Deletion AUC varie jusqu’à 41 %, une variabilité parfois supérieure à l’écart observé entre méthodes. La stratégie `kmeans_colors` obtient les meilleures performances de localisation et figure parmi les meilleures en corrélation, et une tension systématique apparaît entre fidélité et corrélation selon les familles de métriques. Ces travaux confirment que la stratégie de masquage est un facteur aussi déterminant que le choix de la méthode, et plaident pour sa standardisation.

Abstract

Keywords : Explainable AI, computer vision, masking strategies, benchmark, LIME, RISE, H2O, HiPe, evaluation

Explainability of deep learning models has become a central concern, both for user trust and for regulatory compliance (GDPR, European AI Act). In computer vision, occlusion-based explanation methods produce saliency maps by masking regions of the image and observing the resulting change in the model’s prediction. However, masking requires replacing pixels with other values : this choice, known as the masking strategy, is rarely made explicit and differs from one method to another. Published comparisons therefore conflate the effect of the method with that of the strategy, with no way to disentangle them.

This internship, carried out within the ADVANSE team at LIRMM, aimed to isolate and quantify the impact of the masking strategy on the performance of occlusion-based methods. To this end, we designed a unified benchmark framework integrating four reference methods (LIME, H2O, RISE, and HiPe) under a common interface, to which we applied seven standardized masking strategies, including `kmeans_colors`, proposed as part of this work. Evaluation relies on eight metrics covering faithfulness, correlation, localization, and explanation complexity, computed on the PASCAL VOC 2007 dataset with a ResNet-18 model, amounting to thirty method-strategy combinations.

The results show that the choice of strategy strongly influences the maps produced : for a given method, Deletion AUC varies by up to 41 %, a variability that sometimes exceeds the gap observed between methods. The `kmeans_colors` strategy achieves the best localization scores and ranks among the best for correlation, and a systematic tension emerges between faithfulness and correlation depending on the metric family. These findings confirm that the masking strategy is as decisive a factor as the choice of method itself, and call for its standardization.

Table des matières

Attestation sur l'honneur	i
Remerciements	ii
Résumé	iii
Abstract	iv
Liste des Acronymes	xi
1 Introduction	1
1.1 Contexte général	1
1.2 Problématique	3
1.3 Objectifs et contributions	4
1.4 Organisation du rapport	6
2 État de l'Art	7
2.1 Explicabilité en vision par ordinateur	7
2.1.1 Enjeux de l'explicabilité	7
2.1.2 Taxonomie des approches	8
2.2 Méthodes d'explication par perturbation	9
2.2.1 LIME	9
2.2.2 H2O	11
2.2.3 RISE	12
2.2.4 HiPe	14
2.2.5 Comparaison synthétique	15
2.3 Stratégies d'occultation	15

2.3.1	Pourquoi remplacer les pixels ?	15
2.3.2	Stratégies de remplacement simples	16
2.3.3	Stratégies de remplacement élaborées	17
2.3.4	Synthèse	17
2.4	Métriques d'évaluation	18
2.4.1	Métriques de fidélité	18
2.4.2	Métriques de corrélation	19
2.4.3	Métriques de localisation et de complexité	19
2.4.4	Synthèse des métriques	20
2.5	Positionnement	20
3	Méthodologie et Implémentation	23
3.1	Architecture du framework	23
3.1.1	Principes de conception	23
3.1.2	Vue d'ensemble	23
3.2	Intégration des méthodes	25
3.2.1	Adaptation de LIME	25
3.2.2	Adaptation de H2O	26
3.2.3	Adaptation de RISE	26
3.2.4	Adaptation de HiPe	27
3.2.5	Support multi-classe	27
3.3	Standardisation des stratégies d'occultation	27
3.4	Système d'évaluation	28
3.5	Infrastructure technique	28
3.5.1	Technologies	28
3.5.2	Configuration et reproductibilité	28
3.5.3	Système de checkpoints	29
3.5.4	Génération des rapports	29
4	Résultats Expérimentaux	30
4.1	Protocole expérimental	30
4.1.1	Dataset et modèle	30
4.1.2	Combinaisons évaluées	30
4.2	Analyse par méthode	31

4.2.1	LIME	31
4.2.2	H2O	32
4.2.3	RISE	33
4.2.4	HiPe	34
4.3	Observations transversales	35
4.3.1	Synthèse inter-méthodes	35
4.3.2	Tension fidélité/corrélation	36
4.3.3	Analyse temporelle	37
4.4	Résultats qualitatifs	37
4.4.1	Comparaison visuelle des méthodes	37
4.4.2	Impact visuel des stratégies	38
4.4.3	Cas multi-classe	39
4.5	Synthèse et recommandations	40
5	Conclusion	43
5.1	Bilan	43
5.2	Limites	44
5.3	Perspectives	45
5.4	Apports personnels	45
	Références Bibliographiques	46
A	Annexes	51
A.1	Extraits de code source	51
A.1.1	Stratégie K-means Colors	51
A.1.2	Calcul du Deletion AUC	52
A.2	Classes du dataset PASCAL VOC 2007	54
A.3	Résultats détaillés	54
A.4	Exemple de rapport HTML généré	55
B	Gestion de Projet et Retour d'Expérience	57
B.1	Organisation du travail	57
B.2	Planning du stage	59
B.3	Difficultés rencontrées et solutions	59
B.3.1	Prise en main du code existant	60

B.3.2	Évolution du périmètre du projet	60
B.3.3	Coexistence TensorFlow/PyTorch	60
B.3.4	Définition progressive des métriques	61
B.3.5	Variabilité des résultats LIME	61
B.4	Compétences acquises	61

Table des figures

1.1	Le problème de la boîte noire	2
1.2	Méthode et stratégie d’occultation sont deux choix distincts	3
2.1	Principe de l’approximation locale de LIME	10
2.2	Vue d’ensemble de la méthode H2O	11
2.3	Vue d’ensemble de la méthode RISE	13
2.4	Perturbation hiérarchique de HiPe	14
2.5	Illustration du Deletion AUC et de l’Insertion AUC	19
3.1	Architecture générale du framework de benchmark	24
3.2	Pipeline expérimental du framework de benchmark.	25
4.1	Comparaison des cartes de saillance par méthode	38
4.2	Impact des stratégies d’occultation sur les cartes de saillance (HiPe)	39
4.3	Explications spécifiques par classe (H2O)	40
A.1	Rapport HTML – En-tête et tableau récapitulatif	56
A.2	Rapport HTML – Section de visualisation par image	56

Liste des tableaux

2.1	Comparaison des méthodes d'explication étudiées	15
2.2	Caractéristiques des stratégies d'occultation	17
2.3	Récapitulatif des métriques d'évaluation	20
4.1	Combinaisons méthode-stratégie évaluées	31
4.2	Impact des stratégies d'occultation sur LIME	32
4.3	Impact des stratégies d'occultation sur H2O	33
4.4	Impact des stratégies d'occultation sur RISE	34
4.5	Impact des stratégies d'occultation sur HiPe	35
4.6	Performances moyennes par méthode (moyennées sur toutes les stratégies)	36
4.7	Temps d'exécution moyen par image (secondes)	37
4.8	Impact des stratégies d'occultation (moyennes sur les 4 méthodes) . .	41
A.1	Les 20 classes de PASCAL VOC 2007	54
A.2	Résultats détaillés par combinaison méthode-stratégie (53 paires image- classe)	54
B.1	Phases du stage	59

Liste des Acronymes

CNN Convolutional Neural Network. [1](#)

RGPD Règlement Général sur la Protection des Données. [1](#), [7](#)

XAI eXplainable Artificial Intelligence. [2](#)

Chapitre 1

Introduction

1.1 Contexte général

L'intelligence artificielle a connu des avancées considérables au cours de la dernière décennie, en particulier dans le domaine de la vision par ordinateur. Les réseaux de neurones convolutifs ([Convolutional Neural Networks \(CNNs\)](#)) ont transformé la classification d'images, c'est-à-dire la tâche qui consiste à associer une ou plusieurs étiquettes de classe à une image donnée, au point de surpasser les performances humaines sur certains benchmarks de référence (HE et al., [2015](#)). Des architectures comme ResNet (HE et al., [2016](#)) ou VGGNet (SIMONYAN & ZISSERMAN, [2015](#)), entraînées sur de vastes corpus d'images naturelles tels qu'ImageNet (DENG et al., [2009](#)) et PASCAL VOC (EVERINGHAM et al., [2010](#)), atteignent des précisions remarquables en classification multi-label, où plusieurs objets peuvent coexister dans une même image.

Cependant, cette performance s'accompagne d'une complexité croissante qui rend ces modèles largement opaques. Un réseau comme ResNet-18 contient plus de onze millions de paramètres, formant ce que l'on qualifie communément de « boîte noire ». Cette opacité pose des problèmes concrets : en médecine, un praticien hésitera à suivre une recommandation diagnostique dont il ne comprend pas les fondements ; dans l'industrie, un modèle déployé peut apprendre des corrélations parasites sans que les développeurs ne le détectent ; sur le plan réglementaire, le [Règlement Général sur la Protection des Données \(RGPD\)](#) européen exige dans certains cas un « droit à l'explication » pour les décisions automatisées (EUROPEAN PARLIAMENT AND COUNCIL OF THE EUROPEAN UNION, [2016](#)).

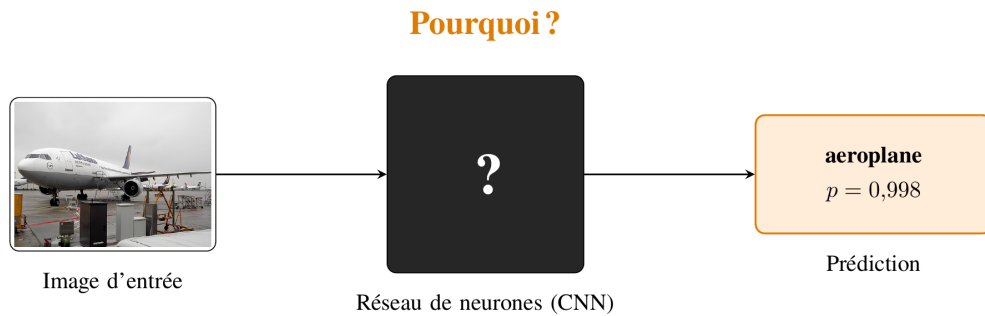


FIGURE 1.1 – Le problème de la boîte noire.

Le réseau attribue à cette image la classe `aeroplane` avec une probabilité de 0,998, mais ne fournit aucune indication sur les régions de l'image qui ont motivé cette décision. Les méthodes d'explicabilité visent précisément à produire cette indication manquante.

Face à ces enjeux, le domaine de l'intelligence artificielle explicable ([eXplainable Artificial Intelligence \(XAI\)](#)) s'est développé pour fournir des outils permettant de comprendre et d'interpréter les décisions des modèles (ARRIETA et al., 2020). Parmi la diversité des approches existantes, qui vont des méthodes par gradient aux méthodes par propagation de pertinence comme le recense le survey de LA ROSA et al. (2023), ce travail se focalise sur les méthodes d'explication par occultation. Ces méthodes génèrent des cartes de saillance en observant comment la prédiction du modèle évolue lorsque différentes régions de l'image sont masquées. Elles présentent l'avantage d'être agnostiques au modèle : elles traitent le classifieur comme une boîte noire et ne nécessitent pas d'accéder à ses paramètres internes.

Expliquer une prédiction par occultation suppose toutefois de trancher une question préalable, rarement explicitée : par quoi remplacer les pixels que l'on occulte ? Ce mémoire aborde donc l'explication de prédictions d'un réseau de neurones sous cet angle précis. Plutôt que de proposer une nouvelle méthode d'explication, il prend pour objet une décision de conception commune à toutes les méthodes par occultation, la stratégie de remplacement des pixels masqués, et mesure ce qu'elle change aux explications obtenues. C'est à ce titre que l'évaluation expérimentale occupe une place centrale dans la suite du document : elle est le moyen d'établir l'influence de ce facteur, et non une fin en soi.

1.2 Problématique

Les méthodes d'explication par occultation partagent un principe commun : masquer des régions de l'image et observer l'effet sur la prédiction du modèle. Or, on ne peut pas simplement « supprimer » des pixels d'une image sans les remplacer par d'autres valeurs (contrairement aux données tabulaires où il est possible de retirer une variable). Ce remplacement, que l'on appelle stratégie de masquage (ou d'occultation), est un choix méthodologique souvent implicite. LIME (RIBEIRO et al., 2016) remplace par défaut les superpixels masqués par leur couleur moyenne, RISE (PETSUK et al., 2018) atténue les pixels au moyen d'un masque basse résolution interpolé (les pixels masqués ne sont donc pas tous remplacés par du noir), H2O (POMMÉ et al., 2023) échantillonne plusieurs couleurs dans un cube RGB restreint. Chaque méthode propose ainsi sa propre stratégie par défaut, et les comparaisons publiées dans la littérature mêlent donc l'effet de la méthode et l'effet de la stratégie, sans possibilité de les dissocier.

La figure 1.2 illustre ce point : à région occultée identique, quatre stratégies de remplacement produisent quatre images perturbées très différentes, que le classifieur ne percevra pas de la même manière. Le choix du remplacement est donc un facteur indépendant de la méthode d'explication qui a sélectionné la région.

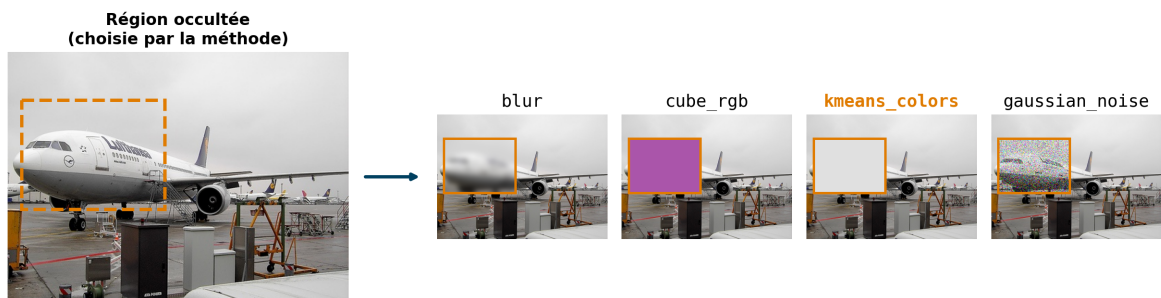


FIGURE 1.2 – Méthode et stratégie d'occultation sont deux choix distincts.

À gauche, la région à occulter, choisie par la méthode d'explication. À droite, la même région remplie par quatre stratégies différentes du framework. Les images perturbées sont obtenues avec le code du framework développé durant ce stage, sur une image de PASCAL VOC 2007.

Ce travail répond ainsi à deux questions de recherche :

QR1. Quel est l'impact de la stratégie de masquage sur les performances de chaque méthode d'explication, à méthode constante ? Autrement dit, pour une méthode

donnée, les résultats changent-ils significativement selon que l'on remplace les pixels masqués par du noir, du flou, du bruit, ou des couleurs issues de l'image ?

QR2. En moyenne sur les méthodes, existe-t-il des stratégies systématiquement meilleures que d'autres ? En particulier, les stratégies proches de la distribution d'entraînement du modèle (in-distribution) surpassent-elles les stratégies hors distribution ?

Cette problématique est d'autant plus subtile que les métriques d'évaluation elles-mêmes ne sont pas neutres. Les métriques d'insertion et de deletion, parmi les plus utilisées pour évaluer les cartes de saillance, reposent elles aussi sur une stratégie de masquage pour supprimer ou révéler progressivement les pixels. On évalue donc des stratégies de masquage via des métriques qui utilisent elles-mêmes une stratégie de masquage, ce qui introduit un biais méthodologique rarement discuté.

La comparaison des méthodes d'explicabilité entre elles (quelle méthode est la « meilleure » ?) constitue un objectif secondaire. L'enjeu principal est de comprendre comment le facteur de masquage influence les résultats au sein de chaque méthode, et d'en tirer des recommandations pratiques.

1.3 Objectifs et contributions

L'objectif principal de ce stage était de concevoir et d'implémenter un framework de benchmark permettant d'isoler et d'étudier l'impact de la stratégie de masquage sur les méthodes d'explication par occultation. Cet objectif se décline en plusieurs axes de travail.

Le premier axe a consisté en une étude bibliographique portant sur les méthodes d'explicabilité par perturbation (LIME, H2O, RISE, HiPe), les différentes approches de masquage décrites dans la littérature (du remplacement par une couleur constante à l'inpainting neuronal), et les métriques d'évaluation proposées par la communauté.

Le deuxième axe concerne le développement du framework. Nous avons conçu une architecture modulaire intégrant les quatre méthodes sous une interface commune, à laquelle nous avons associé sept stratégies de masquage standardisées (`cube_rgb`, `kmeans_colors`, `mean_masked`, `mean_unmasked`, `blur`, `gaussian_noise` et `inpainting`), applicables uniformément à toutes les méthodes. Le framework comprend également huit métriques d'évaluation, un support multi-classe pour les images contenant plusieurs objets, et un système de génération automatique de rapports

HTML.

Parmi ces stratégies, `kmeans_colors` constitue une contribution proposée dans le cadre de ce stage : plutôt que d'utiliser des couleurs arbitraires, cette stratégie extrait les couleurs dominantes de l'image via un clustering K-means (LLOYD, 1982) et les utilise pour le remplacement, produisant des perturbations contextuellement pertinentes.

Le troisième axe porte sur l'analyse comparative et expérimentale. Les expérimentations sur le dataset PASCAL VOC 2007 avec un modèle ResNet-18 ont permis d'évaluer 30 combinaisons méthode-stratégie et de mettre en évidence l'influence significative de la stratégie de masquage sur les résultats de chaque méthode. Le choix d'un échantillon restreint (40 images, 53 paires image-classe) reflète un arbitrage assumé face au coût combinatoire du protocole (30 combinaisons méthode-stratégie, soit 1590 explications générées) ; les implications de cette portée réduite sur la généralisation des résultats sont discutées au chapitre 5. L'analyse, orientée stratégie par stratégie, compare les performances au sein de chaque méthode et entre méthodes, et débouche sur des recommandations pratiques pour le choix d'une stratégie selon le contexte d'utilisation.

Au terme de ces trois axes, les contributions propres de ce travail sont les suivantes :

- C1.** Un cadre expérimental qui traite la stratégie d'occultation comme une variable et non comme un réglage implicite. Les quatre méthodes sont réimplémentées derrière une interface commune de sorte que leur stratégie de remplacement, jusque-là figée dans le code de chacune, devienne un paramètre libre. C'est ce qui rend possible la comparaison à méthode constante ; les benchmarks existants, qui standardisent les données et les métriques mais conservent la stratégie par défaut de chaque méthode, ne le permettent pas.
- C2.** La stratégie `kmeans_colors`, proposée dans le cadre de ce stage, qui construit le substrat de remplacement à partir des couleurs dominantes de l'image elle-même plutôt que d'une palette fixée a priori.
- C3.** Un protocole d'évaluation apparié : chaque combinaison méthode-stratégie est évaluée sur les mêmes paires image-classe, avec le même modèle et les mêmes métriques, de sorte que les écarts observés soient imputables au seul facteur étudié.
- C4.** Un ensemble de résultats et de recommandations : la mise en évidence du fait que la variabilité induite par la stratégie est, pour une méthode donnée, du même ordre de grandeur que l'écart entre méthodes ; la comparaison des

stratégies proches et éloignées de la distribution d’entraînement ; et la tension observée entre métriques de fidélité et métriques de corrélation.

1.4 Organisation du rapport

Ce rapport est organisé en cinq chapitres. Après cette introduction, le chapitre 2 présente l’état de l’art sur l’explicabilité en vision par ordinateur, les quatre méthodes étudiées, les stratégies d’occultation et les métriques d’évaluation. Le chapitre 3 décrit la méthodologie adoptée et les détails d’implémentation du framework, incluant les adaptations apportées à chaque méthode et l’infrastructure technique. Le chapitre 4 présente les résultats expérimentaux, avec une analyse orientée sur l’impact des stratégies de masquage pour chaque méthode, complétée par des observations transversales et des exemples visuels. Le chapitre 5 dresse le bilan du stage, discute les limites de l’étude et propose des perspectives.

Les annexes rassemblent les extraits de code source, les classes du dataset, les résultats détaillés et des captures du rapport HTML généré. Elles comprennent également un retour d’expérience sur la gestion du projet (organisation du travail, planning, difficultés rencontrées et compétences acquises), placé hors du corps du mémoire car il ne participe pas à la démonstration scientifique.

Chapitre 2

État de l'Art

Ce chapitre présente les fondements théoriques sur lesquels repose notre travail. Nous commençons par situer l'explicabilité dans le contexte de la vision par ordinateur, puis nous décrivons les quatre méthodes étudiées, les stratégies d'occultation existantes et les métriques d'évaluation.

2.1 Explicabilité en vision par ordinateur

2.1.1 Enjeux de l'explicabilité

L'explicabilité des systèmes d'intelligence artificielle est devenue un sujet de préoccupation majeur pour plusieurs raisons complémentaires (ARRIETA et al., 2020 ; DOSHI-VELEZ & KIM, 2017). Du point de vue de la confiance, les utilisateurs finaux (médecins, juges, opérateurs industriels) doivent pouvoir comprendre, au moins partiellement, le raisonnement sous-jacent d'un modèle pour en accepter les recommandations. Du point de vue du développeur, l'explicabilité est un outil de débogage : elle permet de détecter les corrélations parasites apprises par le modèle, comme un classifieur d'images médicales qui reconnaîtrait le logo de l'hôpital plutôt que la pathologie. Du point de vue réglementaire, le RGPD européen introduit un « droit à l'explication » pour les décisions automatisées ayant un impact significatif (EUROPEAN PARLIAMENT AND COUNCIL OF THE EUROPEAN UNION, 2016), et le règlement européen sur l'intelligence artificielle (PARLEMENT EUROPÉEN ET CONSEIL DE L'UNION EUROPÉENNE, 2024) renforce ces exigences de transparence et d'information pour les systèmes classés à haut risque.

2.1.2 Taxonomie des approches

Les méthodes d’explicabilité se distinguent selon plusieurs axes (LA ROSA et al., 2023). Une première distinction oppose les méthodes intrinsèques, où la transparence est intégrée au modèle lui-même (arbres de décision, modèles linéaires), aux méthodes post-hoc, qui génèrent une explication après coup, sans modifier le modèle. Une deuxième distinction porte sur la portée de l’explication : les méthodes locales expliquent une prédiction spécifique pour une instance donnée, tandis que les méthodes globales décrivent le comportement d’ensemble du modèle. Enfin, certaines méthodes sont spécifiques à un type de modèle (Grad-CAM exploite les gradients d’un CNN (SELVARAJU et al., 2020)), tandis que d’autres sont agnostiques et traitent le modèle comme une boîte noire.

Parmi les méthodes post-hoc, trois grandes familles se distinguent par leur mécanisme d’attribution. Les méthodes par gradient exploitent la dérivabilité du réseau : l’approche fondatrice de SIMONYAN et al. (2014) calcule directement le gradient du score de prédiction par rapport aux pixels d’entrée pour produire une carte de saillance, tandis que des raffinements comme les gradients intégrés (SUNDARARAJAN et al., 2017) accumulent ce gradient le long d’un chemin entre une image de référence neutre et l’image étudiée, afin de satisfaire des propriétés axiomatiques (sensibilité, invariance à l’implémentation) que le gradient brut ne vérifie pas. Ces méthodes sont rapides, un seul *forward-backward pass* ou quelques dizaines suffisant à produire une explication, mais elles nécessitent un accès direct aux poids et aux gradients du réseau. Les méthodes par propagation de pertinence, dont la référence est la *Layer-wise Relevance Propagation* (LRP) (BACH et al., 2015), redistribuent le score de sortie couche par couche jusqu’à l’entrée selon des règles de conservation locales ; elles produisent des cartes fines, mais leur implémentation dépend étroitement de l’architecture du réseau expliqué, ce qui limite leur portabilité d’un modèle à l’autre. Les méthodes par perturbation, enfin, masquent des régions de l’entrée et observent l’effet sur la prédiction : elles ne nécessitent aucun accès aux paramètres internes du modèle, au prix d’un nombre potentiellement élevé d’évaluations (*forward passes*). Ce sont ces dernières qui font l’objet de ce travail, précisément parce que leur caractère agnostique les rend applicables à n’importe quel classifieur, y compris ceux dont l’architecture interne n’est pas accessible.

En vision par ordinateur, les explications prennent le plus souvent la forme de

cartes de saillance : des images de même dimension que l’entrée où chaque pixel reçoit une valeur indiquant son importance relative pour la prédiction. Ce travail se concentre sur les méthodes post-hoc, locales et agnostiques.

2.2 Méthodes d’explication par perturbation

Les méthodes par perturbation génèrent des explications en masquant différentes parties de l’image d’entrée et en observant comment la prédiction du modèle varie. Le principe remonte aux travaux de ZEILER et FERGUS (2014), qui balaient l’image avec un carré gris et enregistrent la chute du score de prédiction pour localiser les régions déterminantes. COVERT et al. (2021) ont depuis montré que la plupart des méthodes d’explication post-hoc, y compris LIME et les valeurs de Shapley, se ramènent à un même cadre formel : expliquer en retirant de l’information, ce qui suppose systématiquement de définir comment une variable est retirée. Cette section décrit les quatre méthodes retenues pour notre étude : LIME, H2O, RISE et HiPe.

Ce choix n’est pas arbitraire. D’une part, ces quatre méthodes couvrent un éventail représentatif des approches par perturbation : segmentation à un seul niveau (LIME), absence de segmentation (RISE), segmentation hiérarchique exhaustive (H2O) et segmentation hiérarchique adaptative (HiPe), avec des profils de coût de calcul contrastés. D’autre part, et surtout, ces quatre méthodes utilisent chacune, dans la littérature, une stratégie de masquage par défaut différente (couleur moyenne pour LIME, masque interpolé pour RISE, cube RGB pour H2O, flou pour HiPe) : cette hétérogénéité déjà visible entre méthodes de référence est précisément ce qui motive l’étude systématique du choix de la stratégie, plutôt que d’accepter chaque défaut sans le questionner.

2.2.1 LIME

LIME (*Local Interpretable Model-agnostic Explanations*), proposé par RIBEIRO et al. (2016), repose sur l’idée d’approximer localement le comportement d’un modèle complexe par un modèle interprétable simple. Pour expliquer la prédiction $f(x)$ d’un modèle f sur une image x , LIME commence par segmenter l’image en superpixels $\{s_1, \dots, s_n\}$ via l’algorithme QuickShift (VEDALDI & SOATTO, 2008). La méthode génère ensuite N échantillons perturbés en masquant aléatoirement des sous-ensembles de superpixels, pondère chaque échantillon par sa proximité à l’instance originale, puis

ajuste un modèle linéaire local dont les coefficients indiquent l'importance de chaque superpixel.

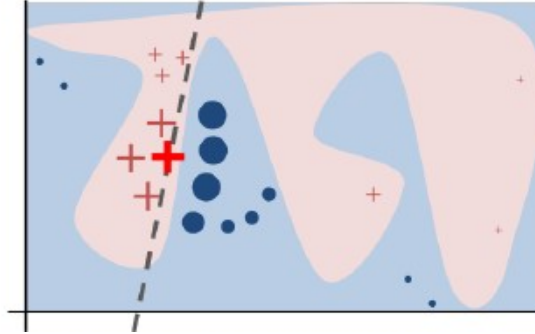


FIGURE 2.1 – Principe de l'approximation locale de LIME. Figure reproduite de RIBEIRO et al. (2016).

La frontière de décision du modèle complexe (fond bleu/rose) ne peut pas être approchée globalement par un modèle linéaire ; LIME échantillonne des instances autour de l'instance expliquée (croix rouge), les pondère par leur proximité (taille des points) et ajuste la droite pointillée, fidèle localement mais non globalement.

Formellement, l'explication est obtenue en minimisant :

$$\xi(x) = \arg \min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \quad (2.1)$$

où G désigne la famille des modèles interprétables (ici, des régressions linéaires), $\mathcal{L}$ mesure la fidélité de l'approximation locale, $\pi_x(z) = \exp(-D(x, z)^2/\sigma^2)$ est un noyau de proximité centré sur x , et $\Omega(g)$ pénalise la complexité du modèle explicatif.

En pratique, g est une régression ridge pondérée par le noyau de proximité : chaque échantillon perturbé contribue à l'ajustement proportionnellement à sa similarité avec l'instance originale x , ce qui privilégie les perturbations les plus proches dans l'approximation locale. Le nombre de superpixels retenus dans l'explication finale est généralement limité par une sélection de variables (par exemple K-Lasso), afin de produire une explication parcimonieuse et donc lisible par un humain.

LIME est agnostique au modèle et fournit des explications en termes de superpixels sémantiquement cohérents, ce qui facilite l'interprétation. En revanche, la nécessité d'entraîner un modèle linéaire local (surrogate model) pour chaque explication rend la méthode relativement lente, surtout lorsque le nombre d'échantillons est élevé. La

qualité de la segmentation en superpixels conditionne directement celle de l'explication, et la variabilité de l'échantillonnage aléatoire limite la reproductibilité. L'hypothèse de linéarité locale n'est par ailleurs pas nécessairement satisfaite pour des modèles fortement non linéaires.

2.2.2 H2O

H2O (*Heatmap by Hierarchical Occlusion*), introduit par POMMÉ et al. (2023), améliore les méthodes d'occultation classiques en combinant une segmentation hiérarchique avec un échantillonnage de couleurs dans un sous-ensemble restreint du cube RGB. L'image est segmentée à plusieurs échelles via l'algorithme SLIC (ACHANTA et al., 2012), puis chaque région est occultée et l'impact sur la prédiction est mesuré. Les contributions des différentes échelles sont ensuite agrégées pour former la carte de saillance finale.

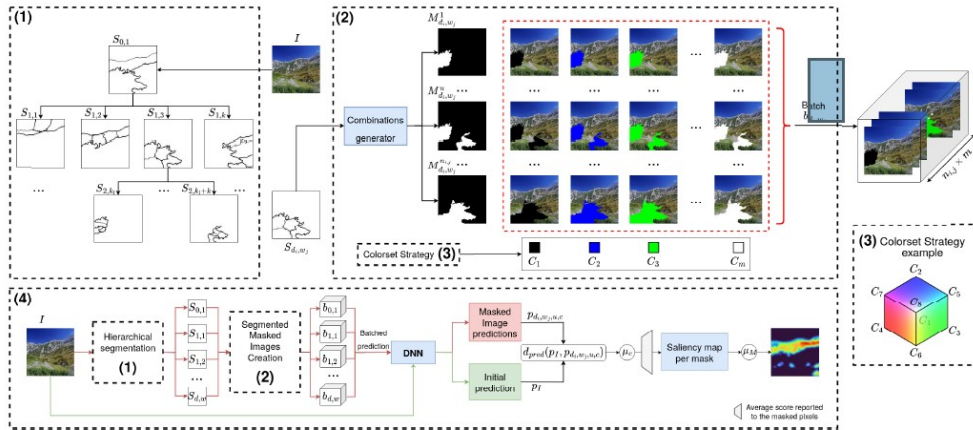


FIGURE 2.2 – Vue d'ensemble de H2O. Figure reproduite de POMMÉ et al. (2023).

(1) L'image est segmentée hiérarchiquement en superpixels. (2) À chaque niveau de la hiérarchie, les combinaisons de superpixels forment un ensemble de masques appliqués à l'image. (3) Les pixels masqués sont remplacés par une couleur à la fois, prélevée dans un cube RGB restreint (`cube_rgb`). (4) Toutes les images altérées sont prédites par lots, et la chute de score est reportée sur les pixels masqués.

La saillance d'un pixel (i, j) est calculée comme :

$$S(i, j) = \sum_k \mathbb{1}_{(i,j) \in s_k} \cdot \Delta p_k \quad (2.2)$$

où $\Delta p_k = f(x)_c - f(x_{-s_k})_c$ représente la chute de prédiction pour la classe cible c lors de l'occultation du superpixel s_k , et la somme porte sur tous les superpixels de tous les

niveaux hiérarchiques. En pratique, chaque contribution Δp_k est d’abord normalisée par la taille (en pixels) du superpixel s_k , de sorte que les régions fines des niveaux profonds de la hiérarchie et les régions larges des niveaux grossiers contribuent sur une échelle comparable ; la carte obtenue est ensuite divisée, pixel par pixel, par le nombre de fois où chaque pixel a été couvert par un superpixel à travers l’ensemble des niveaux, ce qui produit une véritable moyenne pondérée plutôt qu’une simple somme.

L’analyse hiérarchique exhaustive confère à H2O une bonne capacité à capturer l’importance des régions à différentes échelles spatiales, ce qui le rend robuste aux variations de taille des objets. En contrepartie, l’évaluation de chaque superpixel à chaque niveau de la hiérarchie rend la méthode plus coûteuse en temps de calcul que les approches non hiérarchiques, ce qui peut la rendre peu adaptée aux applications interactives.

2.2.3 RISE

RISE (*Randomized Input Sampling for Explanation*), proposé par PETSUK et al. (2018), adopte une approche radicalement différente en s’affranchissant de toute segmentation préalable. La méthode génère N masques binaires aléatoires sur une grille grossière de taille $g \times g$, où chaque cellule est activée indépendamment avec une probabilité p fixée a priori (loi de Bernoulli), puis agrandit ces masques à la résolution de l’image par interpolation bilinéaire avec un décalage aléatoire. Cette interpolation transforme les valeurs binaires en un masque continu dont les coefficients varient entre 0 et 1, produisant des transitions douces aux frontières des cellules. La carte de saillance est ensuite calculée comme une moyenne pondérée des masques par les scores de prédiction (figure 2.3) :

$$S(i, j) = \frac{1}{N \cdot p} \sum_{n=1}^N f(x \odot m_n)_c \cdot m_n(i, j) \quad (2.3)$$

où p est la probabilité qu’un pixel soit visible dans un masque donné et $\odot$ dénote la multiplication élément par élément.

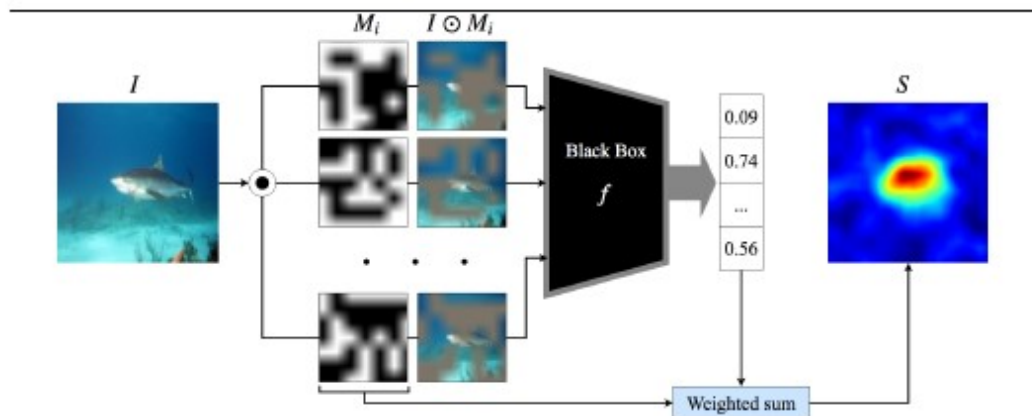


FIGURE 2.3 – Vue d’ensemble de RISE. Figure reproduite de PETSIUK et al. (2018).

L’image d’entrée I est multipliée élément par élément par des masques aléatoires M_i ; les images masquées sont soumises au modèle traité comme une boîte noire, et la carte de saillance S est la combinaison linéaire des masques pondérés par le score obtenu pour la classe cible.

La simplicité conceptuelle de RISE est son principal atout : pas de segmentation, pas de modèle auxiliaire, applicabilité directe à tout classifieur. L’interpolation bilinéaire des masques produit des cartes de saillance lisses (PETSIUK et al., 2018). En revanche, la résolution spatiale des explications est intrinsèquement limitée par la taille de la grille grossière (7×7 dans les expériences originales), ce qui tend à produire des cartes spatialement imprécises où la saillance se répartit sur de larges zones, incluant souvent le fond de l’image. Les auteurs eux-mêmes reconnaissent que RISE peut ne pas parvenir à éliminer ce bruit de fond (PETSIUK et al., 2018). Le nombre élevé de *forward passes* rend par ailleurs la méthode coûteuse en temps de calcul : la convergence nécessite un grand nombre de masques (4 000 à 8 000 selon le modèle), et l’approximation par échantillonnage de Monte Carlo peut produire des cartes bruitées, en particulier pour des objets de tailles variées (PETSIUK et al., 2018). Enfin, la stratégie par défaut atténue progressivement les intensités des pixels via la multiplication par le masque upsampled : les pixels au centre des cellules masquées sont réduits à zéro tandis que ceux aux frontières sont partiellement préservés grâce à l’interpolation. Bien que ce masquage soit plus doux qu’une occultation binaire, il constitue néanmoins une perturbation hors distribution.

2.2.4 HiPe

HiPe (*Hierarchical Perturbation*), développé par COOPER et al. (2022), propose une méthode de perturbation hiérarchique rapide et agnostique au modèle. La méthode commence par perturber de larges régions de l'image, puis raffine progressivement les zones identifiées comme saillantes en doublant la résolution à chaque itération. À chaque niveau, la carte de saillance est mise à jour selon :

$$S \leftarrow S + \text{ReLU}(f(x) - f(x')) \odot |\mathbf{M} - 1| \quad (2.4)$$

où x' est l'image perturbée (les pixels de la région masquée sont remplacés par la moyenne locale), $\mathbf{M}$ est le masque de perturbation upsampleé à la taille de l'image, et $\odot$ dénote le produit de Hadamard. La fonction ReLU garantit que seules les perturbations diminuant la confiance pour la classe cible contribuent à la carte. La figure 2.4 illustre ce raffinement progressif.

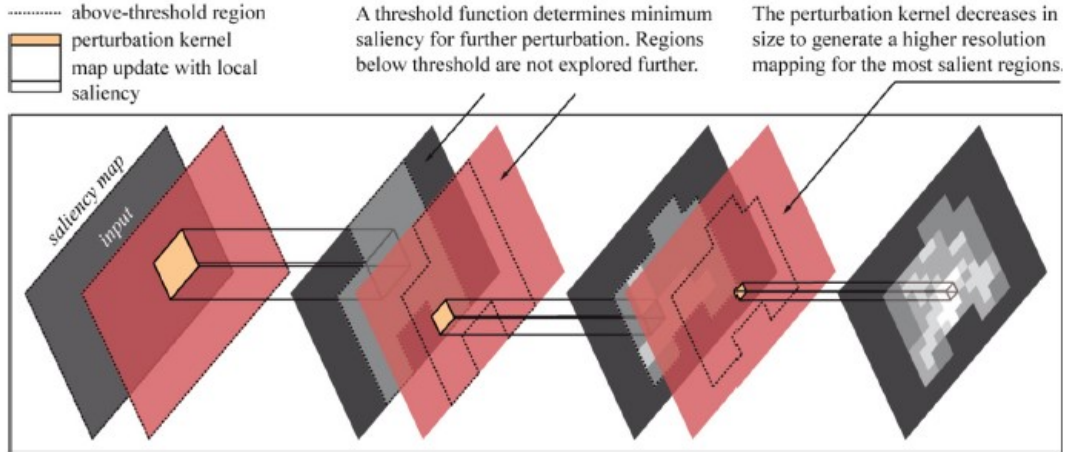


FIGURE 2.4 – Perturbation hiérarchique de HiPe. Figure reproduite de COOPER et al. (2022).

À chaque itération (de gauche à droite), une fonction de seuil détermine la saillance minimale requise pour poursuivre l'exploration : les régions sous le seuil ne sont plus raffinées, tandis que le noyau de perturbation diminue de taille pour cartographier les régions les plus saillantes à une résolution supérieure.

Une région n'est raffinée au niveau de résolution suivant que si sa saillance maximale dépasse le *mid-range* de la carte courante, c'est-à-dire le point médian entre le minimum et le maximum de S : $\max(S \odot m) \geq (\min(S) + \max(S))/2$ pour le masque m considéré. Ce seuillage permet d'ignorer les zones peu saillantes et de concentrer

les calculs sur les régions prometteuses.

HiPe se distingue par sa rapidité : les auteurs rapportent un facteur d'accélération supérieur à 20 par rapport aux autres méthodes agnostiques au modèle, avec un temps inférieur à une seconde par image contre plus de 23 secondes pour RISE (COOPER et al., 2022). Le raffinement progressif concentre les calculs sur les régions pertinentes et la résolution adaptative permet de capturer des détails aussi fins que 2×2 pixels. Le nombre de niveaux hiérarchiques est déterminé automatiquement à partir de la taille de l'image, ce qui réduit le besoin de réglage d'hyperparamètres. En revanche, HiPe ne peut pas capturer les interactions entre régions spatialement distinctes, c'est-à-dire les cas où l'effet combiné de deux régions dépasse la somme de leurs effets individuels (COOPER et al., 2022). Par ailleurs, le choix du substrat de perturbation influence les résultats, bien que les auteurs constatent empiriquement que la moyenne locale de la région perturbée donne des performances légèrement supérieures aux alternatives (zéro, flou gaussien).

2.2.5 Comparaison synthétique

Le tableau 2.1 résume les caractéristiques principales des quatre méthodes.

TABLE 2.1 – Comparaison des méthodes d'explication étudiées

Caractéristique	LIME	H2O	RISE	HiPe
Segmentation requise	Oui	Oui	Non	Non
Approche	Locale linéaire	Hiérarchique	Stochastique	Hiérarchique
Hyperparamètre	N échantillons	Nb. niveaux	$N, g \times g$	Nb. niveaux (auto)
Résolution	Superpixels	Multi-échelle	Grille interpolée	Adaptative
Coût de calcul	Élevé	Élevé	Élevé	Faible

2.3 Stratégies d'occultation

2.3.1 Pourquoi remplacer les pixels ?

Toutes les méthodes par perturbation reposent sur le même principe : masquer des régions de l'image et observer la variation de la prédiction. Contrairement aux données tabulaires, où l'on peut simplement retirer une variable, une image est une grille de pixels contiguës. Il n'est pas possible de « supprimer » des pixels sans les remplacer par d'autres valeurs, car cela modifierait la structure même de l'entrée attendue par le

réseau de neurones. La question fondamentale est donc : comment remplacer les pixels occultés ?

Le choix de la valeur de remplacement n'est pas anodin, et il est documenté comme tel dans la littérature. STURMFELS et al. (2020) montrent, pour les méthodes à chemin d'intégration, que le choix de l'entrée de référence modifie visiblement les cartes d'attribution obtenues, sans qu'aucune référence ne s'impose objectivement aux autres. Un remplacement trop artificiel (par exemple du noir ou du bruit aléatoire) risque de créer des artefacts hors de la distribution d'entraînement du modèle : HOOKER et al. (2019) soulignent que le classifieur est alors évalué sur des entrées qu'il n'a jamais rencontrées, de sorte que la chute de score mesurée peut refléter ce décalage de distribution autant que l'importance réelle de la région masquée. À l'inverse, un remplacement trop naturel (par exemple via un algorithme d'inpainting neuronal) peut préserver une partie de l'information sémantique que l'on souhaitait supprimer, réduisant ainsi le signal mesuré : CHANG et al. (2019) observent qu'un modèle génératif conditionné sur le reste de l'image peut reconstituer l'objet occulté et donc annuler l'effet de l'occultation. C'est cette tension entre réalisme et destruction de l'information qui motive l'exploration de multiples stratégies.

2.3.2 Stratégies de remplacement simples

La stratégie la plus élémentaire consiste à remplacer les pixels masqués par une couleur constante (noir, gris ou blanc), comme le carré gris employé dès les premières expériences d'occultation (ZEILER & FERGUS, 2014). Elle est rapide mais crée des discontinuités visuelles marquées, c'est-à-dire des contours artificiels absents des images naturelles. Le remplacement par la moyenne de l'image (`mean_masked`), ou par la moyenne des pixels non masqués (`mean_unmasked`), préserve mieux la distribution statistique globale, au risque d'introduire localement des couleurs incohérentes. L'application d'un flou gaussien aux régions masquées constitue un compromis intéressant : elle préserve la structure de bas niveau tout en supprimant les détails de haute fréquence discriminants pour le classifieur. FONG et VEDALDI (2017) retiennent explicitement le flou parmi leurs opérateurs de perturbation pour cette raison, en cherchant des perturbations qui suppriment l'information utile au classifieur tout en restant plausibles pour lui.

2.3.3 Stratégies de remplacement élaborées

Le remplacement par du bruit gaussien détruit l’information tout en maintenant une apparence variée, mais les perturbations résultantes restent clairement hors distribution. La stratégie cube RGB (`cube_rgb`), utilisée par défaut dans H2O, échantillonne des couleurs uniformément dans l’espace RGB en subdivisant le cube en z^3 couleurs ; elle produit des perturbations variées mais potentiellement très éloignées du contenu original. La stratégie `kmeans_colors`, que nous proposons dans ce travail, extrait les couleurs dominantes de l’image via un clustering K-means et les utilise pour le remplacement, produisant des perturbations contextuellement pertinentes.

Enfin, l’inpainting neuronal, implémenté ici via le modèle LaMa (SUVOROV et al., 2022), utilise un réseau de neurones pour reconstruire de manière réaliste les régions masquées. Cette approche produit les perturbations les plus naturelles, mais elle est coûteuse en temps de calcul et peut parfois reconstruire l’objet d’intérêt, annulant l’effet de l’occultation (CHANG et al., 2019). On considère parfois l’inpainting comme une forme de baseline idéale, dans la mesure où il supprime l’information sémantique tout en préservant la cohérence visuelle ; c’est l’argument avancé par AFSHAR et al. (2024) pour le préférer aux substrats constants en histopathologie.

2.3.4 Synthèse

Le tableau 2.2 récapitule les caractéristiques théoriques des stratégies présentées. La colonne « Vitesse » qualifie le coût intrinsèque de l’opération de remplacement des pixels, indépendamment de la méthode d’explication utilisée et du nombre de perturbations générées.

TABLE 2.2 – Caractéristiques des stratégies d’occultation

Stratégie	Réalisme	Vitesse	Préservation info
Couleur constante	Faible	Très rapide	Aucune
Moyenne (masked/unmasked)	Moyen	Très rapide	Faible
Flou gaussien	Élevé	Rapide	Partielle (basse fréq.)
Bruit gaussien	Faible	Très rapide	Aucune
Cube RGB	Faible	Très rapide	Aucune
K-means colors	Moyen	Rapide	Contextuelle
Inpainting neuronal	Très élevé	Lent	Contextuelle

2.4 Métriques d'évaluation

L'évaluation quantitative des méthodes d'explicabilité est un domaine de recherche actif (HOOKER et al., 2019 ; SAMEK et al., 2017). Les métriques se regroupent en plusieurs familles mesurant des aspects complémentaires de la qualité d'une explication.

2.4.1 Métriques de fidélité

La fidélité évalue dans quelle mesure l'explication reflète le comportement réel du modèle. Deux métriques complémentaires, introduites conjointement par PETSUK et al. (2018), sont couramment utilisées.

La Deletion AUC (LiB – *Lower is Better*) supprime progressivement les pixels par ordre décroissant d'importance selon la carte de saillance, et mesure l'aire sous la courbe de prédiction résultante :

$$\text{Del-AUC} = \int_0^1 f(x_{\text{del}}(t))_c dt \quad (2.5)$$

où $x_{\text{del}}(t)$ désigne l'image dont les $t\%$ de pixels les plus importants ont été remplacés selon une stratégie de base (flou dans notre implémentation), $f(\cdot)_c$ est le score de prédiction pour la classe cible c , et t parcourt l'intervalle $[0, 1]$. Une valeur basse indique que les pixels identifiés comme importants le sont réellement pour le modèle.

L'Insertion AUC (HiB – *Higher is Better*) fonctionne à l'inverse : partant d'une image de base (tous les pixels masqués), elle révèle progressivement les pixels par ordre d'importance :

$$\text{Ins-AUC} = \int_0^1 f(x_{\text{ins}}(t))_c dt \quad (2.6)$$

Une valeur élevée signifie que les pixels importants suffisent à reconstruire la prédiction. La figure 2.5 illustre les deux procédures sur un cas réel de notre benchmark.

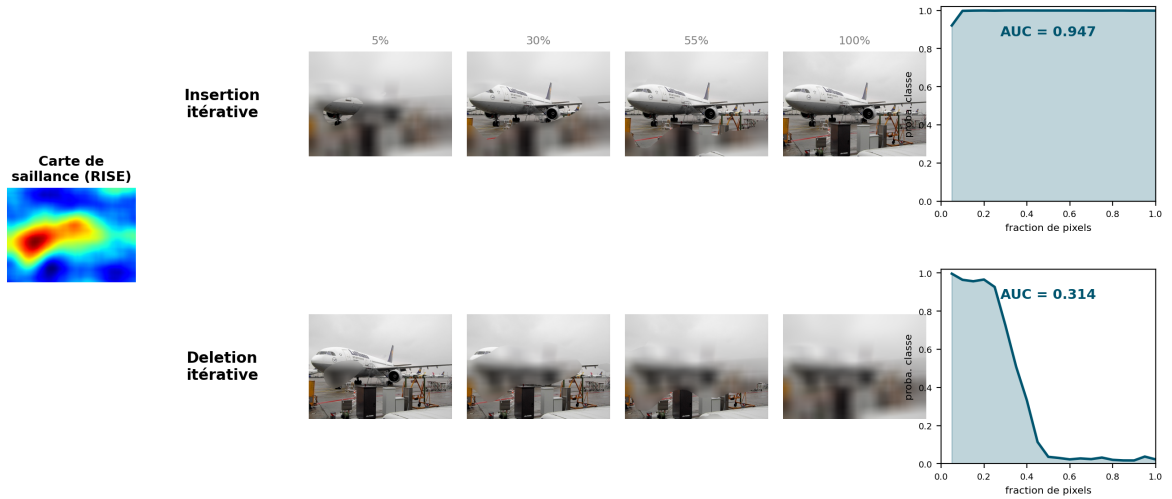


FIGURE 2.5 – Illustration du Deletion AUC et de l'Insertion AUC.

À partir de la carte de saillance (à gauche), les pixels sont révélés (en haut) ou effacés (en bas) par ordre de saillance décroissant ; la courbe de probabilité prédite pour la classe cible est intégrée pour donner l'AUC. Une insertion qui monte vite (AUC élevée) et une deletion qui chute vite (AUC faible) indiquent toutes deux une carte fidèle. Carte, courbes et valeurs proviennent du run expérimental de ce travail (RISE, image 000033, classe aeroplane).

2.4.2 Métriques de corrélation

Les métriques de corrélation évaluent si l'ordre d'importance attribué par la carte de saillance correspond à l'impact réel des pixels sur la prédiction. La corrélation de Spearman (HiB) mesure la corrélation de rang entre le classement d'importance issu de la carte de saillance et le classement issu de la variation réelle de la prédiction lors de la suppression. La corrélation de Pearson (HiB) mesure la corrélation linéaire entre les valeurs de saillance et les changements de prédiction. Ces deux métriques sont calculées à la fois pour la procédure de deletion et d'insertion.

2.4.3 Métriques de localisation et de complexité

Le Pointing Game (HiB) (J. ZHANG et al., 2018) est une métrique binaire qui vérifie si le pixel de saillance maximale se situe dans la région de vérité terrain (bounding box ou masque de segmentation) :

$$\text{PG} = \mathbb{I} \left[\arg \max_{i,j} S(i,j) \in \text{GT}_c \right] \quad (2.7)$$

La Sparsité (HiB) mesure la concentration de l’explication. Elle relève de la famille des critères de complexité proposés par BHATT et al. (2020), qui postulent qu’une explication concentrée sur un petit nombre de variables est plus exploitable par un humain qu’une explication diffuse. Une carte sparse met en évidence peu de régions, ce qui peut être plus interprétable :

$$\text{Sparsity} = 1 - \frac{\text{mean}(S)}{\text{max}(S)} \quad (2.8)$$

2.4.4 Synthèse des métriques

Le tableau 2.3 résume les huit métriques utilisées dans notre benchmark.

TABLE 2.3 – Récapitulatif des métriques d’évaluation

Métrique	Direction	Notation	Mesure
Deletion AUC	↓	LiB	Fidélité
Insertion AUC	↑	HiB	Fidélité
Spearman (deletion)	↑	HiB	Corrélation de rang
Spearman (insertion)	↓	LiB	Corrélation de rang
Pearson (deletion)	↑	HiB	Corrélation linéaire
Pearson (insertion)	↓	LiB	Corrélation linéaire
Pointing Game	↑	HiB	Localisation
Sparsité	↑	HiB	Concentration

2.5 Positionnement

L’analyse de la littérature révèle plusieurs lacunes. Chaque méthode est typiquement évaluée avec ses propres paramètres par défaut et sa propre stratégie de masquage, ce qui rend les comparaisons inter-méthodes biaisées. Les stratégies de masquage par défaut diffèrent d’une méthode à l’autre (couleur moyenne pour LIME, noir pour RISE, cube RGB pour H2O), empêchant de dissocier l’impact de la méthode de celui de la stratégie. Des benchmarks récents comme Saliency-Bench (Y. ZHANG et al., 2023) standardisent les datasets et les métriques pour comparer les méthodes de manière plus rigoureuse, mais ne font pas varier la stratégie de masquage à méthode constante : le remplacement des pixels occultés y reste un paramètre implicite propre à chaque

méthode, jamais isolé comme facteur d'étude à part entière. La plupart des évaluations se limitent par ailleurs aux images mono-objet.

Ce choix de remplacement n'est pourtant pas neutre. Des travaux très récents le confirment : en histopathologie, AFSHAR et al. (2024) montrent que la stratégie d'occultation utilisée influence directement la fiabilité de l'évaluation, et proposent un remplacement par inpainting pour limiter les artefacts hors distribution. Comme la plupart des travaux sur le sujet, cette étude propose et évalue une stratégie de remplacement dans un contexte donné; elle ne compare pas un même ensemble de stratégies à travers plusieurs méthodes d'explication différentes, ce qui reste la question spécifique adressée dans ce travail. Dans une direction voisine, BLÜCHER et al. (2024) évaluent une quarantaine de stratégies d'occultation et montrent que les classements produits par les benchmarks de *pixel flipping* varient selon la stratégie retenue. Leur objectif est toutefois inverse du nôtre : ils cherchent à neutraliser ce facteur, en construisant une mesure agrégée dont le classement ne dépende plus de la stratégie employée, là où ce travail prend ce même facteur pour objet d'étude et cherche à caractériser son effet méthode par méthode. Le constat se retrouve dans d'autres domaines applicatifs : en classification de scènes en télédétection, KLOTZ et al. (2025) relèvent que les performances des méthodes par perturbation, dont l'occultation et LIME, dépendent fortement du substrat de perturbation retenu, et que les métriques de fidélité héritent de cette même dépendance. Leur analyse porte toutefois sur le choix d'une méthode d'explication à stratégie fixée, et non sur la comparaison systématique d'un ensemble de stratégies au sein de chaque méthode.

Par ailleurs, les métriques d'évaluation elles-mêmes posent un problème rarement discuté : les métriques d'insertion et de deletion, parmi les plus utilisées pour évaluer les cartes de saillance, s'appuient sur une stratégie de masquage bien précise (typiquement le flou pour l'insertion, le remplacement par zéro pour la deletion) pour évaluer des méthodes qui utilisent elles-mêmes des stratégies de masquage différentes. On évalue donc l'impact d'une stratégie de masquage au moyen d'une métrique qui repose elle-même sur une stratégie de masquage, ce qui introduit une circularité méthodologique. Ce problème est identifié dans la littérature sur l'évaluation : RONG et al. (2022) montrent que le masquage employé par les métriques de suppression laisse fuiter de l'information (la forme même du masque renseigne le classifieur sur les pixels retirés) et proposent une procédure de débiaisage pour y remédier. Dans le même sens,

NIERADZIK et al. (2024) montrent que les métriques d’insertion et de deletion sont sensibles au décalage de distribution induit par la perturbation elle-même, au point d’en affecter la fiabilité du classement des méthodes. Ce constat rejoint des travaux très récents montrant que les résultats d’évaluation en XAI sont sensibles à des choix d’hyperparamètres a priori mineurs, au point de pouvoir être orientés artificiellement en faveur d’une méthode donnée (WICKSTRØM et al., 2024), ce qui souligne la nécessité de protocoles d’évaluation transparents et standardisés.

Notre travail adresse ces limites en proposant un framework où les quatre méthodes sont accessibles via une interface commune, évaluées avec les mêmes sept stratégies de masquage standardisées, sur les mêmes images et avec les mêmes huit métriques. L’objectif n’est pas tant de désigner une « meilleure méthode » que d’étudier, pour chaque méthode, l’impact du choix de la stratégie d’occultation sur les résultats obtenus.

Chapitre 3

Méthodologie et Implémentation

Ce chapitre décrit la conception et la réalisation du framework de benchmark. Nous présentons d’abord l’architecture générale et les principes de conception, puis les adaptations apportées à chaque méthode pour supporter les stratégies multiples, la standardisation des stratégies d’occultation, le système d’évaluation et enfin l’infrastructure technique.

3.1 Architecture du framework

3.1.1 Principes de conception

Le framework a été conçu pour qu’un chercheur puisse ajouter une nouvelle méthode ou une nouvelle stratégie sans modifier le code existant. Concrètement, les méthodes d’explication, les stratégies d’occultation, les métriques et la visualisation sont séparées en modules indépendants, reliés par des interfaces communes. Les expériences sont décrites dans des fichiers de configuration YAML versionnables, et les seeds aléatoires (Python, NumPy, PyTorch, CUDA) sont fixés pour garantir la reproductibilité. L’accélération GPU et le traitement par lots assurent des temps de calcul raisonnables sur l’ensemble du benchmark.

3.1.2 Vue d’ensemble

La figure [3.1](#) illustre l’architecture modulaire du framework.

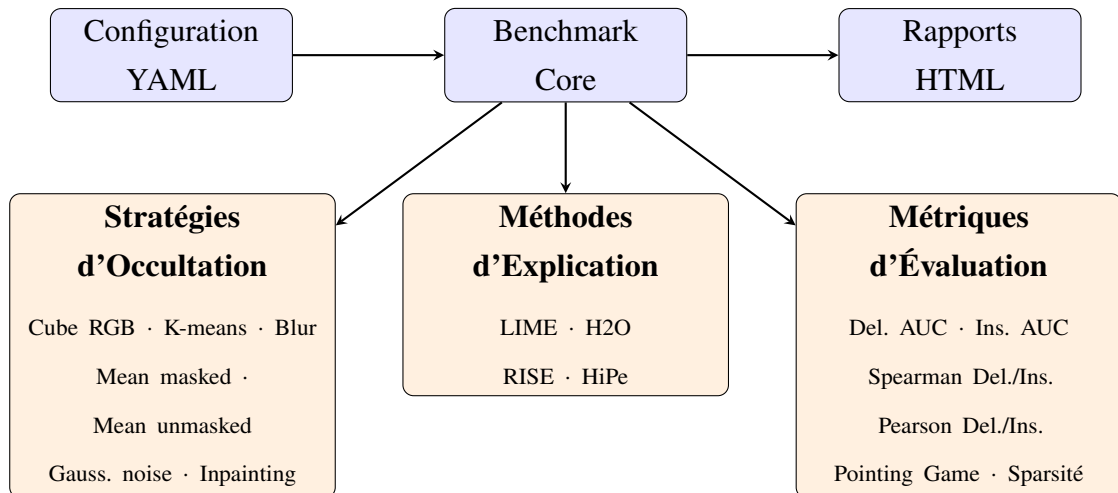


FIGURE 3.1 – Architecture générale du framework de benchmark.

Le Benchmark Core orchestre les trois modules principaux : stratégies d'occultation (7 stratégies), méthodes d'explication (4 méthodes) et métriques d'évaluation (8 métriques). La configuration est paramétrable via YAML et les résultats sont exportés sous forme de rapports HTML.

La figure 3.2 détaille le flux de données à travers les cinq étapes du pipeline expérimental.

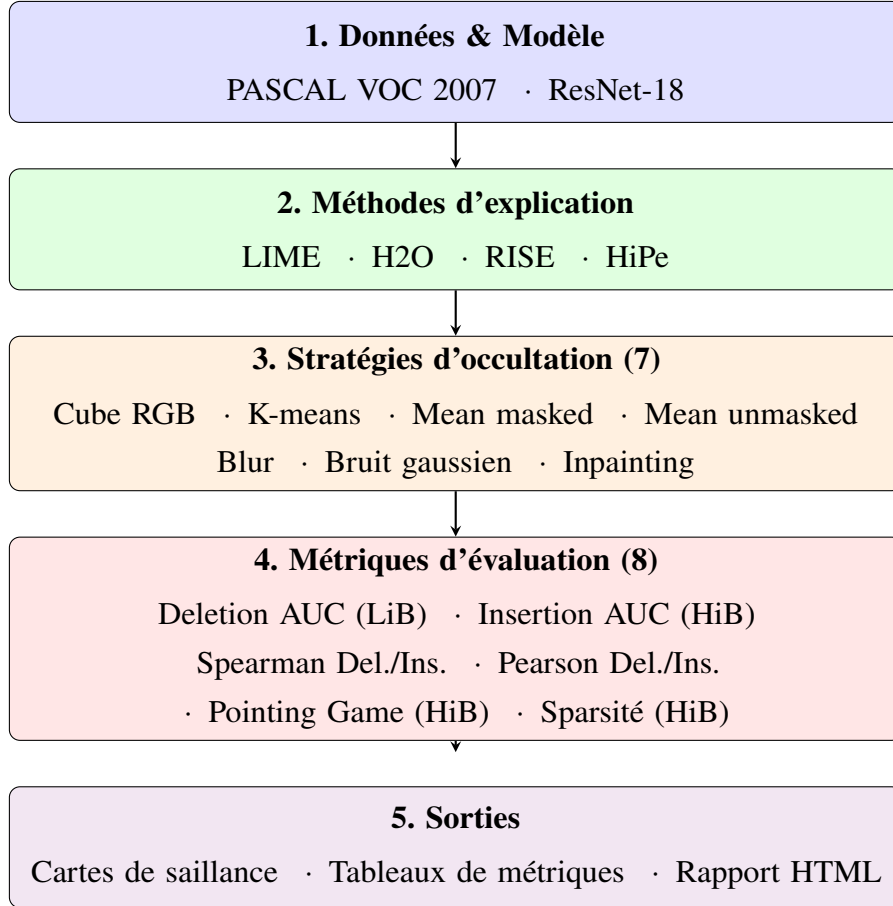


FIGURE 3.2 – Pipeline expérimental du framework de benchmark.

3.2 Intégration des méthodes

Chaque méthode suit une interface commune imposant une méthode `explain(image, target_class)` qui retourne une carte de saillance. Cette section détaille les adaptations spécifiques apportées à chaque méthode pour le support multi-stratégie.

3.2.1 Adaptation de LIME

L'intégration des stratégies multiples dans LIME a nécessité une refonte de la méthode `data_labels()`, cœur de la génération des perturbations. Dans l'implémentation originale, pour chaque vecteur binaire $z \in \{0, 1\}^n$ échantillonné, les superpixels désactivés ($z_i = 0$) sont remplacés par une image pré-calculée où chaque superpixel est rempli par sa couleur moyenne. Chaque échantillon produit exactement une image perturbée.

Notre adaptation introduit un paramètre `color_strategy` qui conditionne le

comportement de la boucle de perturbation selon trois modes. Pour les stratégies multi-couleurs (`cube_rgb`, `kmeans_colors`), chaque vecteur z engendre $|C|$ images perturbées, une par couleur de la palette. L'ensemble d'entraînement du surrogate model passe ainsi de N à $N \times |C|$ échantillons, enrichissant l'approximation locale au prix d'un temps de calcul proportionnellement plus élevé. Pour la stratégie `blur`, chaque vecteur z engendre K versions floutées à différents niveaux de σ , produisant $N \times K$ échantillons. Pour les stratégies à image unique (`mean_masked`, `mean_unmasked`, `gaussian_noise`, `inpainting`), le ratio 1:1 original est préservé.

3.2.2 Adaptation de H2O

L'implémentation de référence de H2O repose sur un pipeline monolithique conçu pour une seule stratégie (`cube RGB`). Notre adaptation restructure trois composants. Premièrement, la méthode `generate_mask_combinations()` retourne désormais, en plus des masques combinatoires, une liste `labels_list` associant à chaque masque sa carte de labels SLIC. Cette information est indispensable pour les stratégies opérant au niveau des superpixels individuels (`mean_masked`, `blur_superpixel`). Deuxièmement, la méthode `alter_and_predict_images()` reçoit un paramètre `strategy` qui dispatche vers la logique de remplacement appropriée. Troisièmement, la fonction `alteration_dist()` a été étendue avec un mode *class-specific* calculant la détérioration sur la seule classe cible, plutôt que sur l'ensemble du vecteur de prédiction.

Ces modifications préservent le mécanisme d'agrégation hiérarchique original (moyennage pondéré des scores de détérioration) tout en le rendant compatible avec des perturbations de nature variée.

3.2.3 Adaptation de RISE

Pour RISE, la généralisation est plus directe. L'opération originale, à savoir la multiplication élément par élément $x' = x \odot m$, est remplacée par un appel à la stratégie de remplacement sélectionnée. Pour chaque masque m_n , les pixels où $m_n(i, j) = 0$ sont remplacés selon la stratégie choisie plutôt que mis à zéro. Le reste de l'algorithme (génération des masques, agrégation pondérée) reste inchangé.

3.2.4 Adaptation de HiPe

HiPe accepte nativement un paramètre de perturbation. Notre intégration se limite à router ce paramètre vers les sept stratégies standardisées et à assurer la compatibilité des formats d’entrée/sortie avec l’interface commune du framework.

3.2.5 Support multi-classe

Les images du monde réel contiennent fréquemment plusieurs objets de classes différentes. Le framework détecte les classes présentes au-dessus d’un seuil de confiance ($\theta = 0.5$) et génère une carte de saillance spécifique pour chaque classe détectée. Pour LIME, les poids de l’explication sont extraits par classe depuis le modèle linéaire local. Pour H2O, le paramètre `target_class` oriente le calcul de la détérioration. Pour RISE et HiPe, les scores de prédiction utilisés dans l’agrégation sont extraits pour la classe cible.

3.3 Standardisation des stratégies d’occultation

Le chapitre précédent a présenté les stratégies d’occultation d’un point de vue conceptuel (voir tableau 2.2). Cette section décrit les choix d’implémentation technique pour chacune des sept stratégies intégrées au framework.

La stratégie `cube_rgb` subdivise l’espace RGB en z^3 couleurs (par défaut $z = 3$, soit 27 couleurs) et échantillonne uniformément parmi celles-ci. La stratégie `kmeans_colors`, proposée dans le cadre de ce stage, applique un clustering K-means sur les pixels de l’image pour extraire k couleurs dominantes, puis les utilise pour le remplacement. L’annexe A.1 fournit le pseudo-code de cette stratégie.

Les stratégies `mean_masked` et `mean_unmasked` calculent respectivement la moyenne des pixels dans la zone masquée et dans la zone non masquée, puis l’appliquent uniformément à la zone de remplacement. La stratégie `blur` applique un flou gaussien à plusieurs niveaux de $\sigma \in \{5, 10, 20, 40\}$. La stratégie `gaussian_noise` ajoute un bruit gaussien de $\sigma = 100$ aux pixels de la zone masquée. Enfin, la stratégie `inpainting` fait appel au modèle LaMa (SUVOROV et al., 2022) pour reconstruire les régions masquées.

3.4 Système d'évaluation

Le framework implémente les huit métriques présentées au chapitre 2. La Deletion AUC (équation 2.5) est calculée en supprimant progressivement les pixels par pas de 5 %, en utilisant le flou comme stratégie de remplacement. L'Insertion AUC (équation 2.6) révèle progressivement les pixels selon le même protocole inversé. Les corrélations de Spearman et Pearson sont calculées en deletion et en insertion. Le Pointing Game (équation 2.7) utilise les masques de segmentation de PASCAL VOC comme vérité terrain. La Sparsité (équation 2.8) est calculée sur la carte de saillance normalisée.

Pour chaque combinaison méthode-stratégie, les huit métriques sont calculées sur chacune des 53 paires image-classe, et nous rapportons les moyennes et écarts-types.

3.5 Infrastructure technique

3.5.1 Technologies

Le framework repose principalement sur PyTorch pour l'implémentation des méthodes LIME, RISE et HiPe. H2O, agnostique au modèle, est appliqué au même ResNet-18 (PyTorch) que les autres méthodes via une fonction de prédiction commune ; son implémentation hérite toutefois d'une dépendance à TensorFlow dont la coexistence avec PyTorch impose de libérer explicitement la mémoire GPU (`clear_session()` et ramasse-miettes) après chaque batch de prédictions. Le traitement d'images s'appuie sur OpenCV et scikit-image (segmentation SLIC, QuickShift), l'inpainting sur la bibliothèque `simple-lama-inpainting`, et les calculs statistiques sur NumPy, SciPy et scikit-learn.

3.5.2 Configuration et reproductibilité

Les expériences sont configurées via des fichiers YAML spécifiant le dataset, le modèle, les méthodes avec leurs stratégies, les paramètres de chaque méthode et les métriques à calculer. La reproductibilité est garantie par le contrôle des seeds aléatoires (Python, NumPy, PyTorch, CUDA).

3.5.3 Système de checkpoints

L'exécution complète du benchmark nécessitant plusieurs jours de calcul, un système de checkpoints sauvegarde l'état après chaque image traitée. En cas d'interruption, l'exécution reprend là où elle s'est arrêtée. Un mode `fill_missing` permet de ne réexécuter que les combinaisons méthode-stratégie ayant échoué lors d'une exécution précédente. Ce mode charge le checkpoint existant, identifie pour chaque image les méthodes marquées comme échouées ou absentes, les réexécute, puis fusionne les nouveaux résultats avec les résultats existants.

3.5.4 Génération des rapports

Le framework génère automatiquement des rapports HTML interactifs incluant un tableau récapitulatif avec classement coloré (premier en bleu, deuxième en vert, avant-dernier en orange, dernier en rouge), des grilles de cartes de saillance comparatives par image, et les temps d'exécution détaillés. Ces rapports constituent un outil de visualisation pratique pour l'analyse comparative des résultats.

Chapitre 4

Résultats Expérimentaux

Ce chapitre présente les résultats des expérimentations menées avec le framework. Après une description du protocole expérimental, nous analysons l’impact des stratégies d’occultation sur chaque méthode, puis nous formulons des observations transversales avant de terminer par les résultats qualitatifs.

4.1 Protocole expérimental

4.1.1 Dataset et modèle

Les expériences ont été menées sur le dataset PASCAL VOC 2007 (EVERINGHAM et al., 2010), un benchmark standard en classification et détection d’objets comptant 20 classes (aeroplan, bicycle, bird, . . . , tvmonitor). Le dataset est multi-label : plusieurs objets de classes différentes peuvent coexister dans une même image. Pour des raisons de temps de calcul, un sous-ensemble de 40 images a été sélectionné, produisant 53 paires (image, classe) distinctes.

Le modèle de classification est un ResNet-18 (HE et al., 2016) pré-entraîné sur ImageNet puis fine-tuné sur PASCAL VOC 2007. Ses performances sur le split de validation (2 510 images) sont les suivantes : mAP de 76.0 %, précision macro de 81.6 % et F1-score macro de 66.9 %.

4.1.2 Combinaisons évaluées

Le benchmark a évalué 30 combinaisons méthode-stratégie, récapitulées dans le tableau 4.1. LIME et RISE supportent chacune 8 stratégies, incluant leur stratégie default respective (remplacement par la couleur moyenne du superpixel pour LIME,

atténuation progressive des pixels via le masque upsampled pour RISE). H2O et HiPe en supportent chacune 7.

TABLE 4.1 – Combinaisons méthode-stratégie évaluées

Méthode	Nb.	Stratégies
LIME	8	default, cube_rgb, kmeans_colors, blur, mean_masked, mean_unmasked, gaussian_noise, inpainting
H2O	7	cube_rgb, kmeans_colors, mean_masked, mean_unmasked, blur_superpixel, gaussian_noise, inpainting
RISE	8	default, cube_rgb, kmeans_colors, blur, mean_masked, mean_unmasked, gaussian_noise, inpainting
HiPe	7	cube_rgb, kmeans_colors, blur, mean_masked, mean_unmasked, gaussian_noise, inpainting

Au total, 30 combinaisons ont produit des résultats exploitables. Pour chacune, les 8 métriques ont été calculées sur les 53 paires image-classe, et nous rapportons les moyennes.

Le nombre d'échantillons perturbés pour LIME a été fixé à $N = 1\,000$ pour le benchmark final, conformément aux pratiques recommandées dans la littérature, afin de garantir une approximation locale fiable et des résultats reproductibles. Des valeurs réduites ont pu être employées lors de tests rapides en cours de développement, mais l'ensemble des résultats présentés dans ce chapitre provient de cette configuration complète.

4.2 Analyse par méthode

L'objectif principal de ce benchmark est de comprendre comment le choix de la stratégie d'occultation influence les performances au sein de chaque méthode. Cette section présente cette analyse pour les quatre méthodes.

4.2.1 LIME

Le tableau 4.2 détaille les performances de LIME pour ses huit stratégies.

TABLE 4.2 – Impact des stratégies d’occultation sur LIME

Stratégie	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
default	0.339	0.773	0.849	0.719	0.753	3.8	9.4 s
cube_rgb	0.355	0.770	0.774	0.749	0.759	5.0	521 s
kmeans_colors	0.308	0.789	0.906	0.723	0.782	5.2	505 s
blur	0.323	0.785	0.887	0.715	0.777	3.7	5 801 s
mean_masked	0.327	0.788	0.849	0.719	0.759	4.3	10.1 s
mean_unmasked	0.337	0.783	0.868	0.737	0.755	4.6	10.6 s
gaussian_noise	0.363	0.760	0.887	0.728	0.744	3.6	19.5 s
inpainting	0.334	0.776	0.906	0.712	0.728	4.2	9.3 s

Del. = Deletion AUC, Ins. = Insertion AUC, PG = Pointing Game, DS = Deletion Spearman, DP = Deletion Pearson, Sp. = Sparsity. ↓ : plus bas est mieux, ↑ : plus haut est mieux.

Pour LIME, le Deletion AUC varie de 0.308 (`kmeans_colors`) à 0.363 (`gaussian_noise`), soit un écart de 18 %. Les stratégies `kmeans_colors` et `blur` obtiennent les meilleures fidélités, mais à des coûts très différents : `blur` nécessite près de 97 minutes par image en raison des multiples niveaux de flou appliqués à chaque superpixel, tandis que `kmeans_colors` atteint une fidélité encore meilleure (Del. 0.308) en 505 secondes.

Un résultat notable concerne l’`inpainting` et le `kmeans_colors`, qui obtiennent les meilleurs Pointing Game de LIME (0.906 tous les deux) mais avec des fidélités différentes : `kmeans_colors` est aussi le meilleur en fidélité (Del. 0.308), tandis que `inpainting` reste intermédiaire (Del. 0.334). Les stratégies `blur` et `gaussian_noise` partagent le même Pointing Game (0.887). La stratégie `default` obtient une bonne fidélité (Del. 0.339) pour un temps minimal (9.4 s).

4.2.2 H2O

Le tableau 4.3 présente les résultats de H2O avec ses sept stratégies.

TABLE 4.3 – Impact des stratégies d’occultation sur H2O

Stratégie	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
blur_superpixel	0.418	0.720	0.604	0.877	0.761	8.6	239 s
cube_rgb	0.372	0.714	0.717	0.836	0.799	5.3	497 s
kmeans_colors	0.432	0.697	0.623	0.881	0.820	10.1	506 s
mean_masked	0.390	0.720	0.679	0.861	0.777	7.4	32.1 s
mean_unmasked	0.434	0.672	0.679	0.882	0.753	7.1	38.9 s
gaussian_noise	0.524	0.630	0.566	0.822	0.708	11.0	46.2 s
inpainting	0.485	0.628	0.509	0.717	0.615	11.3	49.1 s

H2O est la méthode pour laquelle les stratégies ont l’impact le plus prononcé : le Deletion AUC varie de 0.372 (`cube_rgb`) à 0.524 (`gaussian_noise`), soit un écart de 0.151, le plus élevé de toutes les méthodes. La stratégie `cube_rgb` obtient le meilleur Deletion AUC (0.372) et le meilleur Pointing Game (0.717), tandis que `kmeans_colors` et `mean_unmasked` dominent sur la corrélation ($DS > 0.88$). La stratégie `blur_superpixel`, qui applique un flou gaussien par superpixel, se situe en position intermédiaire (Del. 0.418, DS 0.877) avec un bon compromis fidélité-corrélation.

Les stratégies `gaussian_noise` et `inpainting` se distinguent par des performances nettement plus faibles : Del. AUC de 0.524 et 0.485 respectivement, et un Pointing Game inférieur à 0.57. Les stratégies multi-couleurs (`cube_rgb`, `kmeans_colors`) et `blur_superpixel` sont coûteuses (239–506 s) car elles multiplient les images perturbées. Pour les stratégies simples, H2O reste dans un temps acceptable (32–49 s).

4.2.3 RISE

Le tableau 4.4 présente les résultats de RISE pour ses huit stratégies.

TABLE 4.4 – Impact des stratégies d’occultation sur RISE

Stratégie	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
default	0.316	0.768	0.811	0.846	0.834	2.5	11.8 s
cube_rgb	0.268	0.792	0.925	0.799	0.825	2.7	13.3 s
kmeans_colors	0.262	0.796	0.943	0.821	0.827	2.8	18.7 s
blur	0.281	0.797	0.868	0.808	0.814	2.8	15.5 s
mean_masked	0.268	0.803	0.925	0.802	0.829	2.8	14.8 s
mean_unmasked	0.263	0.799	0.943	0.803	0.830	2.8	14.7 s
gaussian_noise	0.308	0.774	0.943	0.834	0.831	2.7	23.0 s
inpainting	0.365	0.735	0.811	0.875	0.838	2.4	42.8 s

RISE est la méthode pour laquelle la stratégie a le plus d’impact en termes absolus : le Deletion AUC varie de 0.262 (kmeans_colors) à 0.365 (inpainting), soit un écart de 39 %. RISE domine en fidélité quelle que soit la stratégie, mais le choix de celle-ci influence considérablement les résultats.

Les stratégies kmeans_colors et mean_unmasked obtiennent des résultats quasi identiques (Del. 0.262 vs 0.263, PG 0.943 pour les deux). La stratégie default est paradoxalement l’une des moins performantes en fidélité (Del. 0.316), tout en obtenant la meilleure corrélation (DS 0.846). Cela suggère que les stratégies de remplacement par des couleurs contextuelles améliorent la fidélité en évitant les artefacts du masquage noir. Le temps d’exécution de RISE est peu sensible à la stratégie pour les approches simples (12–23 s), seul l’inpainting provoquant un surcoût significatif (42.8 s).

4.2.4 HiPe

Le tableau 4.5 détaille les performances de HiPe.

TABLE 4.5 – Impact des stratégies d’occultation sur HiPe

Stratégie	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
cube_rgb	0.491	0.611	0.377	0.833	0.612	15.9	2.0 s
kmeans_colors	0.378	0.692	0.717	0.883	0.816	5.4	7.4 s
blur	0.431	0.611	0.623	0.907	0.696	10.4	7.8 s
mean_masked	0.467	0.596	0.585	0.818	0.557	18.9	1.7 s
mean_unmasked	0.415	0.649	0.642	0.889	0.723	9.2	1.9 s
gaussian_noise	0.396	0.677	0.566	0.858	0.739	7.4	26.7 s
inpainting	0.436	0.625	0.717	0.864	0.683	7.7	34.7 s

HiPe est la méthode la plus sensible à la stratégie : le Pointing Game varie de 0.377 (cube_rgb) à 0.717 (kmeans_colors et inpainting), un écart considérable qui montre que la stratégie peut radicalement altérer la localisation de la saillance.

La stratégie blur produit la meilleure corrélation (DS 0.907), qui est aussi le meilleur score toutes méthodes confondues. En revanche, kmeans_colors offre un profil plus équilibré (Del. 0.378, PG 0.717, DS 0.883). HiPe est systématiquement la méthode la plus rapide pour les stratégies simples (1.7–7.8 s), mais sa vulnérabilité au cube_rgb (PG 0.377, Del. 0.491) et au gaussian_noise (PG 0.566, Del. 0.396) révèle que les perturbations hors distribution dégradent le signal à chaque niveau de résolution de la perturbation hiérarchique multi-échelle.

4.3 Observations transversales

4.3.1 Synthèse inter-méthodes

L’analyse par méthode révèle plusieurs constats transversaux. Les quatre méthodes réagissent différemment aux mêmes stratégies : kmeans_colors améliore les résultats dans la majorité des cas (trois méthodes sur quatre), tandis que gaussian_noise et inpainting produisent des effets contrastés selon la méthode.

Un constat frappant est que la variabilité intra-méthode (liée au choix de la stratégie) peut dépasser la variabilité inter-méthodes : le Deletion AUC de H2O varie de 0.151 selon la stratégie, celui de HiPe de 0.113 et celui de RISE de 0.103, alors que l’écart entre la meilleure et la moins bonne méthode (moyennée sur toutes les stratégies) est de 0.145. Cette comparaison appelle toutefois deux précautions d’interprétation : d’une

part, l'étendue intra-méthode est calculée sur sept à huit stratégies alors que l'étendue inter-méthodes ne porte que sur quatre méthodes, ce qui favorise mécaniquement la première ; d'autre part, seule H2O (0.151) dépasse effectivement l'écart inter-méthodes, RISE (0.103), HiPe (0.113) et LIME (0.055) restant en deçà. Le constat demeure néanmoins : pour une méthode donnée, le choix de la stratégie d'occultation peut avoir un impact du même ordre de grandeur, voire supérieur, que celui du choix de la méthode elle-même.

Par ailleurs, les méthodes ne s'accordent pas entre elles sur les régions importantes d'une même image, même lorsqu'elles obtiennent des scores similaires : RISE localise la saillance de manière diffuse, HiPe la concentre sur des régions précises, LIME la segmente en superpixels. Ce désaccord spatial suggère que les différentes architectures de perturbation capturent des facettes complémentaires de l'importance des régions, et souligne l'intérêt d'approfondir les métriques de comparaison.

4.3.2 Tension fidélité/corrélation

Les résultats mettent en évidence un phénomène important : les méthodes les meilleures en fidélité (RISE, LIME) ne sont pas les meilleures en corrélation (H2O, HiPe), et inversement. Le tableau 4.6 résume cette tension.

TABLE 4.6 – Performances moyennes par méthode (moyennées sur toutes les stratégies)

Méthode	Del. AUC ↓	Ins. AUC ↑	DS ↑	PG ↑
LIME	0.336	0.778	0.725	0.866
H2O	0.436	0.683	0.839	0.625
RISE	0.292	0.783	0.824	0.896
HiPe	0.431	0.638	0.865	0.604

colonnes.

Cette divergence tient à ce que les deux familles de métriques ne mesurent pas la même chose. Les métriques de fidélité (Deletion/Insertion AUC) évaluent l'impact absolu de la suppression ou de l'insertion de pixels sur la prédiction, tandis que les métriques de corrélation (Spearman, Pearson) vérifient si l'ordre d'importance attribué par la carte de saillance correspond à l'impact réel observé. Ainsi, RISE domine en fidélité (Del. 0.292) et localisation (PG 0.896), tandis que HiPe obtient la meilleure corrélation (DS 0.865). Les approches hiérarchiques (H2O, HiPe) excellent à ordonner

correctement les régions, même si leurs valeurs absolues d'importance diffèrent de celles estimées par le classifieur.

4.3.3 Analyse temporelle

Le tableau 4.7 synthétise les temps d'exécution par méthode et par stratégie.

TABLE 4.7 – Temps d'exécution moyen par image (secondes)

Stratégie	LIME	H2O	RISE	HiPe
default	9.4	–	11.8	–
cube_rgb	521	497	13.3	2.0
kmeans_colors	505	506	18.7	7.4
mean_masked	10.1	32.1	14.8	1.7
mean_unmasked	10.6	38.9	14.7	1.9
blur	5 801	–	15.5	7.8
blur_superpixel	–	239	–	–
gaussian_noise	19.5	46.2	23.0	26.7
inpainting	9.3	49.1	42.8	34.7

méthode.

Les stratégies multi-couleurs (cube_rgb, kmeans_colors) provoquent une explosion du temps pour LIME (505 s) et H2O (506 s) car elles multiplient le nombre d'images perturbées. En revanche, RISE et HiPe restent rapides car la stratégie ne modifie que le type de remplissage, sans multiplier les masques. LIME-blur est la combinaison la plus lente (5 801 s $\approx$ 97 min) en raison de la convolution gaussienne multi-niveaux appliquée à chaque superpixel. HiPe est systématiquement la méthode la plus rapide pour les stratégies simples ($\approx$ 2 s).

4.4 Résultats qualitatifs

4.4.1 Comparaison visuelle des méthodes

La figure 4.1 compare les cartes de saillance produites par les quatre méthodes pour une même image avec la stratégie blur.

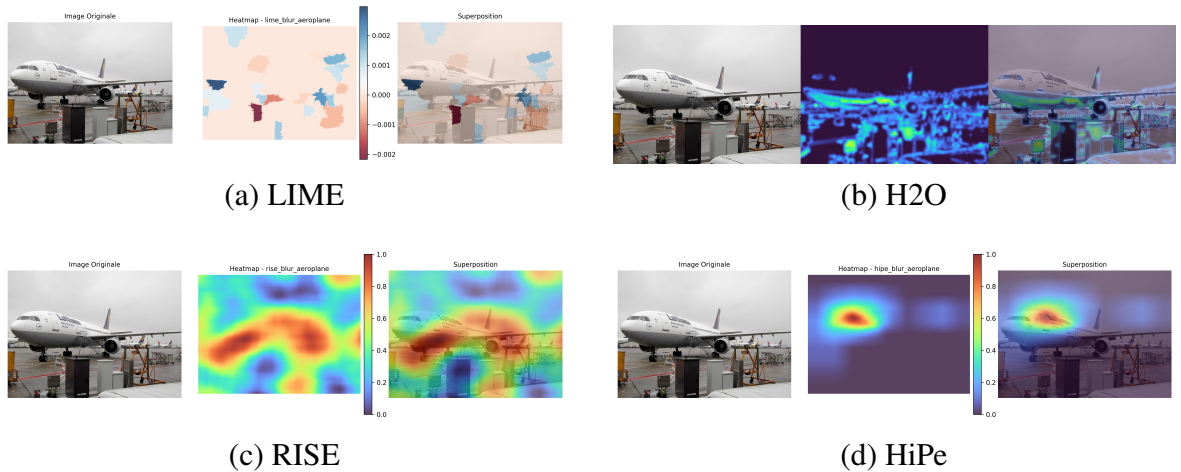


FIGURE 4.1 – Comparaison des cartes de saillance par méthode (stratégie blur, image 000033, classe avion).

LIME produit une carte granuleuse suivant les contours des superpixels. H2O génère une heatmap fine mais peu contrastée. RISE offre une carte lisse mais diffuse, la saillance se répartissant sur de larges zones. HiPe concentre la saillance sur le fuselage de l’avion.

Cette comparaison illustre visuellement le désaccord spatial entre les méthodes évoqué en section précédente. LIME découpe l’image en superpixels et attribue un poids uniforme à chacun, produisant des contours nets mais une résolution limitée par la taille des superpixels. H2O, par son approche hiérarchique multi-échelle, génère une carte plus fine mais dont le contraste est faible, rendant l’interprétation moins immédiate. RISE produit une carte lisse grâce à l’interpolation bilinéaire de ses masques, mais la saillance se répartit sur une large zone incluant le fond, ce qui est cohérent avec la limite de résolution spatiale de la grille 7×7 . HiPe, grâce au raffinement progressif, concentre la saillance sur la zone la plus discriminante, au prix d’une couverture spatiale plus restreinte.

4.4.2 Impact visuel des stratégies

La figure 4.2 illustre comment différentes stratégies modifient la carte de saillance produite par une même méthode (HiPe) sur une même image.

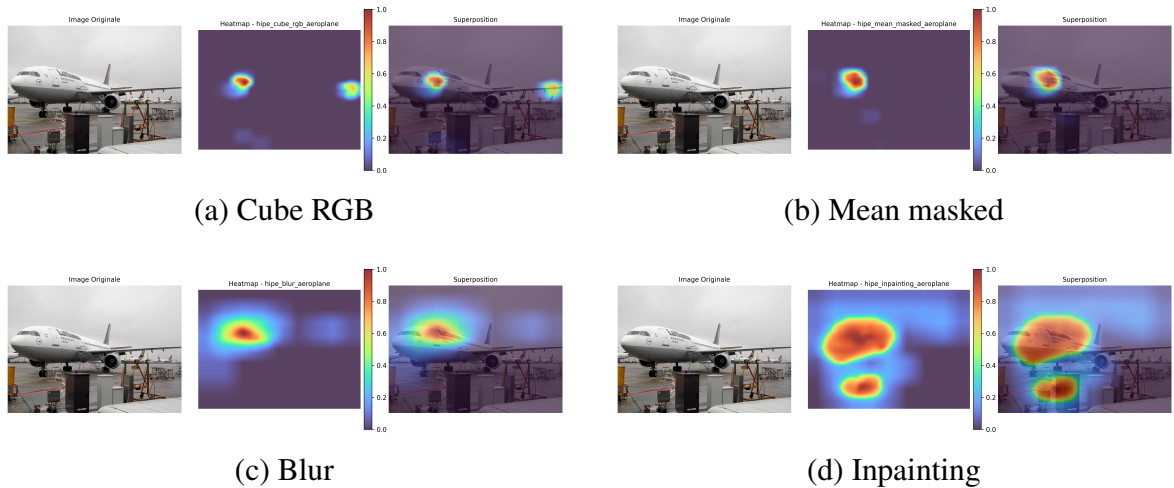


FIGURE 4.2 – Impact des stratégies d’occultation sur les cartes de saillance (méthode HiPe, image 000033, classe avion).

Cube RGB produit deux spots localisés sur le fuselage et la queue. Mean masked concentre un seul spot sur le fuselage (carte la plus parcimonieuse). Blur offre une couverture étendue (bon compromis). Inpainting génère la couverture la plus large, avec deux zones distinctes.

Cet exemple met en évidence l’impact concret du choix de la stratégie sur la carte de saillance produite par une même méthode. Cube RGB, en remplaçant les pixels par des couleurs arbitraires éloignées du contenu original, provoque une réaction forte du modèle qui se traduit par deux spots très localisés. Mean masked, en remplaçant par la moyenne locale, préserve mieux la cohérence visuelle et produit la carte la plus parcimonieuse. Blur, en détruisant les hautes fréquences tout en conservant la structure globale, offre un compromis entre sensibilité et couverture. Inpainting, en reconstruisant les régions de manière réaliste, préserve une partie de l’information sémantique et force le modèle à réagir à des indices plus subtils, ce qui étend la zone saillante. Ces différences visuelles confirment que la stratégie d’occultation n’est pas un simple détail d’implémentation mais un choix méthodologique qui influence directement l’explication produite.

4.4.3 Cas multi-classe

La figure 4.3 montre un exemple d’explications spécifiques par classe sur une image multi-objets.



(a) Classe bicycle



(b) Classe person

FIGURE 4.3 – Explications spécifiques par classe (H2O, stratégie blur, image 000129).

Sur cette image de cyclistes, H2O parvient à discriminer les classes : pour bicycle, la saillance se concentre sur les roues et le cadre, tandis que pour person, elle se déplace vers les corps et les visages.

Cet exemple montre que H2O parvient à discriminer les classes au sein d’une même image, attribuant des zones de saillance distinctes au vélo et aux personnes. Ce comportement est rendu possible par le paramètre `target_class` qui oriente le calcul de la détérioration vers une classe spécifique, et valide le fonctionnement du mode multi-classe du framework.

4.5 Synthèse et recommandations

L’objectif principal de ce benchmark étant d’évaluer l’impact de la stratégie d’occultation sur les performances, cette section synthétise les enseignements par stratégie.

Le tableau 4.8 résume l’effet de chaque stratégie, moyenné sur les quatre méthodes.

TABLE 4.8 – Impact des stratégies d’occultation (moyennes sur les 4 méthodes)

Stratégie	Del. AUC ↓	PG ↑	DS ↑
kmeans_colors	0.345	0.797	0.827
mean_unmasked	0.362	0.783	0.828
mean_masked	0.363	0.759	0.800
blur	0.345	0.792	0.810
cube_rgb	0.372	0.698	0.804
gaussian_noise	0.398	0.741	0.811
inpainting	0.405	0.736	0.792

Valeurs en gras : meilleur score par

colonne. Les stratégies default et blur_superpixel sont exclues car elles ne sont pas communes aux quatre méthodes.

La stratégie kmeans_colors se révèle être le meilleur compromis global : elle obtient la meilleure fidélité moyenne (Del. 0.345, à égalité avec blur) et le meilleur Pointing Game (0.797), tout en améliorant les résultats par rapport à la stratégie par défaut pour la majorité des méthodes (l’exception notable étant H2O, dont la stratégie par défaut cube_rgb conserve la meilleure fidélité). Ce résultat s’explique par le fait que le remplacement par des couleurs contextuellement cohérentes (extraites de l’image elle-même) produit des perturbations qui restent dans la distribution d’entraînement du modèle, évitant les artefacts artificiels. Une précaution d’interprétation s’impose néanmoins : le Deletion AUC repose lui-même sur une perturbation par flou (équation 2.5) et peut, par construction, avantager les stratégies produisant des perturbations proches de la distribution naturelle des images, telles que kmeans_colors et blur. Ce classement doit donc s’entendre au sens des métriques retenues ; des protocoles d’évaluation fondés sur le ré-entraînement (HOOKER et al., 2019) permettraient de le confirmer indépendamment.

La stratégie mean_unmasked offre des performances proches (Del. 0.362, PG 0.783) pour un coût en temps nettement inférieur, ce qui en fait une excellente alternative pour le prototypage. À l’inverse, les stratégies cube_rgb et gaussian_noise produisent des perturbations hors distribution qui dégradent la fidélité et la localisation. L’inpainting présente un profil contrasté : la pire fidélité (Del. 0.405) malgré un Pointing Game honorable (0.736), confirmant la tension entre réalisme de la perturbation et destruction effective de l’information.

Ces résultats confirment l'hypothèse centrale du stage : le choix de la stratégie d'occultation influence significativement les résultats, et les stratégies par défaut de chaque méthode ne sont pas nécessairement les plus performantes.

Ces recommandations prennent tout leur sens dans des contextes de déploiement où l'explication conditionne une décision à enjeu. Considérons, à titre d'illustration, une équipe de validation sécurité d'un système de perception pour véhicule autonome ou d'aide à la conduite, chargée d'enquêter sur un incident de détection (par exemple, un piéton ou un cycliste mal reconnu). Pour établir si le modèle s'est appuyé sur les indices visuels pertinents (silhouette, position) plutôt que sur une corrélation parasite (arrière-plan, éclairage), les ingénieurs génèrent une carte de saillance post-hoc. Nos résultats montrent qu'un choix naïf de stratégie de masquage (en particulier une stratégie hors distribution comme `gaussian_noise`, souvent le défaut implicite d'une librairie XAI) peut produire une explication de moins bonne qualité de localisation, risquant de biaiser le diagnostic de l'incident. Préférer une stratégie in-distribution comme `kmeans_colors` pour ce type d'analyse, comme le suggèrent nos résultats (tableau 4.8), constitue une recommandation directement actionnable. Ce type d'arbitrage stratégie-par-contexte est précisément ce que le framework développé dans ce stage permet de tester de façon reproductible, sur les données propres à un déploiement donné, plutôt que de s'en remettre aux réglages par défaut d'un outil XAI générique.

Chapitre 5

Conclusion

5.1 Bilan

Ce stage de six mois au sein de l'équipe ADVANSE du LIRMM a permis de concevoir, d'implémenter et de valider un framework de benchmark unifié pour l'évaluation des méthodes d'explicabilité en vision par ordinateur. La question centrale, à savoir l'impact de la stratégie d'occultation sur les performances de chaque méthode, a reçu une réponse claire : le choix de la stratégie modifie fortement les cartes de saillance produites, avec des variations du Deletion AUC pouvant atteindre 41 % pour une même méthode. Pour H2O, cette variabilité intra-méthode dépasse même l'écart mesuré entre les quatre méthodes, et elle en reste proche pour RISE et HiPe : le choix de la stratégie d'occultation a donc, pour une méthode donnée, un impact du même ordre de grandeur que le choix de la méthode elle-même.

Le framework intègre quatre méthodes de référence (LIME, H2O, RISE, HiPe) avec sept stratégies de masquage standardisées et huit métriques d'évaluation. L'ensemble des objectifs initiaux a été atteint, de l'étude bibliographique à la validation expérimentale sur PASCAL VOC 2007, en passant par la conception modulaire et l'implémentation du système d'évaluation.

Parmi les résultats les plus marquants, RISE domine les métriques de fidélité quelle que soit la stratégie, tandis que les approches hiérarchiques (H2O, HiPe) excellent en corrélation. La stratégie `kmeans_colors`, proposée dans le cadre de ce stage, obtient le meilleur Pointing Game (localisation, 0.797) de toutes les stratégies testées et figure parmi les meilleures en corrélation (Spearman 0.827). La tension observée entre fidélité et corrélation invite à repenser la manière dont les méthodes d'explicabilité

sont évaluées : une seule famille de métriques ne suffit pas à capturer la qualité d’une explication.

Rapportées aux contributions annoncées en introduction, ces résultats se lisent ainsi : le cadre expérimental (C1) a effectivement permis de faire varier la stratégie à méthode constante, ce qu’aucun des benchmarks consultés ne permettait ; la stratégie `kmeans_colors` (C2) se révèle compétitive et vient en tête sur la localisation ; le protocole apparié (C3) a fourni les 53 paires image-classe sur lesquelles reposent l’ensemble des comparaisons ; et les résultats (C4) débouchent sur des recommandations différenciées selon que l’on privilégie la fidélité ou la localisation.

Ce travail ne prétend pas révolutionner le vaste domaine de l’explicabilité en intelligence artificielle ; il en constitue une contribution modeste et ciblée, portant sur une question méthodologique jusqu’ici peu questionnée : celle du choix de la stratégie d’occultation dans les méthodes d’explication par perturbation.

5.2 Limites

Ce travail présente plusieurs limites qu’il convient de souligner. Les résultats sont spécifiques à un seul modèle (ResNet-18) et un seul dataset (PASCAL VOC 2007, 40 images). La généralisation à d’autres architectures (Vision Transformers, EfficientNet) et d’autres domaines (imagerie médicale, imagerie satellite) reste à vérifier. Le nombre d’images utilisé, bien que suffisant pour dégager des tendances, limite la significativité statistique des résultats : aucun test de significativité formel n’a été conduit, et les écarts les plus faibles (en particulier entre stratégies pour LIME, où la variabilité inter-images domine) doivent être considérés comme indicatifs, une validation par tests appariés sur un échantillon élargi restant nécessaire pour les confirmer.

Les métriques d’évaluation elles-mêmes ne sont pas exemptes de biais : les métriques de deletion et d’insertion reposent sur une stratégie de masquage (le flou dans notre cas) pour évaluer l’impact de la suppression des pixels, ce qui introduit une circularité méthodologique. De plus, les méthodes ne s’accordent pas entre elles sur les régions importantes d’une même image, même lorsqu’elles obtiennent des scores métriques similaires : RISE diffuse la saillance sur de larges zones, HiPe la concentre sur des régions précises, LIME la segmente en superpixels. Ce désaccord spatial montre que les métriques actuelles ne suffisent pas à capturer pleinement la qualité d’une explication, et souligne l’intérêt d’approfondir la recherche sur les méthodes de

comparaison entre explications. Une évaluation humaine, demandant à des utilisateurs de juger la qualité et l'utilité des explications, compléterait utilement les métriques automatiques, mais n'a pas pu être conduite dans le cadre de ce stage.

5.3 Perspectives

Ce travail ouvre plusieurs perspectives. À court terme, une augmentation du nombre d'images renforcerait la significativité statistique des résultats. À plus long terme, l'intégration de méthodes supplémentaires (Grad-CAM (SELVARAJU et al., 2020), SHAP (LUNDBERG & LEE, 2017)) et d'architectures plus récentes (Vision Transformers) permettrait de vérifier si les conclusions sur l'impact du masquage se généralisent au-delà des méthodes par perturbation et des CNN. Une étude utilisateur évaluant l'utilité perçue des explications selon la stratégie de masquage apporterait un éclairage complémentaire aux métriques automatiques. Enfin, la mise à disposition du framework en open source fournirait à la communauté un outil de benchmark standardisé, facilitant les comparaisons objectives entre méthodes.

5.4 Apports personnels

Sur le plan personnel, ce stage a été une expérience formatrice à plusieurs titres. Il m'a permis de mener un projet de recherche appliquée de bout en bout : de la lecture bibliographique à l'analyse des résultats, en passant par la conception logicielle et l'expérimentation. Les réunions hebdomadaires avec mon encadrant m'ont appris à structurer ma pensée, à présenter des résultats intermédiaires de manière claire, à ajuster mes priorités en fonction des retours, et m'ont sensibilisé à l'importance de la documentation et de la robustesse du code.

La résolution des conflits mémoire entre TensorFlow et PyTorch et la mise au point du système de checkpoints ont été des moments formateurs : ils m'ont montré que le travail d'ingénierie (fiabilité, robustesse, gestion des erreurs) est aussi important que le travail scientifique.

Cette expérience a renforcé mon intérêt pour le domaine de l'intelligence artificielle explicable et responsable, et m'a donné les compétences pour poursuivre dans cette voie, que ce soit en recherche ou dans une équipe R&D travaillant sur l'IA de confiance.

Références Bibliographiques

- ACHANTA, R., SHAJI, A., SMITH, K., LUCCHI, A., FUA, P., & SÜSTRUNK, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274-2282. <https://doi.org/10.1109/TPAMI.2012.120>
- AFSHAR, P., HASHEMBEIKI, S., KHANI, P., FATEMIZADEH, E., & ROHBAN, M. H. (2024). IBO : Inpainting-Based Occlusion to Enhance Explainable Artificial Intelligence Evaluation in Histopathology [Préprint arXiv:2408.16395].
- ARRIETA, A. B., DÍAZ-RODRÍGUEZ, N., DEL SER, J., BENNETOT, A., TABIK, S., BARBADO, A., GARCÍA, S., GIL-LÓPEZ, S., MOLINA, D., BENJAMINS, R., et al. (2020). Explainable Artificial Intelligence (XAI) : Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information Fusion*, 58, 82-115. <https://doi.org/10.1016/j.inffus.2019.12.012>
- BACH, S., BINDER, A., MONTAVON, G., KLAUSCHEN, F., MÜLLER, K.-R., & SAMEK, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7), e0130140. <https://doi.org/10.1371/journal.pone.0130140>
- BHATT, U., WELLER, A., & MOURA, J. M. F. (2020). Evaluating and Aggregating Feature-based Model Explanations. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 3016-3022. <https://doi.org/10.24963/ijcai.2020/417>
- BLÜCHER, S., VIELHABEN, J., & STRODTHOFF, N. (2024). Decoupling Pixel Flipping and Occlusion Strategy for Consistent XAI Benchmarks. *Transactions on Machine Learning Research*.

- CHANG, C.-H., CREAGER, E., GOLDBERG, A., & DUVENAUD, D. (2019). Explaining Image Classifiers by Counterfactual Generation. *International Conference on Learning Representations (ICLR)*.
- COOPER, J., ARANDJELOVIĆ, O., & HARRISON, D. J. (2022). Believe the HiPe : Hierarchical Perturbation for Fast, Robust, and Model-agnostic Saliency Mapping. *Pattern Recognition*, 129, 108743. <https://doi.org/10.1016/j.patcog.2022.108743>
- COVERT, I., LUNDBERG, S., & LEE, S.-I. (2021). Explaining by Removing : A Unified Framework for Model Explanation. *Journal of Machine Learning Research*, 22(209), 1-90.
- DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., & FEI-FEI, L. (2009). ImageNet : A Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 248-255.
- DOSHI-VELEZ, F., & KIM, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *arXiv preprint arXiv :1702.08608*.
- EUROPEAN PARLIAMENT AND COUNCIL OF THE EUROPEAN UNION. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation). <https://eur-lex.europa.eu/eli/reg/2016/679/oj>
- EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K. I., WINN, J., & ZISSERMAN, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338. <https://doi.org/10.1007/s11263-009-0275-4>
- FONG, R. C., & VEDALDI, A. (2017). Interpretable Explanations of Black Boxes by Meaningful Perturbation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 3429-3437. <https://doi.org/10.1109/ICCV.2017.371>
- HE, K., ZHANG, X., REN, S., & SUN, J. (2015). Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision (ICCV)*, 1026-1034.
- HE, K., ZHANG, X., REN, S., & SUN, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.
- HOOKE, S., ERHAN, D., KINDERMANS, P.-J., & KIM, B. (2019). A Benchmark for Interpretability Methods in Deep Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

- KLOTZ, J., BURGERT, T., & DEMIR, B. (2025). On the Effectiveness of Methods and Metrics for Explainable AI in Remote Sensing Image Scene Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.
- LA ROSA, B., BLASILLI, G., BOURQUI, R., AUBER, D., SANTUCCI, G., CAPOBIANCO, R., BERTINI, E., GIOT, R., & ANGELINI, M. (2023). State of the Art of Visual Analytics for eXplainable Deep Learning. *Computer Graphics Forum*, 42(1), 319-355. <https://doi.org/10.1111/cgf.14733>
- LLOYD, S. P. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129-137.
- LUNDBERG, S. M., & LEE, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
- NIERADZIK, L., STEPHANI, H., & KEUPER, J. (2024). Reliable Evaluation of Attribution Maps in CNNs : A Perturbation-Based Approach [Préprint arXiv:2411.14946].
- PARLEMENT EUROPÉEN ET CONSEIL DE L'UNION EUROPÉENNE. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) [ELI : <http://data.europa.eu/eli/reg/2024/1689/oj>].
- PETSIUK, V., DAS, A., & SAENKO, K. (2018). RISE : Randomized Input Sampling for Explanation of Black-box Models. *Proceedings of the British Machine Vision Conference (BMVC)*.
- POMMÉ, L.-E., BOURQUI, R., & GIOT, R. (2023). H2O : Heatmap by Hierarchical Occlusion [hal-04212098]. *Proceedings of Content-based Multimedia Indexing (CBMI)*.
- RIBEIRO, M. T., SINGH, S., & GUESTRIN, C. (2016). “Why Should I Trust You ?” : Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135-1144. <https://doi.org/10.1145/2939672.2939778>
- RONG, Y., LEEMANN, T., BORISOV, V., KASNECI, G., & KASNECI, E. (2022). A Consistent and Efficient Evaluation Strategy for Attribution Methods. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 162, 18770-18795.
- SAMEK, W., BINDER, A., MONTAVON, G., LAPUSCHKIN, S., & MÜLLER, K.-R. (2017). Evaluating the Visualization of What a Deep Neural Network has Learned. *IEEE*

- Transactions on Neural Networks and Learning Systems*, 28(11), 2660-2673.
<https://doi.org/10.1109/TNNLS.2016.2599820>
- SELVARAJU, R. R., COGSWELL, M., DAS, A., VEDANTAM, R., PARIKH, D., & BATRA, D. (2020). Grad-CAM : Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, 128(2), 336-359.
<https://doi.org/10.1007/s11263-019-01228-7>
- SIMONYAN, K., VEDALDI, A., & ZISSERMAN, A. (2014). Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv :1312.6034*.
- SIMONYAN, K., & ZISSERMAN, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*.
- STURMFELS, P., LUNDBERG, S., & LEE, S.-I. (2020). Visualizing the Impact of Feature Attribution Baselines. *Distill*. <https://doi.org/10.23915/distill.00022>
- SUNDARARAJAN, M., TALY, A., & YAN, Q. (2017). Axiomatic Attribution for Deep Networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 70, 3319-3328.
- SUVOROV, R., LOGACHEVA, E., MASHIKHIN, A., REMIZOVA, A., ASHUKHA, A., SILVESTROV, A., KONG, N., GOKA, H., PARK, K., & LEMPITSKY, V. (2022). Resolution-robust Large Mask Inpainting with Fourier Convolutions. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2149-2159.
- VEDALDI, A., & SOATTO, S. (2008). Quick Shift and Kernel Methods for Mode Seeking. *European Conference on Computer Vision (ECCV)*, 705-718.
- WICKSTRØM, K., HÖHNE, M. M.-C., & HEDSTRÖM, A. (2024). From Flexibility to Manipulation : The Slippery Slope of XAI Evaluation [Préprint arXiv:2412.05592].
- ZEILER, M. D., & FERGUS, R. (2014). Visualizing and Understanding Convolutional Networks. *Computer Vision – ECCV 2014*, 8689, 818-833. https://doi.org/10.1007/978-3-319-10590-1_53
- ZHANG, J., BARGAL, S. A., LIN, Z., BRANDT, J., SHEN, X., & SCLAROFF, S. (2018). Top-Down Neural Attention by Excitation Backprop. *International Journal of Computer Vision*, 126(10), 1084-1102.

ZHANG, Y., SONG, J., GU, S., JIANG, T., PAN, B., BAI, G., & ZHAO, L. (2023). Saliency-Bench : A Comprehensive Benchmark for Evaluating Visual Explanations [Préprint arXiv:2310.08537].

Annexe A

Annexes

A.1 Extraits de code source

Cette section présente deux extraits de code représentatifs du travail réalisé : l'implémentation de la stratégie `kmeans_colors`, proposée dans le cadre de ce stage, et le calcul de la métrique Deletion AUC.

A.1.1 Stratégie K-means Colors

Le listing [A.1](#) présente l'implémentation de la stratégie `kmeans_colors`, proposée dans le cadre de ce stage.

```
1 import numpy as np
2 from sklearn.cluster import KMeans
3
4 def kmeans_colors_mask(image, mask, k=8, random_state=42):
5     """Masquage par couleurs dominantes via K-means.
6
7     Args:
8         image: Image originale (H, W, 3), valeurs 0-255
9         mask: Masque binaire (H, W), 1 = pixels a masquer
10        k: Nombre de clusters K-means
11        random_state: Seed pour reproductibilite
12
13    Returns:
14        masked_image: Image avec regions masquées remplacees
15    """
```

```

16     h, w, c = image.shape
17     pixels = image.reshape(-1, c).astype(np.float32)
18
19     kmeans = KMeans(n_clusters=k, random_state=random_state,
20 n_init=10)
21     kmeans.fit(pixels)
22     dominant_colors = kmeans.cluster_centers_.astype(np.uint8
23 )
24
25     masked_image = image.copy()
26     mask_flat = mask.flatten()
27
28     n_masked = np.sum(mask_flat)
29     if n_masked > 0:
30         random_indices = np.random.randint(0, k, size=
31 n_masked)
32         replacement_colors = dominant_colors[random_indices]
33         masked_image_flat = masked_image.reshape(-1, c)
34         masked_image_flat[mask_flat == 1] =
35 replacement_colors
36         masked_image = masked_image_flat.reshape(h, w, c)
37
38     return masked_image

```

Listing A.1 – Implémentation de la stratégie kmeans_colors

A.1.2 Calcul du Deletion AUC

Le listing [A.2](#) illustre le calcul de la métrique Deletion AUC.

```

1 import numpy as np
2 import torch
3
4 def compute_deletion_auc(model, image, saliency, target_class
5 ,
6                             steps=100, baseline_fn=None, device=
7 'cuda'):
8     """Calcule l'AUC de deletion."""

```

```

7     if baseline_fn is None:
8         baseline_fn = lambda img, mask: gaussian_blur_mask(
img, mask)
9
10    saliency_flat = saliency.flatten()
11    sorted_indices = np.argsort(saliency_flat)[::-1]
12    n_pixels = len(saliency_flat)
13
14    scores = []
15    for step in range(steps + 1):
16        pct = step / steps
17        n_remove = int(pct * n_pixels)
18
19        mask = np.zeros(n_pixels, dtype=np.float32)
20        if n_remove > 0:
21            mask[sorted_indices[:n_remove]] = 1
22        mask = mask.reshape(saliency.shape)
23
24        perturbed = baseline_fn(image, mask)
25
26        with torch.no_grad():
27            perturbed_tensor = torch.from_numpy(
                perturbed).unsqueeze(0).to(device)
28            output = model(perturbed_tensor)
29            prob = torch.softmax(output, dim=1)[0,
target_class].item()
30
31
32        scores.append(prob)
33
34    auc = np.trapz(scores, dx=1/steps)
35    return auc, scores

```

Listing A.2 – Implémentation du Deletion AUC

A.2 Classes du dataset PASCAL VOC 2007

TABLE A.1 – Les 20 classes de PASCAL VOC 2007

Index	Classe	Index	Classe
0	aeroplane	10	diningtable
1	bicycle	11	dog
2	bird	12	horse
3	boat	13	motorbike
4	bottle	14	person
5	bus	15	pottedplant
6	car	16	sheep
7	cat	17	sofa
8	chair	18	train
9	cow	19	tvmonitor

A.3 Résultats détaillés

Le tableau A.2 présente les résultats pour chacune des 30 combinaisons méthode-stratégie.

TABLE A.2 – Résultats détaillés par combinaison méthode-stratégie (53 paires image-classe)

Combinaison	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
LIME-default	0.339	0.773	0.849	0.719	0.753	3.8	9.4 s
LIME-cube_rgb	0.355	0.770	0.774	0.749	0.759	5.0	521 s
LIME-kmeans_colors	0.308	0.789	0.906	0.723	0.782	5.2	505 s
LIME-blur	0.323	0.785	0.887	0.715	0.777	3.7	5 801 s
LIME-mean_masked	0.327	0.788	0.849	0.719	0.759	4.3	10.1 s
LIME-mean_unmasked	0.337	0.783	0.868	0.737	0.755	4.6	10.6 s
LIME-gaussian_noise	0.363	0.760	0.887	0.728	0.744	3.6	19.5 s
LIME-inpainting	0.334	0.776	0.906	0.712	0.728	4.2	9.3 s

Suite page suivante

Table A.2 – *Suite*

Combinaison	Del. ↓	Ins. ↑	PG ↑	DS ↑	DP ↑	Sp.	Temps
H2O-cube_rgb	0.372	0.714	0.717	0.836	0.799	5.3	497 s
H2O-kmeans_colors	0.432	0.697	0.623	0.881	0.820	10.1	506 s
H2O-mean_masked	0.390	0.720	0.679	0.861	0.777	7.4	32.1 s
H2O-mean_unmasked	0.434	0.672	0.679	0.882	0.753	7.1	38.9 s
H2O-blur_superpixel	0.418	0.720	0.604	0.877	0.761	8.6	239 s
H2O-gaussian_noise	0.524	0.630	0.566	0.822	0.708	11.0	46.2 s
H2O-inpainting	0.485	0.628	0.509	0.717	0.615	11.3	49.1 s
RISE-default	0.316	0.768	0.811	0.846	0.834	2.5	11.8 s
RISE-cube_rgb	0.268	0.792	0.925	0.799	0.825	2.7	13.3 s
RISE-kmeans_colors	0.262	0.796	0.943	0.821	0.827	2.8	18.7 s
RISE-mean_masked	0.268	0.803	0.925	0.802	0.829	2.8	14.8 s
RISE-mean_unmasked	0.263	0.799	0.943	0.803	0.830	2.8	14.7 s
RISE-blur	0.281	0.797	0.868	0.808	0.814	2.8	15.5 s
RISE-gaussian_noise	0.308	0.774	0.943	0.834	0.831	2.7	23.0 s
RISE-inpainting	0.365	0.735	0.811	0.875	0.838	2.4	42.8 s
HiPe-cube_rgb	0.491	0.611	0.377	0.833	0.612	15.9	2.0 s
HiPe-kmeans_colors	0.378	0.692	0.717	0.883	0.816	5.4	7.4 s
HiPe-mean_masked	0.467	0.596	0.585	0.818	0.557	18.9	1.7 s
HiPe-mean_unmasked	0.415	0.649	0.642	0.889	0.723	9.2	1.9 s
HiPe-blur	0.431	0.611	0.623	0.907	0.696	10.4	7.8 s
HiPe-gaussian_noise	0.396	0.677	0.566	0.858	0.739	7.4	26.7 s
HiPe-inpainting	0.436	0.625	0.717	0.864	0.683	7.7	34.7 s

A.4 Exemple de rapport HTML généré

Les figures A.1 et A.2 présentent des captures d’écran du rapport HTML généré automatiquement par le framework. Ce rapport constitue l’outil principal de visualisation des résultats : il inclut un tableau récapitulatif avec classement coloré, des icônes SVG pour les stratégies et des grilles de cartes de saillance comparatives par image.

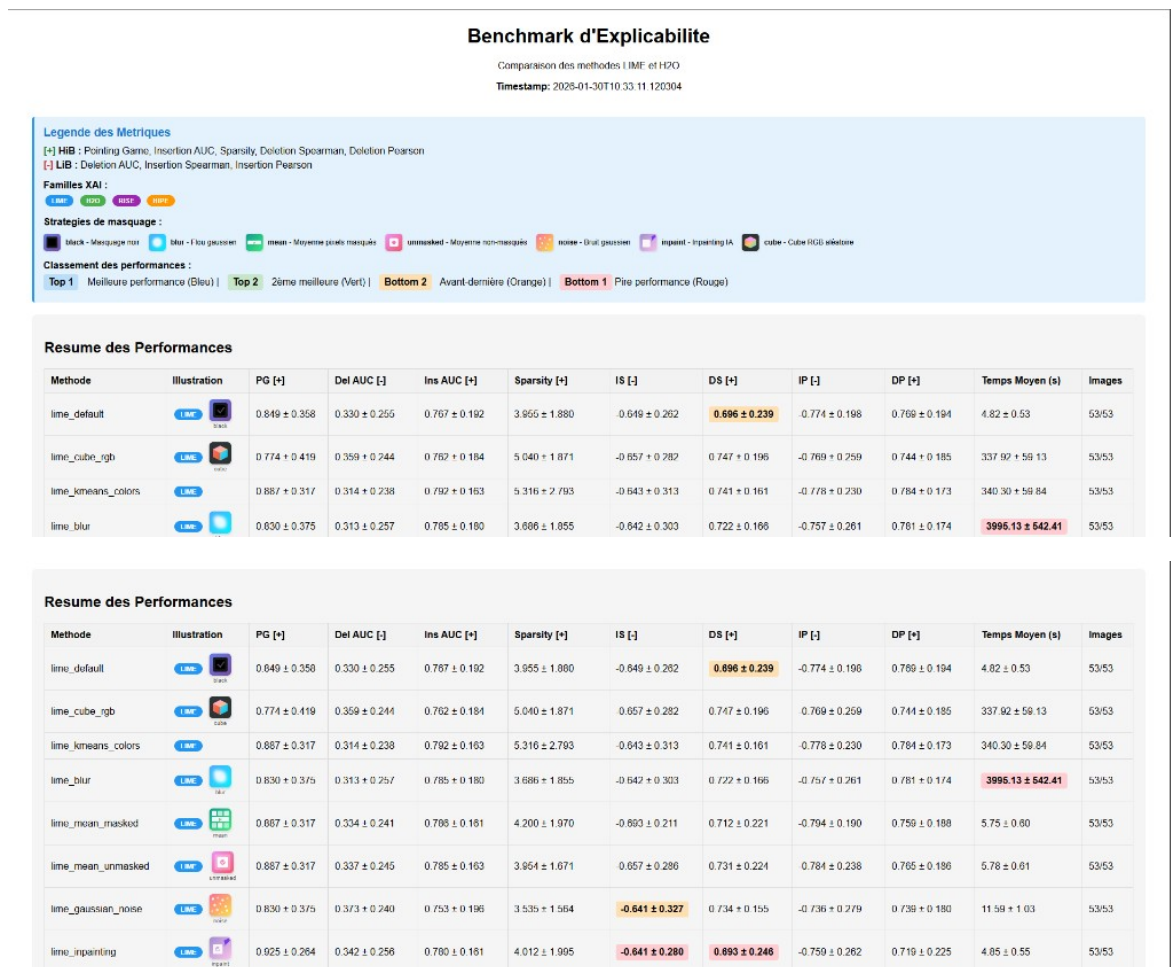


FIGURE A.1 – Rapport HTML – En-tête et tableau récapitulatif.

L'en-tête présente la légende des métriques. Le tableau récapitulatif des performances applique un classement coloré (bleu : 1^{er}, vert : 2^e, orange : avant-dernier, rouge : dernier).

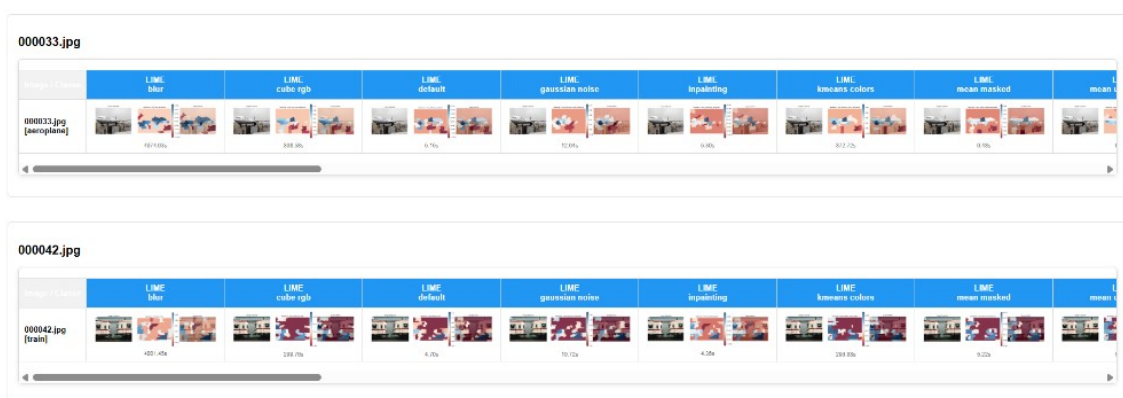


FIGURE A.2 – Rapport HTML – Section de visualisation par image.

Pour chaque image du dataset, une grille horizontale présente les cartes de saillance produites par chaque combinaison méthode-stratégie.

Annexe B

Gestion de Projet et Retour d'Expérience

Cette annexe est consacrée aux aspects organisationnels du stage : la manière dont le travail a été conduit, les phases du projet, les difficultés rencontrées et les compétences acquises. Elle complète le corps du mémoire, qui se focalise sur le contenu scientifique et technique, en apportant un regard sur la démarche et le vécu du stagiaire.

B.1 Organisation du travail

Le stage s'est déroulé du 1^{er} août 2025 au 31 janvier 2026 au sein du LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier), une unité mixte de recherche de l'Université de Montpellier et du CNRS regroupant environ 500 personnes réparties en plusieurs départements de recherche couvrant l'informatique, la robotique et la microélectronique. Le projet a été réalisé dans l'équipe ADVANSE (ADVanced ANalytics for data SciencE), qui se spécialise dans l'analyse avancée de données, incluant la visualisation d'information et l'intelligence artificielle appliquée. L'encadrement quotidien était assuré par Luc Etienne Pommé, postdoctorant et auteur de la méthode H2O (POMMÉ et al., [2023](#)), avec un suivi par Arnaud Sallaberry, Professeur des Universités, en tant que tuteur de stage. Le suivi académique côté IFI était assuré par Ho Tuong Vinh, enseignant-chercheur.

Le stage s'est effectué entièrement à distance. Pour maintenir un rythme de travail régulier malgré cette contrainte, je m'imposais des horaires fixes en semaine comme le week-end, organisés autour des tâches à accomplir pour la réunion suivante. Je tenais

un carnet de notes pour consigner les idées, les résultats intermédiaires et les questions à poser.

Le travail reposait sur des réunions hebdomadaires avec l'encadrant, généralement le jeudi. Pour chaque réunion, je préparais des slides récapitulant les avancées de la semaine ainsi que les résultats des exécutions. L'encadrant fixait ensuite les tâches pour la semaine suivante, en précisant parfois un ordre de priorité. Entre les réunions, j'échangeais ponctuellement avec lui via Discord pour poser des questions ou partager des résultats préliminaires, ce qui permettait de débloquer rapidement certaines situations sans attendre le jeudi.

Le développement s'est fait avec l'IDE Cursor, dans un environnement Python géré via pip. Le code était hébergé sur un dépôt GitHub privé partagé avec l'encadrant : j'y poussais mes modifications au fil de la semaine, et lors des réunions, nous passions en revue le code ensemble. Mon encadrant me fournissait également des scripts à adapter et à intégrer dans le framework (notamment pour HiPe et RISE). N'ayant pas accès aux ressources de calcul du laboratoire, j'exécutais le benchmark sur ma machine personnelle, ce qui imposait de paramétrer les stratégies avec des valeurs réduites pour obtenir des résultats dans un temps raisonnable.

Le travail à distance a posé quelques difficultés spécifiques. Le décalage horaire de cinq à six heures entre le Vietnam et la France faisait que les messages sur Discord étaient parfois lus tardivement, ce qui pouvait retarder un déblocage. Certaines notions auraient gagné à être expliquées sur un tableau blanc en présentiel, même si nous parvenions à les schématiser numériquement. Malgré les réunions régulières et les échanges sur Discord, un sentiment d'isolement persistait par moments. J'aurais préféré être en présentiel pour un meilleur contact humain, bien que les interactions à distance aient toujours été respectueuses, cordiales et enrichissantes.

B.2 Planning du stage

Le stage s’est organisé en plusieurs phases, dont les dates approximatives sont données ci-dessous.

TABLE B.1 – Phases du stage

Période	Phase	Description
Août 2025	Prise en main	Installation de l’environnement, prise en main du code existant (H2O), premières lectures bibliographiques.
Sept.–Oct. 2025	Bibliographie et conception	Étude approfondie de LIME, RISE, HiPe. Conception de l’architecture modulaire du framework. Premiers prototypes.
Nov.–Déc. 2025	Implémentation	Intégration des quatre méthodes, implémentation des stratégies de masquage, développement du système de métriques et du rapport HTML.
Déc. 2025–Jan. 2026	Expérimentation	Exécution des benchmarks, analyse des résultats, développement du mode fill_missing.
Janvier 2026	Rédaction	Rédaction du rapport de stage, dernières corrections du code, préparation de la soutenance.

Il est important de noter que ces phases ne sont pas strictement séquentielles : par exemple, l’implémentation a commencé avant la fin de la bibliographie, et des corrections de code ont été nécessaires pendant la phase d’expérimentation.

B.3 Difficultés rencontrées et solutions

Le déroulement du stage n’a pas été exempt de difficultés, tant sur le plan technique que sur le plan organisationnel. J’en décris ici les principales, en les reliant à leur

impact sur la conduite du projet.

B.3.1 Prise en main du code existant

La première phase du stage a été consacrée à la compréhension du pipeline H2O, qui comporte six étapes (segmentation SLIC, génération des masques, calcul des combinaisons, prédiction, calcul de la dégradation et agrégation). Plusieurs problèmes de format non documentés ont ralenti cette prise en main : l'image devait être convertie en float32 et normalisée dans $[0, 1]$ avec une dimension batch ajoutée, faute de quoi SLIC produisait des superpixels dégradés. Une incohérence entre les plages de valeurs (float $[0, 1]$ pour l'image, $[0, 255]$ pour le flou gaussien) a également causé des prédictions nulles, nécessitant plusieurs échanges sur Discord avec l'encadrant avant d'identifier la cause. Cette phase a mobilisé les premières semaines du stage et retardé le début des expérimentations.

B.3.2 Évolution du périmètre du projet

Le périmètre du stage a évolué progressivement au fil des réunions. Initialement centré sur H2O et quelques stratégies de masquage, le projet s'est étendu à l'intégration de LIME, puis de RISE et HiPe, et enfin de l'inpainting avec LaMa. Chaque nouvelle méthode a nécessité une phase de compréhension de l'algorithme, d'adaptation au framework et de débogage spécifique. L'intégration de LIME a été la plus coûteuse : la méthode `data_labels()`, conçue pour un fonctionnement en ratio 1:1 (un vecteur binaire produit une seule image perturbée), a dû être refondue pour gérer les stratégies multi-couleurs ($|C|$ images par vecteur) et le flou (K images par vecteur). Cette refonte, non prévue initialement, a mobilisé près de deux semaines et nécessité plusieurs échanges avec l'encadrant pour valider l'approche avant de l'implémenter.

B.3.3 Coexistence TensorFlow/PyTorch

L'implémentation de H2O importe TensorFlow (hérité de son code d'origine), tandis que les autres méthodes et le modèle ResNet-18 lui-même reposent sur PyTorch. Lorsque les deux frameworks tentent d'allouer la mémoire GPU simultanément, des conflits apparaissent, pouvant aller jusqu'au crash du processus. La solution adoptée a consisté à isoler les contextes d'exécution : avant chaque appel à H2O, la mémoire PyTorch est explicitement libérée ; après l'exécution, c'est la session TensorFlow qui est nettoyée. Sur Windows, un problème supplémentaire de DLL CUDA/cuDNN

incompatibles a nécessité une installation manuelle des bibliothèques. Ce problème a retardé le début des expérimentations d'environ une semaine et a été remonté lors d'une réunion hebdomadaire, où l'encadrant a confirmé que la priorité était d'assurer la stabilité plutôt que la performance GPU.

B.3.4 Définition progressive des métriques

Les métriques d'évaluation n'ont pas été fixées dès le début du stage mais définies progressivement au fil des réunions. Les premières métriques implémentées (Deletion AUC, Insertion AUC, Pointing Game) ont été complétées par les corrélations de Spearman (Insertion et Deletion), puis par les métriques IPGS et DNGS, ces dernières ayant finalement été écartées au profit de la corrélation de Pearson et de la Sparsité. Cette définition incrémentale a entraîné plusieurs refactorisations du code de benchmark et des bugs dans les calculs intermédiaires : l'Insertion AUC se retrouvait plus élevé que le Deletion AUC en raison d'un calcul cumulé au lieu de non cumulé, et le Spearman utilisait les index des pixels au lieu de leurs valeurs. Ces corrections ont nécessité des allers-retours avec l'encadrant via Discord pour valider les formules.

B.3.5 Variabilité des résultats LIME

L'échantillonnage aléatoire dans LIME produisait des résultats variables d'une exécution à l'autre, ce qui compliquait la comparaison et la reproductibilité. Ce point a été identifié lors d'une réunion hebdomadaire où les résultats présentés différaient de ceux de la semaine précédente, ce qui a conduit à mettre en place un protocole strict de reproductibilité : fixation de tous les seeds aléatoires (Python, NumPy, PyTorch, CUDA) et augmentation du nombre d'échantillons à 1 000.

B.4 Compétences acquises

Ce stage m'a permis de progresser dans des directions que je n'avais pas anticipées. La coexistence de PyTorch et TensorFlow dans un même pipeline m'a confronté à des problèmes concrets de gestion mémoire GPU et de compatibilité CUDA que l'on ne rencontre pas dans les projets académiques classiques. J'ai acquis au passage une compréhension plus fine des architectures CNN, du traitement d'images (segmentation SLIC, QuickShift, inpainting neuronal) et du calcul de métriques statistiques (corrélations de Spearman et Pearson).

Le développement du framework m'a aussi appris la valeur d'une architecture modulaire et de la configuration par fichiers YAML : lorsqu'il a fallu ajouter une septième stratégie ou relancer un benchmark partiel, c'est cette modularité qui a rendu l'opération rapide. Le versioning avec Git et la génération de rapports HTML ont complété cet apprentissage d'ingénierie logicielle.

Le domaine de l'explicabilité en IA m'était peu familier avant ce stage. J'ai dû apprendre à distinguer les familles de méthodes (gradient, propagation de pertinence, perturbation), à lire les articles fondateurs et à comprendre les métriques d'évaluation associées.

Les réunions hebdomadaires m'ont aussi appris à structurer mes résultats intermédiaires et à ajuster mes priorités. Plus largement, j'ai appris à gérer un projet de six mois en autonomie et à communiquer mes résultats aussi bien à l'oral qu'à l'écrit.