



Université de la Manouba
École Nationale des Sciences de l'Informatique



RAPPORT DE CONCEPTION ET DE DÉVELOPPEMENT

**Sujet : Reconnaissance d'images falsifiées avec
le Transfer Learning dans les procédures
criminalistiques**

Auteurs :

M. Khalil KRIFI M^{lle}. Siwar HADDAD

M. Mohamed Aziz SOUID

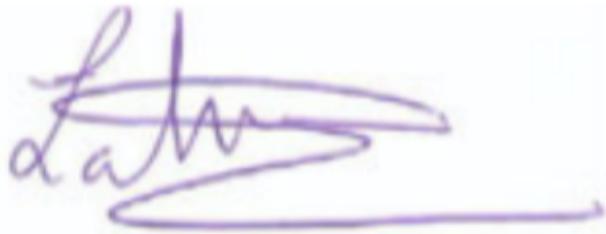
Encadrant :

Pr. Lassaad LATRACH Pr. Assoc. Jaouhar FATTAHI

Appréciations et signature de l'encadrant

LATRACH LASSAAD

Professeur à l'Ecole Nationale des sciences de l'Informatique



Jaouhar Fattah, Ph.D.

Laboratoire de sécurité informatique, Université Laval



Résumé

À une époque marquée par la manipulation numérique omniprésente, garantir l'authenticité des preuves visuelles est essentiel dans les procédures judiciaires. Ce rapport présente une étude approfondie sur le développement d'un système avancé de détection d'images falsifiées, avec un accent particulier sur les signatures manuscrites, un élément clé dans les contextes légaux.

En exploitant le Transfer Learning, nous avons évalué huit modèles de pointe — ResNetRS50, Xception, MobileNetV3_Large, EfficientNetV2-S, ConvNeXt_Base, ViT_Base, DeiT_Base et BEiT_Large — optimisés pour la classification binaire des signatures authentiques par rapport aux signatures falsifiées.

Pour renforcer la transparence, nous avons intégré des techniques d'Intelligence Artificielle Explicable (XAI), offrant des visualisations claires des décisions des modèles pour les acteurs judiciaires.

La solution est intégrée de manière fluide dans une plateforme web modulaire basée sur une architecture de microservices, garantissant sécurité, évolutivité et traçabilité.

Mots clés : Détection des images falsifiées, Transfer Learning, Signatures manuscrites, Explainable Artificial Intelligence (XAI), Microservices.

Abstract

In the era of pervasive digital manipulation, ensuring the authenticity of visual evidence is paramount in judicial proceedings. This report presents a comprehensive study on the development of an advanced system for detecting falsified images, with a focus on handwritten signatures, a critical element in legal contexts.

Leveraging Transfer Learning, we evaluated eight state-of-the-art models—ResNetRS50, Xception, MobileNetV3_Large, ViT_Base, DeiT_Base, BEiT_Large, EfficientNetV2-S, and ConvNeXt_Base—optimized for binary classification of authentic versus forged signatures.

To enhance transparency, we integrated Explainable AI (XAI) techniques, providing clear visualizations of model decisions for judicial stakeholders.

The solution is seamlessly embedded within a modular web platform built on a microservices architecture, ensuring security, scalability, and traceability.

Keywords : Image Forgery Detection, Transfer Learning, Handwritten Signatures, Explainable Artificial Intelligence (XAI), Microservices.

Remerciements

À l'issue de ce projet, nous souhaitons prendre un moment pour rendre hommage à celles et ceux qui ont rendu possible la réalisation de ce système de détection d'images falsifiées (FIDS). Ce travail, fruit d'une collaboration, n'aurait pas vu le jour sans l'implication de nombreuses personnes que nous tenons à remercier du fond du cœur.

Nous exprimons notre profonde gratitude à nos encadrants Pr. Lassaad LATRACH et Dr. Jaouhar Fattahi, pour leur accompagnement, leurs conseils éclairés et leur disponibilité tout au long de ce projet. Leur expertise et leur patience nous ont permis de relever les nombreux défis techniques et conceptuels rencontrés.

Un grand merci également à nos collègues et camarades, qui ont partagé ce parcours avec nous. Leur collaboration, leurs idées innovantes et leur aide lors des phases de développement et de test ont été essentielles à la réussite de ce projet. Leur esprit d'équipe et leur enthousiasme nous ont constamment motivés.

Enfin, nous adressons nos remerciements les plus chaleureux à nos familles et amis, pour leur soutien indéfectible, leurs encouragements et leur compréhension tout au long de cette aventure. Leur présence et leur amour nous ont portés dans les moments de doute et nous ont donné la force d'aller au bout de ce projet.

À vous tous, un immense merci pour avoir cru en nous et pour avoir contribué à faire de ce projet une réalité. Votre soutien a fait toute la différence.

Table des matières

Introduction générale	1
1 Etat de l'art	3
1.1 Techniques de falsification des images	4
1.2 Approches de détection	4
1.3 Modèles de Transfer Learning	5
1.3.1 ResNetRS50	5
1.3.2 Xception	6
1.3.3 MobileNetV3_Large	7
1.3.4 ViT_Base (Vision Transformer)	8
1.3.5 DeiT_Base (Data-efficient image Transformer)	9
1.3.6 BEiT_Large (Bidirectional Encoder representation from Image Transformers)	10
1.3.7 EfficientNetV2-S	11
1.3.8 ConvNeXt_Base	12
1.3.9 Comparaison analytique des architectures pour la détection de falsifications	13
1.4 Explicabilité de l'Intelligence Artificielle (XAI)	13
1.5 Architectures microservices	15
2 Analyse et conception	16
2.1 Analyse des besoins	17
2.1.1 Identification des acteurs	17
2.1.2 Besoins fonctionnels	18
2.1.3 Besoins non fonctionnels	19
2.1.4 Contraintes judiciaires	20
2.1.5 Diagrammes de l'analyse	21
2.2 Conception de l'architecture	27
2.2.1 Vue d'ensemble	27
2.2.2 Diagramme de composants	27

TABLE DES MATIÈRES

3 Méthodologie	29
3.1 Présentation générale	30
3.2 Collecte et préparation des données	30
3.2.1 Collecte des données	30
3.2.2 Préparation des Données	31
3.3 Étude et développement des modèles	31
3.3.1 Sélection des modèles	32
3.3.2 Adaptation des modèles	32
3.3.3 Entraînement	33
3.3.4 Comparaison des modèles	34
3.4 Évaluation des performances	34
3.5 Intégration de l'explicabilité	35
3.5.1 Méthodes sélectionnées	35
3.5.2 Objectifs	36
4 Implémentation	37
4.1 Choix de l'environnement technique	38
4.2 Implémentation du Transfer Learning	39
4.3 Implémentation des microservices	40
4.4 Intégration et déploiement	42
4.5 Conception des interfaces graphiques	42
5 Résultats et Analyse	48
5.1 Résultats des modèles	49
5.2 Discussion des résultats obtenus	50
5.2.1 Analyse Comparative des Modèles	50
5.2.2 Analyse Comparative des Modèles CNN vs. Transformers	52
5.2.3 Facteurs influençant les performances	54
5.3 Résultats et analyse de l'explicabilité (XAI)	54
5.4 Critiques et limites	56
Conclusion et perspectives	58
Bibliographie	61

Table des figures

1.1	Architecture du ResNetRS50 [26]	6
1.2	Architecture du Xception [1]	7
1.3	Architecture du MobileNetV3	8
1.4	Architecture du ViT [22]	8
1.5	Architecture du DeiT (1)[28]	9
1.6	Architecture du DeiT (2)[22]	9
1.7	Architecture du BEiT [22]	10
1.8	Architecture du EfficientNetV2 [2]	11
1.9	Architecture du ConvNeXt [4]	12
1.10	méthodes XAI (Grad-CAM, LIME, and SHAP) avec quatre différentes images. [20, 7]	14
1.11	architecture microservices [19]	15
2.1	Diagramme de cas d'utilisation	21
2.2	Diagramme de classes	22
2.3	Diagramme de séquence : S'authentifier au système	23
2.4	Diagramme de séquence : Produire et gérer les rapports	24
2.5	Diagramme de séquence détaillé : Produire et gérer les rapports	25
2.6	Diagramme d'activité : Gérer les preuves visuelles	26
2.7	Diagramme de composants	28
3.1	Matrice de confusion	35
4.1	Interface des Statistiques du Système (Vue Admin)	44
4.2	Interface de Bibliothèque d'Images (Vue DéTECTIVE)	44
4.3	Interface de Téléversement d'Image Initial (Vue DéTECTIVE)	45
4.4	Interface des Rapports d'Expert (Vue Expert)	45
4.5	Interface de Révision de Document Juridique (Vue Avocat)	46
4.6	Interface d'Analyse d'Image	46
4.7	Interface de Revue des affaires (Vue Juge)	47
5.1	Histogramme comparant les performances des modèles sur l'ensemble de test	49

TABLE DES FIGURES

5.2	Résultat du modèle MobileNetV3_Large	50
5.3	Résultat du modèle Deit_Base	51
5.4	Résultat du modèle Xception	51
5.5	Résultat du modèle ViT_Base	51
5.6	Résultat du modèle BEiT_Base	52
5.7	Résultat du modèle ConvNeXt_Base	52
5.8	Résultat du modèle ResNetRS50	53
5.9	Visualisations XAI	55

Liste des tableaux

2.1	Liste des acteurs	18
2.2	Besoins fonctionnels	19
2.3	Besoins non fonctionnels	20
3.1	Aperçu des modèles étudiés	32
3.2	Métriques quantitatives	34
5.1	Comparaison des performances des modèles sur l'ensemble de test	50

Liste des acronymes

API	Application Programming Interface
BEiT	Bidirectional Encoder representation from Image Transformers
CNN	Convolutional Neural Network
CORS	Cross-Origin Resource Sharing
DCT	Discrete Cosine Transform
DeiT	Data-efficient Image Transformer
EXIF	Exchangeable Image File Format
GAN	Generative Adversarial Network
Grad-CAM	Gradient-weighted Class Activation Mapping
HTTPS	Hypertext Transfer Protocol Secure
JWT	JSON Web Token
KPI	Key Performance Indicator
LIME	Local Interpretable Model-agnostic Explanations
MLP	Multi-Layer Perceptron
NAS	Neural Architecture Search
ReLU	Rectified Linear Unit
SHA-256	Secure Hash Algorithm 256-bit
SHAP	SHapley Additive exPlanations
ViT	Vision Transformer
XAI	Explainable Artificial Intelligence

Introduction générale

La montée en puissance des images falsifiées représente un défi majeur dans de nombreux secteurs, en particulier dans les procédures judiciaires où l'authenticité des preuves visuelles est essentielle. Ces manipulations, qu'elles reposent sur des retouches numériques ou des imitations manuelles, peuvent altérer des documents, des photographies ou d'autres éléments probants, compromettant ainsi la fiabilité des systèmes légaux.

Les approches traditionnelles de détection, souvent tributaires de l'expertise humaine, peinent à répondre aux exigences de rapidité, de précision et de volume imposées par la multiplication des données à traiter.

Dans ce contexte, les progrès en intelligence artificielle, notamment dans le domaine de la vision par ordinateur, offrent une opportunité prometteuse pour automatiser et renforcer la détection des falsifications. En combinant des techniques avancées d'apprentissage automatique avec une plateforme web sécurisée, il devient envisageable de proposer une solution efficace, capable de répondre aux besoins spécifiques d'un cadre judiciaire tout en s'adaptant à une variété de cas d'utilisation.

Conscient de cette logique, nous avons pris l'initiative de développer un système pour identifier les images falsifiées, en prenant comme cas d'étude les signatures manuscrites, un exemple concret et pertinent dans un cadre légal.

Dans ce cadre, notre stratégie repose sur le Transfer Learning pour analyser les images et distinguer les authentiques des contrefaites ;

Suivi parallèlement de l'élaboration d'une plateforme web basée sur une architecture microservices pour intégrer cette solution dans un système pratique.

Le projet poursuit plusieurs objectifs interconnectés. Il cherche à évaluer et perfectionner des modèles d'apprentissage pour garantir une détection efficace des falsifications, en s'appuyant sur un cas d'étude représentatif. Il vise également à intégrer ces capacités dans une plateforme web intuitive et modulaire, répondant à des attentes fonctionnelles, comme l'analyse d'images et la production de rapports, tout en respectant des contraintes de sécurité et de traçabilité.

Ce rapport retrace notre démarche dans son ensemble. Il débute par un état de l'art dé-

crivant les techniques de falsification et les solutions de détection, suivi d'une analyse des besoins identifiant les acteurs et les exigences de la plateforme avec une conception. La méthodologie expose le développement des modèles. Les chapitres suivantes présentent l'implémentation, les résultats, une discussion critique et des perspectives, avant de conclure sur les contributions de notre approche.

Chapitre 1

Etat de l'art

Introduction

Dans ce chapitre, nous examinons les avancées et les concepts clés liés à la détection des images falsifiées dans un cadre technologique et applicatif. Nous explorons aussi successivement les techniques de falsification des images, les méthodes de détection, les approches de Transfer Learning, l'explicabilité de l'Intelligence Artificielle (IA), et les architectures microservices, qui constituent les piliers du projet.

1.1 Techniques de falsification des images

Les techniques de falsification des images se divisent principalement en deux catégories : l'imitation manuelle et la retouche numérique.

- **L'imitation manuelle** Cette méthode artisanale repose sur la reproduction manuelle d'éléments visuels, comme des signatures ou des annotations. Les altérations sont souvent détectables par des incohérences dans les textures ou les alignements, mais leur identification nécessite une expertise approfondie
- **Les retouches numériques** exploitent des outils logiciels, allant de modifications simples (ajustement de couleurs, suppression d'objets) à des transformations complexes basées sur l'intelligence artificielle. Goodfellow et al. (2016) décrivent les réseaux adversatifs génératifs (GANs) comme des outils puissants pour produire des images synthétiques quasi indiscernables de la réalité, ce qui complique leur détection. [12]

1.2 Approches de détection

Les approches de détection des images falsifiées opposent les méthodes traditionnelles aux techniques basées sur le Deep Learning.

- **Les méthodes traditionnelles** analysent des indices physiques ou numériques, comme les métadonnées EXIF ou les incohérences dans les motifs de bruit. Farid (2009) explique que "l'analyse des statistiques de compression JPEG peut identifier des manipulations en détectant des discontinuités dans les blocs d'image" [10]. Popescu et Farid (2005) précisent que "les corrélations dans les coefficients de transformation en cosinus discrète (DCT) révèlent souvent des altérations subtiles" [21]. Cependant, ces techniques sont limitées face aux falsifications sophistiquées qui éliminent ces artefacts.
- **Le Deep Learning** offre une alternative puissante en apprenant automatiquement des caractéristiques discriminantes. Verdoliva (2020) note que "les réseaux convolutifs (CNNs) surpassent les approches traditionnelles en capturant des motifs complexes à partir de grands ensembles de données" [30]. Li et al. (2018) ajoutent que "les modèles entraînés sur des datasets spécifiques de falsifications améliorent la détection des deepfakes en analysant les incohérences temporelles ou spatiales" [15].

Étude des Solutions Existantes : Dans ce même contexte, plusieurs solutions ont été proposées dans la littérature et l'industrie. *ForensicFocus* s'appuie sur des méthodes statistiques pour détecter les artefacts de compression, mais échoue sur les images recompressées [10]. *Deepware Scanner* (<https://deepware.ai/>) utilise des CNNs pour identifier les deepfakes, avec une précision dépendante de la qualité des données d'entraînement. Wang et al. (2020) ont développé un système basé sur ResNet pour détecter les manipulations de visages, atteignant un taux de succès élevé sur des datasets comme *FaceForensics++* [31]. Zhang et al. (2019) proposent une approche hybride combinant CNNs et analyse statistique, améliorant la robustesse face aux falsifications variées [32].

1.3 Modèles de Transfer Learning

Le Transfer Learning est une approche clé pour la détection des images falsifiées, exploitant des modèles pré-entraînés sur des datasets massifs comme ImageNet.

Cette approche réduit les besoins en données et en ressources computationnelles, tout en améliorant les performances sur des cas d'utilisation ciblés.

Les modèles suivants illustrent la diversité des architectures adaptées à ce problème.

1.3.1 ResNetRS50

— **Présentation :** ResNetRS50, une version optimisée de ResNet, a été introduit par He et al. en 2016 pour résoudre les problèmes de dégradation des performances dans les réseaux profonds [13].

Grâce à ses connexions résiduelles, ce modèle permet l'entraînement de réseaux très profonds tout en améliorant l'efficacité, ce qui le rend adapté à des tâches complexes comme la classification d'images.

— **Architecture et Conception :** ResNetRS50 introduit des connexions résiduelles pour faciliter l'apprentissage des couches profondes.

- **Blocs de construction :** Blocs ResNet, composés de convolutions avec connexions résiduelles (skip connections), comme illustré dans la figure 1.1.
- **Caractéristiques distinctives :**
 - Connexions résiduelles pour éviter la disparition des gradients.
 - Normalisation par lots pour stabiliser l'entraînement.
 - Activation ReLU pour introduire de la non-linéarité.
 - Taille d'entrée : 224×224 pixels
- **Paramètres :** Environ 25 millions

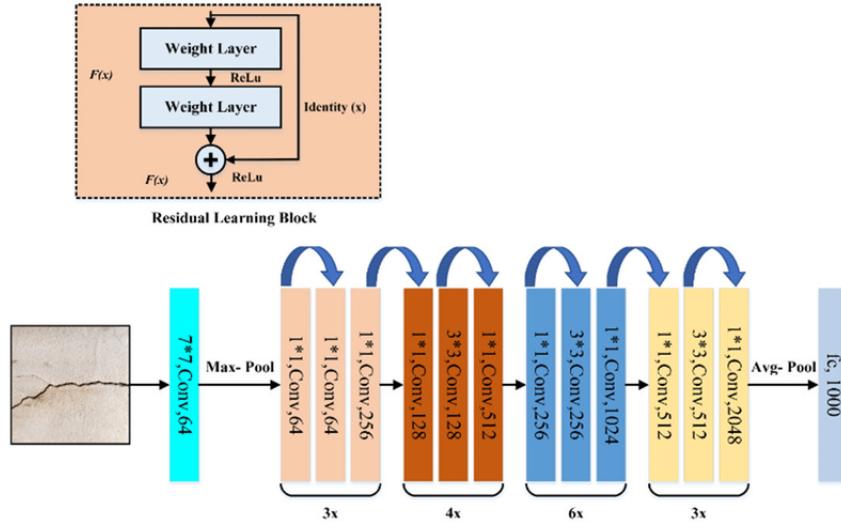


FIGURE 1.1 – Architecture du ResNetRS50 [26]

- **Pertinence pour la Détection d’Images Falsifiées** : ResNetRS50 offre une stabilité dans l’extraction des caractéristiques, comme l’a démontré Shetty et al. (2023) dans l’identification d’implants médicaux à partir d’images radiographiques [26]. Cette stabilité est essentielle pour détecter des images falsifiées dans des contextes bruités, un scénario fréquent dans les applications judiciaires.

1.3.2 Xception

- **Présentation** : Xception, proposé par Chollet en 2017, est une évolution de l’architecture Inception, visant à réduire la complexité computationnelle des CNN tout en améliorant leur précision [5]. En remplaçant les modules Inception par des convolutions séparables en profondeur, Xception offre un équilibre entre efficacité et performance, le rendant adapté à des applications nécessitant une analyse détaillée.
- **Architecture et Conception** : Xception simplifie l’approche Inception en utilisant des convolutions séparables pour optimiser les performances.
 - **Blocs de construction** : Blocs de convolutions séparables en profondeur (Depthwise Separable Convolutions), comme illustré dans la figure 1.2.
 - **Caractéristiques distinctives** :
 - Séparation spatiale et par canal pour réduire les paramètres.
 - Activation ReLU pour introduire de la non-linéarité.
 - Connexions résiduelles pour faciliter l’entraînement.
 - Taille d’entrée : 299×299 pixels
 - **Paramètres** : Environ 23 millions

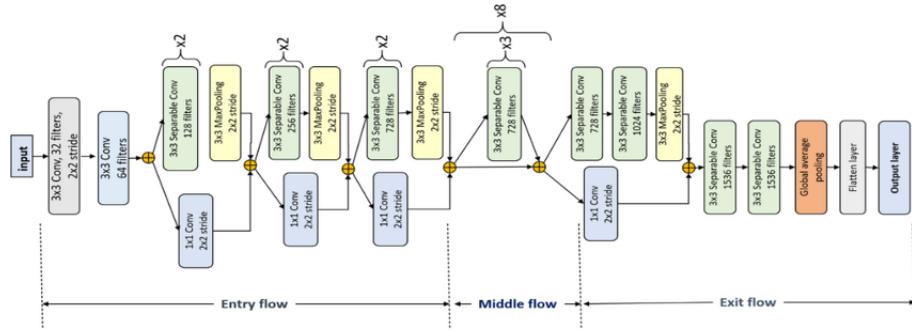


FIGURE 1.2 – Architecture du Xception [1]

- **Pertinence pour la Détection d'Images Falsifiées** : Xception excelle dans la capture des détails multi-échelles, comme l'a montré Abbas et al. (2023) dans la classification d'images de cancer du sein [1].
Cette capacité est importante pour détecter des incohérences subtiles dans les images falsifiées, un enjeu clé dans un contexte judiciaire.
Nous évaluerons sa performance pour identifier des manipulations dans notre étude.

1.3.3 MobileNetV3_Large

- **Présentation** : MobileNetV3_Large, développé par Howard et al. en 2019, est une architecture CNN légère conçue pour les appareils à ressources limitées, comme les smartphones [14].
Grâce à la recherche d'architecture neuronale (NAS), ce modèle améliore la précision tout en réduisant les coûts computationnels, le rendant idéal pour des applications nécessitant une analyse rapide.
- **Architecture et Conception** : MobileNetV3_Large optimise les CNN pour des performances élevées sur des appareils contraints.
 - **Blocs de construction** : Blocs inversés (Inverted Residual Blocks) avec modules Squeeze-and-Excitation (SE), comme illustré dans la figure 1.3.
 - **Caractéristiques distinctives** :
 - Convolutions séparables pour réduire les calculs.
 - Activation h-swish pour minimiser les pertes d'information.
 - Attention légère via les modules SE.
 - Taille d'entrée : 224×224 pixels
 - **Paramètres** : Environ 5.5 millions
- **Pertinence pour la Détection d'Images Falsifiées** : MobileNetV3_Large est adapté à la détection rapide des incohérences locales grâce à son efficacité computationnelle. Des études récentes ont démontré son efficacité dans la classification binaire de signatures falsifiées, un cas d'usage clé dans notre projet.
Sa légèreté le rend idéal pour une intégration dans une plateforme web, et nous éva-

luerons sa performance pour distinguer les images authentiques des falsifiées.

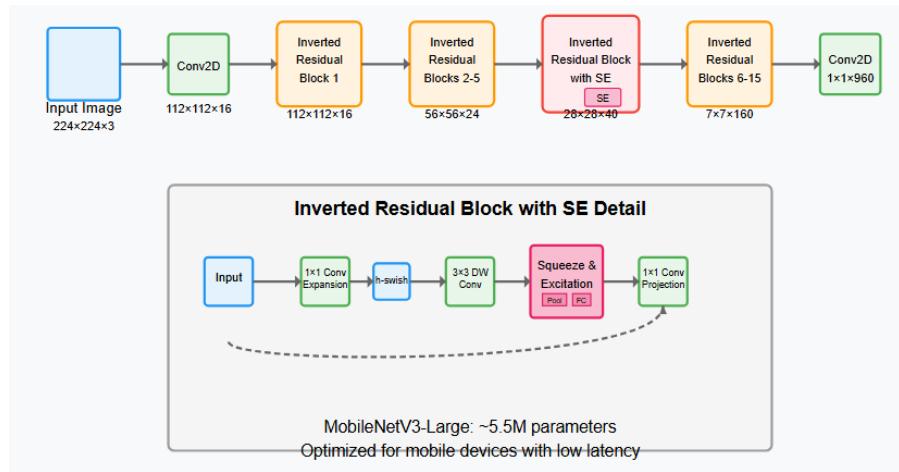


FIGURE 1.3 – Architecture du MobileNetV3

1.3.4 ViT_Base (Vision Transformer)

- **Présentation :** ViT_Base (Vision Transformer), introduit par Dosovitskiy et al. en 2020, marque une transition des CNN vers les Transformers en vision par ordinateur [8].
En divisant les images en patchs et en utilisant des mécanismes d'attention, ViT capture des relations globales, une approche qui surpassé les CNN pour certaines tâches contextuelles, bien qu'elle soit gourmande en ressources.
- **Architecture et Conception :** ViT_Base applique des mécanismes de Transformers pour modéliser les relations globales dans les images.
 - **Blocs de construction :** Blocs Transformer avec multi-head self-attention et MLP blocks, comme illustré dans la figure 1.4.
 - **Caractéristiques distinctives :**
 - Patch embeddings pour diviser les images en segments.
 - Attention positionnelle pour modéliser les relations entre patchs.
 - Normalisation par couches pour stabiliser l'entraînement.
 - Taille d'entrée : 224×224 pixels
 - **Paramètres :** Environ 86 millions



FIGURE 1.4 – Architecture du ViT [22]

- **Pertinence pour la Détection d'Images Falsifiées :** ViT_Base est efficace pour modéliser des relations globales, comme l'a démontré Ramineni et al. (2024) dans

l'analyse d'images médicales [22].

Cette propriété est utile pour détecter des falsifications structurelles affectant l'ensemble d'une image, un scénario pertinent dans un contexte judiciaire. Nous évaluerons son efficacité dans notre projet.

1.3.5 DeiT_Base (Data-efficient image Transformer)

— **Présentation** : DeiT_Base (Data-efficient Image Transformer), proposé par Touvron et al. en 2020, est une avancée dans l'utilisation des Transformers pour la vision par ordinateur [29].

Conçu pour réduire la dépendance des Transformers aux grands volumes de données, DeiT utilise une distillation de connaissances pour atteindre des performances élevées avec moins de données d'entraînement.

— **Architecture et Conception** : DeiT_Base optimise l'apprentissage des Transformers en intégrant des techniques de régularisation et de distillation.

- **Blocs de construction** : Blocs Transformer, incluant des mécanismes de multi-head self-attention et des MLP blocks, avec normalisation par couches, comme illustré dans les figures 1.5 et 1.6.
- **Caractéristiques distinctives** :
 - Distillation de connaissances (apprentissage guidé par un CNN enseignant)
 - Régularisation forte pour éviter le surapprentissage.
 - Utilisation de patch embeddings pour diviser les images.
 - Taille d'entrée : 224×224 pixels
- **Paramètres** : Environ 86 millions

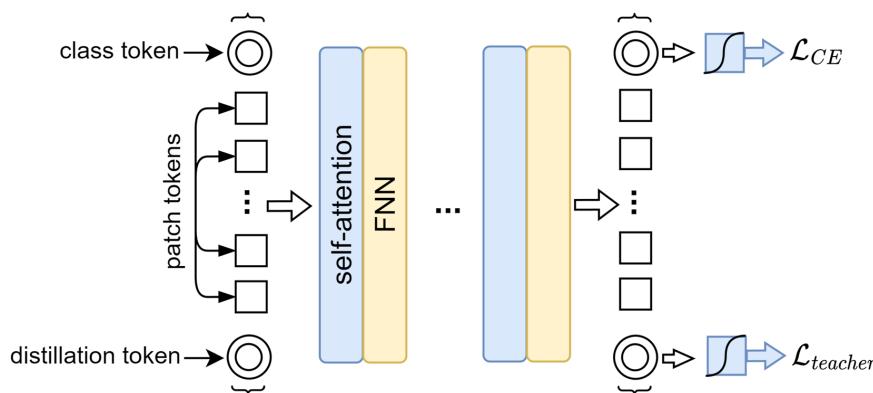


FIGURE 1.5 – Architecture du DeiT (1)[28]

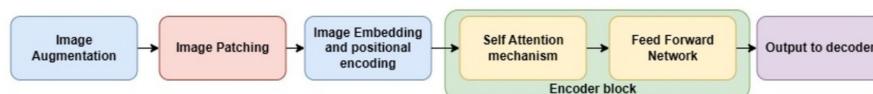


FIGURE 1.6 – Architecture du DeiT (2)[22]

- **Pertinence pour la Détection d'Images Falsifiées** : DeiT_Base excelle dans la capture des relations globales, comme démontré par Teng et al. (2023) dans la classification multi-label d'images [28].

Cette propriété est essentielle pour détecter des incohérences contextuelles dans les images falsifiées, notamment dans un contexte judiciaire où les modifications结构uelles doivent être identifiées.

Nous évaluerons sa capacité à distinguer les images authentiques des falsifiées dans notre projet.

1.3.6 BEiT_Large (Bidirectional Encoder representation from Image Transformers)

- **Présentation** : BEiT_Large (Bidirectional Encoder representation from Image Transformers), introduit par Bao et al. en 2021, s'inspire de BERT pour proposer un apprentissage auto-supervisé en vision par ordinateur [3]. En utilisant une approche de modélisation masquée, BEiT apprend des représentations visuelles sans étiquettes, une innovation qui le rend adapté aux domaines où les données annotées sont rares.
- **Architecture et Conception** : BEiT_Large repose sur une méthode de pré-entraînement auto-supervisé pour capturer des motifs contextuels.
 - **Blocs de construction** : Blocs Transformer avec une approche de modélisation masquée (masked image modeling), comme illustré dans la figure 1.7.
 - **Caractéristiques distinctives** :
 - Apprentissage contextuel des patchs masqués.
 - Pré-entraînement auto-supervisé sans étiquettes.
 - Attention multi-échelle pour une analyse détaillée.
 - Taille d'entrée : 224×224 pixels
 - **Paramètres** : Environ 307 millions

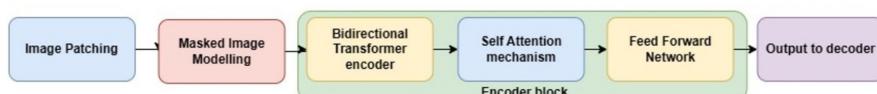


FIGURE 1.7 – Architecture du BEiT [22]

- **Pertinence pour la Détection d'Images Falsifiées** : BEiT_Large est efficace pour l'analyse fine des motifs locaux, comme l'a montré Raminedi et al. (2024) dans l'analyse d'images médicales [22].

Cette capacité est essentielle pour détecter des altérations subtiles, telles que des modifications de texture dans des images falsifiées.

Dans notre étude, nous évaluerons son efficacité à identifier des manipulations dans

un contexte judiciaire.

1.3.7 EfficientNetV2-S

— **Présentation** : EfficientNetV2-S, introduit par Tan et Le (2021), est un réseau de neurones convolutifs (CNN) avancé, dérivé d'EfficientNet (2019) [27].

Conçu pour optimiser l'efficacité computationnelle tout en maintenant une haute précision, il surmonte les limitations de son prédécesseur, notamment les longs temps d'entraînement.

Ce modèle est particulièrement adapté aux environnements à ressources limitées.

— **Architecture et Conception** : EfficientNetV2-S améliore de manière optimisée EfficientNet en utilisant la recherche d'architecture neuronale (NAS).

- **Blocs de construction** : MBConv (Mobile Inverted Bottleneck Convolution) et Fused-MBConv, comme illustré dans la figure 1.8.
- **Caractéristiques distinctives** :
 - Scaling progressif (résolution, profondeur, largeur).
 - Normalisation stochastique de profondeur.
 - Mise à l'échelle composée équilibrée.
 - Taille d'entrée : 384×384 pixels.
- **Paramètres** : Environ 22 millions

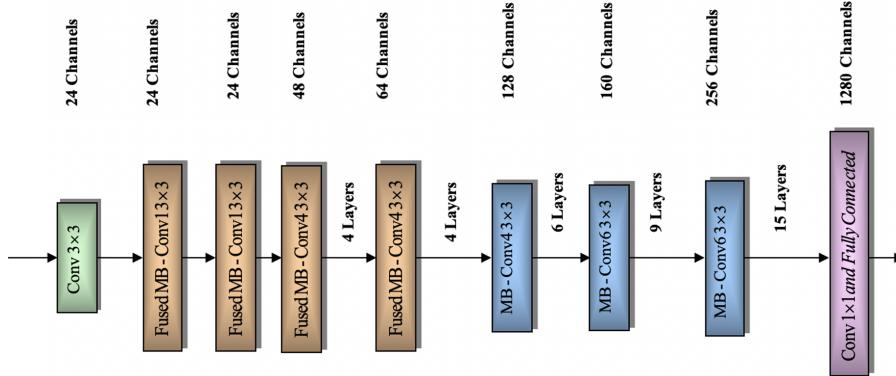


FIGURE 1.8 – Architecture du EfficientNetV2 [2]

— **Pertinence pour la Détection d'Images Falsifiées** : Dans le cadre de la détection d'images manipulées à des fins juridiques, EfficientNetV2-S excelle à identifier des motifs complexes et des anomalies indiquant une falsification.

Sa capacité à capturer des détails fins à différentes échelles en fait un choix idéal pour détecter des incohérences subtiles dans les images falsifiées, comme l'ont démontré AlTakrouri et al. (2023) en l'utilisant pour améliorer la résolution d'images via des réseaux antagonistes génératifs (GANs). [2]

La robustesse et la précision du modèle seront évaluées dans cette étude pour déterminer son efficacité à distinguer les images authentiques des images falsifiées.

1.3.8 ConvNeXt_Base

— **Présentation** : ConvNeXt_Base, introduit par Liu et al. en 2022, est une évolution moderne des réseaux de neurones convolutifs (CNN), visant à combler l'écart de performance entre les CNN traditionnelles et les Transformers [16].

En intégrant des concepts issus des Transformers tout en conservant l'efficacité des CNN, ce modèle offre une robustesse accrue pour des tâches de vision par ordinateur, comme la détection d'objets ou la classification d'images, dans des contextes variés.

— **Architecture et Conception** : ConvNeXt_Base modernise les CNN en adoptant des mécanismes inspirés des Transformers, tout en optimisant leur efficacité computationnelle

- **Blocs de construction** : Blocs ConvNeXt, composés de convolutions avec normalisation par couches et activation GELU, comme illustré dans la figure 1.9.
- **Caractéristiques distinctives** :
 - Attention locale pour capturer les dépendances spatiales.
 - Hiérarchie pyramidale pour une analyse multi-échelle.
 - Réduction des paramètres grâce à des couches simplifiées.
 - Taille d'entrée : 224×224 pixels.
- **Paramètres** : Environ 88 millions

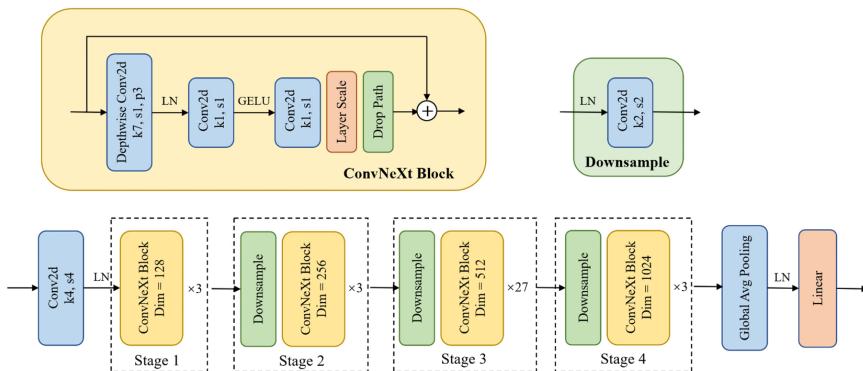


FIGURE 1.9 – Architecture du ConvNeXt [4]

— **Pertinence pour la Détection d'Images Falsifiées** : ConvNeXt_Base est particulièrement adapté à la détection des images falsifiées grâce à sa robustesse aux variations spatiales, comme démontré par Chen et al. (2023) dans l'extraction de bâtiments à partir d'images satellites [4].

Cette capacité permet d'identifier des artefacts discrets dans des contextes variés, un atout pour repérer des manipulations subtiles dans des images judiciaires.

Sa performance sera évaluée dans notre étude pour confirmer son efficacité dans la distinction entre images authentiques et falsifiées.

1.3.9 Comparaison analytique des architectures pour la détection de falsifications

- **CNNs vs. Transformers** : Les CNNs (EfficientNetV2, ConvNeXt, ResNetRS, Xception, MobileNetV3) excellent dans la capture des caractéristiques locales et des artefacts de manipulation, tandis que les Transformers (ViT, DeiT, BEiT) sont plus performants pour détecter les incohérences globales et contextuelles.
- **Impact de la profondeur et de la largeur** : Les modèles plus grands comme BEiT_Large offrent une capacité supérieure à modéliser des manipulations complexes, mais au prix d'exigences computationnelles accrues. Les modèles plus légers comme MobileNetV3 permettent un déploiement plus rapide mais avec une sensibilité potentiellement réduite.
- **Sensibilité multi-échelle** : Les architectures comme EfficientNetV2 avec son scaling progressif et ConvNeXt avec ses convolutions larges sont particulièrement adaptées pour détecter des manipulations à différentes échelles, des altérations subtiles de texture aux modifications structurelles importantes.

Ces modèles, disponibles via timm (<https://zenodo.org/records/7618837>), sont adaptés à la détection des falsifications grâce à leurs architectures variées, comme le montrent leurs performances dans la littérature.

1.4 Explicabilité de l'Intelligence Artificielle (XAI)

L'explicabilité de l'intelligence artificielle (XAI, pour Explainable Artificial Intelligence) est devenue un domaine essentiel, notamment dans des applications critiques comme la détection d'images falsifiées dans un cadre judiciaire. Alors que les modèles d'apprentissage profond, tels que ceux explorés dans les sections précédentes, offrent des performances impressionnantes, leur nature de "boîte noire" pose des défis en termes de confiance, de responsabilité et de validation légale [24], dont la nécessité de fournir des explications compréhensibles.

Plusieurs approches ont été développées pour rendre les modèles d'IA plus explicables. Parmi elles, on trouve :

- **SHAP (SHapley Additive exPlanations)** : Cette méthode, basée sur la théorie des jeux, attribue une importance à chaque caractéristique contribuant à une prédiction. Une étude de Lundberg et Lee (2017) montre que SHAP permet d'identifier les pixels influençant la détection de falsifications dans une

- image, offrant ainsi une visualisation claire aux experts judiciaires [17].
- **LIME (Local Interpretable Model-agnostic Explanations)** : Proposé par Ribeiro et al. (2016), LIME approxime localement un modèle complexe par un modèle interprétable, comme une décision d'arbre. Cela est particulièrement utile pour expliquer pourquoi une image est classée comme falsifiée, en soulignant les zones suspectes [23].
 - **Grad-CAM (Gradient-weighted Class Activation Mapping)** : Cette technique, développée par Selvaraju et al. (2017), utilise les gradients des couches convolutionnelles pour générer des cartes de chaleur montrant les régions d'une image influençant une décision. Par exemple, Grad-CAM a été appliqué avec succès pour localiser les artefacts de manipulation dans des deepfakes [25].

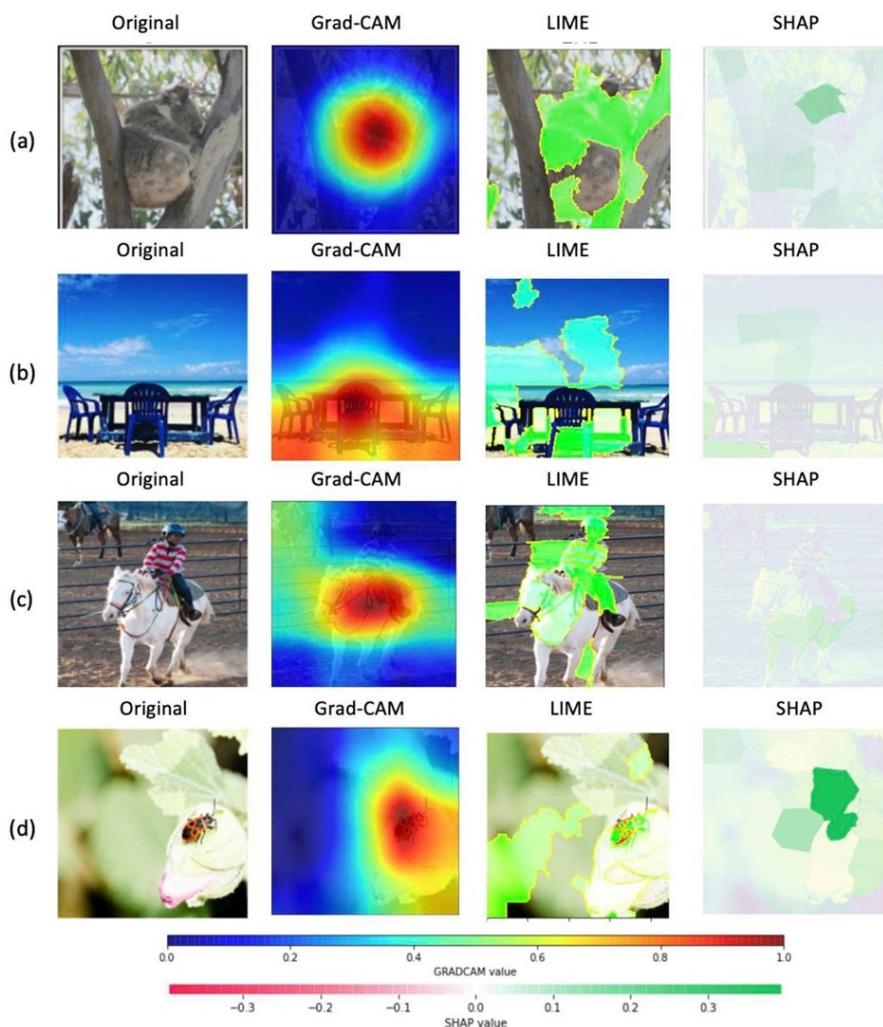


FIGURE 1.10 – méthodes XAI (Grad-CAM, LIME, and SHAP) avec quatre différentes images. [20, 7]

Ces outils permettent non seulement de valider les prédictions comme illustré dans la figure 1.10, mais aussi de former les utilisateurs à interpréter les résultats.

1.5 Architectures microservices

Les architectures microservices offrent des avantages significatifs pour la modularité et la sécurité dans les systèmes modernes. Elles décomposent les applications en services indépendants, chacun dédié à une tâche spécifique. Fowler et Lewis (2014) expliquent que "les microservices permettent une évolutivité ciblée et une résilience en isolant les pannes" [11]. Dragoni et al. (2017) ajoutent que "cette approche améliore la flexibilité et réduit les temps d'arrêt par rapport aux architectures monolithiques" [9].

Dans le cadre de la détection des images falsifiées, les microservices séparent des fonctions comme l'upload, l'analyse et la génération de rapports, renforçant la sécurité via des protocoles comme HTTPS et le chiffrement. Newman (2021) précise que "l'intégration de conteneurs (Docker) et de files d'attente (Kafka) garantit une communication efficace et une traçabilité essentielle pour les applications critiques" [18].

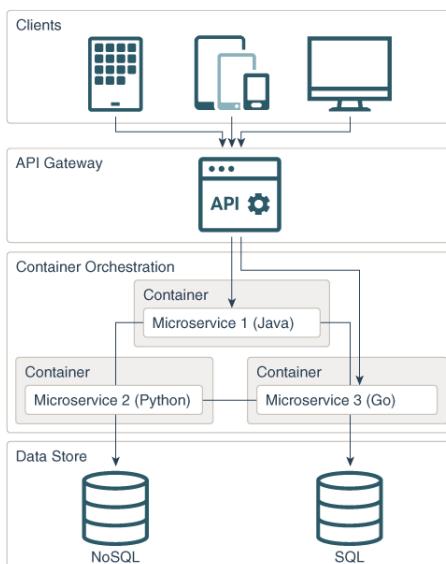


FIGURE 1.11 – architecture microservices [19]

Conclusion

Dans ce chapitre, nous avons exploré les diverses techniques de falsification d'images, les approches de détection, ainsi que huit architectures de modèles de Transfer Learning potentiellement efficaces pour l'identification des manipulations. Nous avons également examiné les architectures microservices adaptées au contexte judiciaire, soulignant l'importance de l'explicabilité dans un cadre légal. Ces analyses posent les bases théoriques pour le développement de notre solution, détaillé dans les chapitres suivants.

Chapitre 2

Analyse et conception

Introduction

Nous consacrons ce chapitre à l'analyse des besoins et à la conception du système de détection d'images falsifiées, spécifiquement adapté aux exigences judiciaires. Nous identifions les acteurs, détaillons les exigences fonctionnelles et non fonctionnelles, modélisons les interactions à l'aide de diagrammes, et définissons une conception architecturale.

2.1 Analyse des besoins

2.1.1 Identification des acteurs

La plateforme web est conçue pour être utilisée par plusieurs acteurs clés du système judiciaire, chacun exploitant des fonctionnalités spécifiques pour répondre à ses besoins dans le cadre de procédures légales. Le tableau suivant (Table 2.1) résume les acteurs et leurs responsabilités :

Acteurs	Description
Administrateur	Responsable de la gestion technique de la plateforme, l'administrateur gère les comptes des utilisateurs, surveille les performances du système pour évaluer l'efficacité de la plateforme, et accède aux journaux d'activité pour résoudre d'éventuels problèmes techniques ou assurer la traçabilité.
Détective	Chargé des investigations initiales, le détective utilise la plateforme pour créer des affaires judiciaires et soumettre des images suspectes. Il peut également analyser ces images pour détecter des falsifications et gérer les images uploadées, par exemple en les supprimant ou en les modifiant si nécessaire.
Avocat	Représentant les intérêts d'une partie dans une affaire, l'avocat utilise la plateforme pour consulter les détails des affaires judiciaires et assigner un expert judiciaire pour une analyse approfondie. Il peut également gérer les documents légaux associés, en les consultant et en les annotant pour appuyer sa plaidoirie, et télécharger les rapports finaux pour une utilisation en audience.
Expert Judiciaire	Mandaté pour fournir une expertise technique, l'expert judiciaire consulte les affaires qui lui sont assignées et accède à leur historique pour contextualiser son analyse. Il annote les rapports avec des observations détaillées pour les rendre exploitables par le juge ou l'avocat. Enfin, il peut télécharger les rapports finaux pour un usage hors ligne.

Juge	En tant qu'autorité décisionnelle, le juge utilise la plateforme pour consulter les rapports d'analyse et accéder à l'historique des analyses pour retracer le processus décisionnel. Il peut comparer plusieurs rapports pour une même affaire afin d'évaluer les divergences ou les consistances, et rendre un jugement éclairé basé sur les preuves visuelles analysées.
------	---

TABLE 2.1 – Liste des acteurs

2.1.2 Besoins fonctionnels

Nous identifions les besoins fonctionnels principaux que l'application doit offrir, détaillés dans le Tableau 2.2 ci-dessous.

	Besoins fonctionnels	Description
1	S'authentifier au système	Processus d'accès sécurisé permettant à tous les acteurs (DéTECTIVE, Avocat, Expert Judiciaire, Juge, Administrateur) de se connecter via une authentification renforcée à deux facteurs (2FA).
2	Gérer les utilisateurs	Gestion des comptes utilisateurs par l'Administrateur, incluant la création, la modification et la suppression des profils. Ce besoin est une extension de la surveillance de la plateforme, permettant un contrôle granulaire des accès.
3	Surveiller et maintenir la plateforme	Supervision des performances et des utilisateurs, confiée à l'Administrateur, qui contrôle les comptes, surveille les indicateurs clés de performance (KPI), et suit les journaux d'activité.
4	Affecter expert à une affaire	L'Avocat assigne un Expert Judiciaire à une affaire pour une analyse approfondie, après authentification, afin de garantir que l'expertise est réalisée par une personne qualifiée.
5	Gérer les documents légaux	L'Avocat consulte, annote et télécharge les documents légaux associés à une affaire, après authentification, pour préparer sa plaidoirie ou documenter les procédures.

6	Administrer les affaires judiciaires	Gestion complète des affaires judiciaires, impliquant principalement le DéTECTIVE pour la création initiale des affaires et leur suivi, ainsi que l'AVOCAT et le JUGE pour la consultation des détails.
7	Gérer les preuves visuelles	Traitements global des images suspectes, impliquant le DéTECTIVE pour leur soumission, l'EXPERT JUDICIAIRE pour leur analyse et annotation, et l'AVOCAT pour leur validation dans un contexte de défense. Ce besoin inclut une authentification pour garantir la sécurité des données.
8	Analyser les preuves	Le DéTECTIVE utilise les outils de la plateforme pour analyser les preuves visuelles et détecter les falsifications. Ce processus nécessite une authentification et s'appuie sur les preuves préalablement soumises.
9	Produire et gérer les rapports	Élaboration, personnalisation, et distribution des rapports d'analyse, principalement par l'EXPERT JUDICIAIRE pour la génération et l'annotation, le JUGE pour la consultation dans la prise de décision, et l'AVOCAT pour leur utilisation en audience.
10	Evaluer et finaliser les décisions judiciaires	Le JUGE consulte les rapports et documents légaux pour évaluer les preuves et finaliser les affaires judiciaires en rendant un jugement éclairé. Ce processus nécessite une authentification.

TABLE 2.2 – Besoins fonctionnels

Ces besoins sont modélisés dans un diagramme de cas d'utilisation (figure 2.1) dans la section suivante.

2.1.3 Besoins non fonctionnels

Par la suite, nous avons identifié les exigences non fonctionnelles auxquelles le système doit répondre, listées dans le tableau 2.3 ci-dessous :

	Besoins non fonctionnels	Description
1	Sécurité	Les données (images, rapports) doivent être chiffrées avec SHA-256, et les communications sécurisées via HTTPS.
2	Traçabilité	Chaque événement (soumission, analyse, génération de rapport) doit être journalisé avec un identifiant unique, un horodatage, et les détails de l'opération, pour assurer une transparence légale .
3	Performance	L'analyse complète d'une image (de la soumission au rapport) doit être effectuée en moins de 5 secondes pour garantir une expérience utilisateur fluide, une exigence implicite dans un contexte judiciaire.
4	Évolutivité	Le système doit gérer des requêtes simultanées, avec une architecture capable de s'adapter à une charge croissante.
5	Modularité	Chaque composant doit être indépendant pour permettre des mises à jour ou des maintenances sans affecter les autres.

TABLE 2.3 – Besoins non fonctionnels

2.1.4 Contraintes judiciaires

En plus des exigences standards, L'application doit respecter les contraintes suivantes imposées par le contexte judiciaire :

- **Confidentialité** : Les données (images, rapports) doivent être protégées conformément aux lois sur la protection des données , avec un accès restreint aux acteurs autorisés.
- **Tracabilité Légale** : Toutes les analyses et modifications doivent être enregistrées avec des horodatages et des identifiants d'utilisateur pour garantir leur admissibilité en justice.
- **Auditabilité** : Les journaux doivent permettre un audit complet des opérations, avec une rétention des données d'au moins 5 ans pour répondre aux exigences légales potentielles.
- **Preuve irréfutable** : Les résultats d'analyse doivent être accompagnés de justificatifs techniques (ex. : visualisations XAI) pour être acceptés comme preuves en cour.

2.1.5 Diagrammes de l'analyse

Cette partie d'analyse s'appuie sur la création de diagrammes qui permettent de structurer et de formaliser les besoins du système. Nous avons utilisé :

- Le diagramme de cas d'utilisation
- Les diagrammes structurels : Dans notre cas, nous avons utilisé le diagramme de classes (Figure 2.2)
- Les diagrammes comportementaux : Nous avons élaboré des diagrammes de séquence (Figures 2.3, 2.4 et 2.5), ainsi que le diagramme d'activité (Figure 2.6).

1. Diagramme de cas d'utilisation

Nous présentons ci-dessous le diagramme de cas d'utilisation (Figure 2.1) qui modélise les interactions principales entre les acteurs (DéTECTIVE, Avocat, Expert Judiciaire, Juge, Administrateur) et leur utilisation active du système.

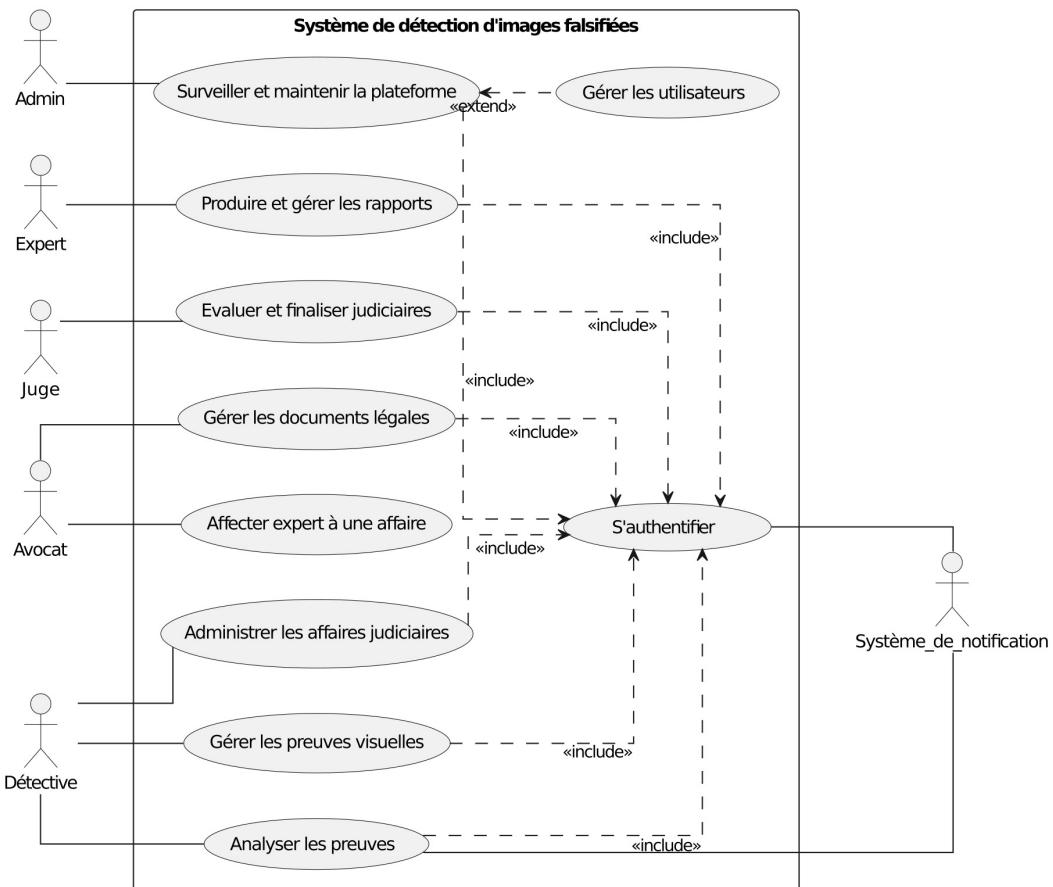


FIGURE 2.1 – Diagramme de cas d'utilisation

2. Diagramme de Classes

Nous présentons ici le diagramme de classes pour modéliser les entités principales de notre système et leurs relations (figure 2.2).

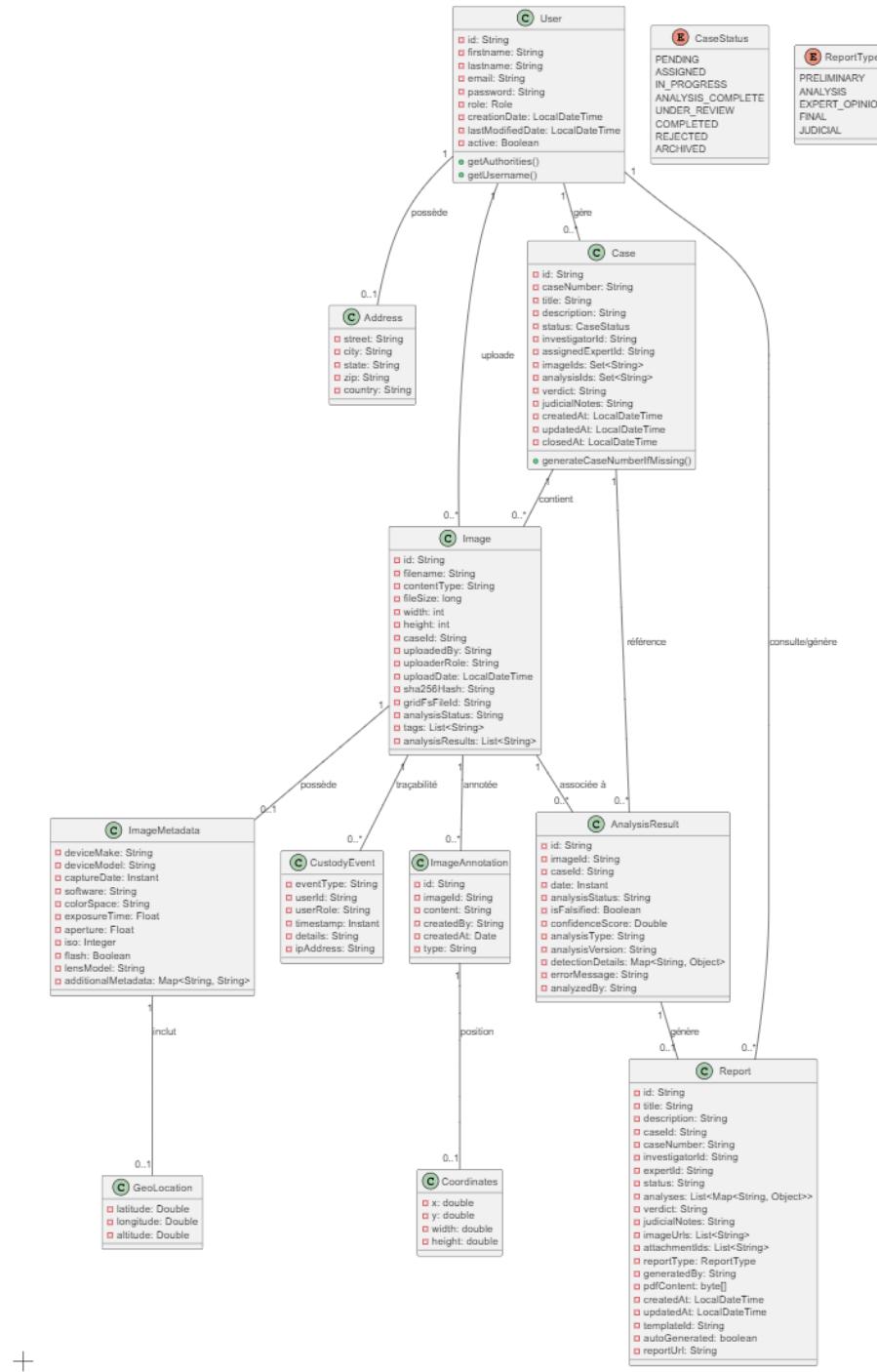


FIGURE 2.2 – Diagramme de classes

Ce diagramme de classes (Figure 2.2) illustre la structure du système. Au cœur du système, un Utilisateur (User) gère des Affaires (Case), chacune possédant un statut (CaseStatus) et contenant une ou plusieurs Images. L'(Image) elle-même est une entité centrale, détaillée par des Métadonnées (ImageMetadata). Le processus d'analyse d'une Image aboutit à un Résultat d'Analyse (AnalysisResult), lequel alimente la création d'un Rapport (Report). Ce Rapport, caractérisé par un Type de Rapport (ReportType), est ensuite associé à l'Affaire correspondante.

3. Diagramme de séquences

Dans cette partie, nous présentons les diagrammes de séquence pour deux cas d'utilisation sélectionnés parmi ceux identifiés dans la section 2.1.2 : **S'authentifier au système** et **Produire et gérer les rapports**. Ces diagrammes détaillent les interactions entre les acteurs et les services du système.

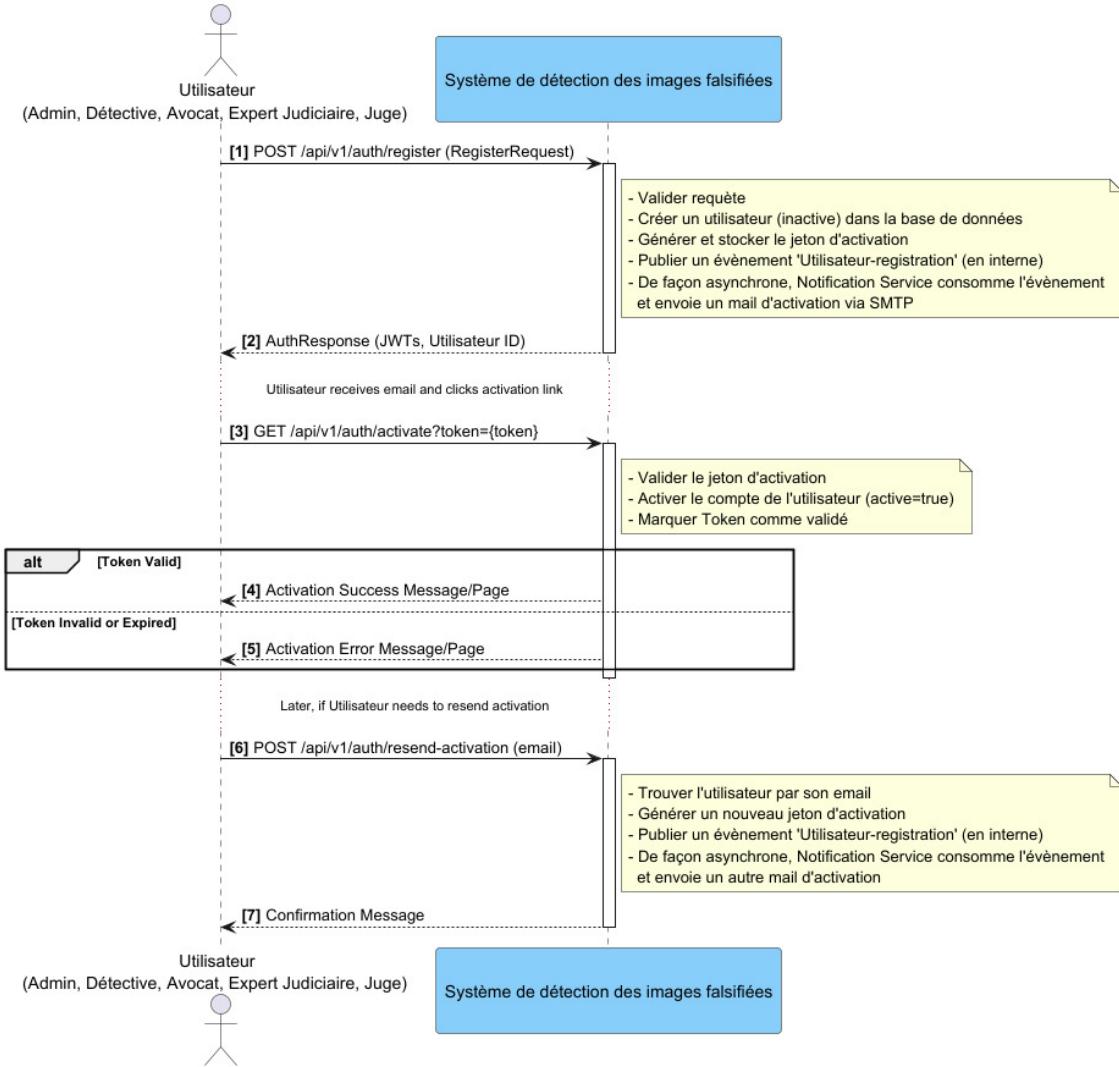


FIGURE 2.3 – Diagramme de séquence : S'authentifier au système

Ce diagramme (figure 2.3) illustre le processus d'authentification et d'activation et de réactivation d'un compte utilisateur dans le système de détection des images falsifiées. L'Utilisateur initie une requête d'enregistrement via une API (POST `/api/v1/auth/register`), ce qui déclenche la validation de la requête, la création du compte, et l'envoi d'un jeton d'activation par email via le Notification Service. L'Utilisateur active ensuite son compte en cliquant sur le lien fourni (GET `/api/v1/auth/activate`), et le système confirme l'activation ou affiche une erreur si le jeton est invalide ou expiré. Enfin, si nécessaire, l'Utilisateur peut demander une

réactivation (POST /api/v1/auth/resend-activation), générant un nouveau jeton d'activation envoyé par email.

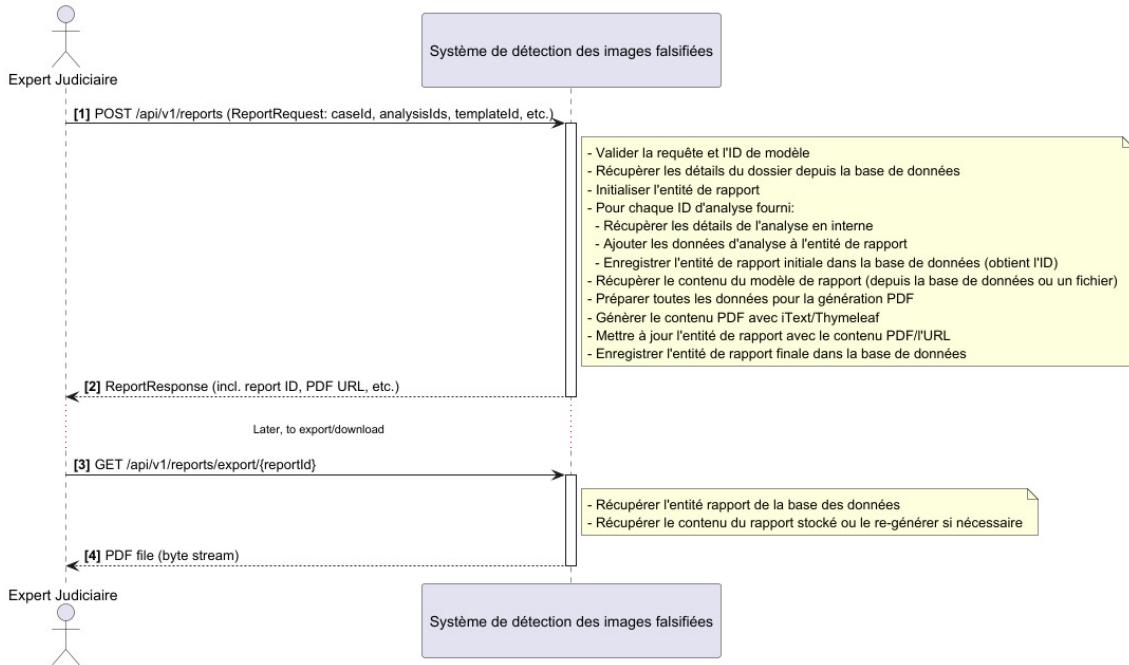


FIGURE 2.4 – Diagramme de séquence : Produire et gérer les rapports

Ce diagramme (figure 2.4) illustre le processus de génération et de récupération de rapports dans le système de détection des images falsifiées. L'Utilisateur initie une requête de création de rapport via une API (POST /api/v1/reports), en fournissant des données telles que l'ID de l'affaire judiciaire, les IDs des analyses et le modèle de rapport. Le système valide la requête, récupère les informations nécessaires depuis la base de données, effectue une analyse intermédiaire si requis, génère un rapport au format PDF avec Thymeleaf, et enregistre le tout, renvoyant un ID de rapport et une URL PDF. Plus tard, l'Utilisateur peut récupérer ce rapport via une requête (GET /api/v1/reports/export), et le système retourne le contenu du rapport sous forme de flux de bytes, en le re-générant si nécessaire.

Pour offrir une vue plus approfondie, ce diagramme de séquence peut être enrichi et transformé en une version plus détaillée, intégrant des interactions supplémentaires et des étapes intermédiaires.

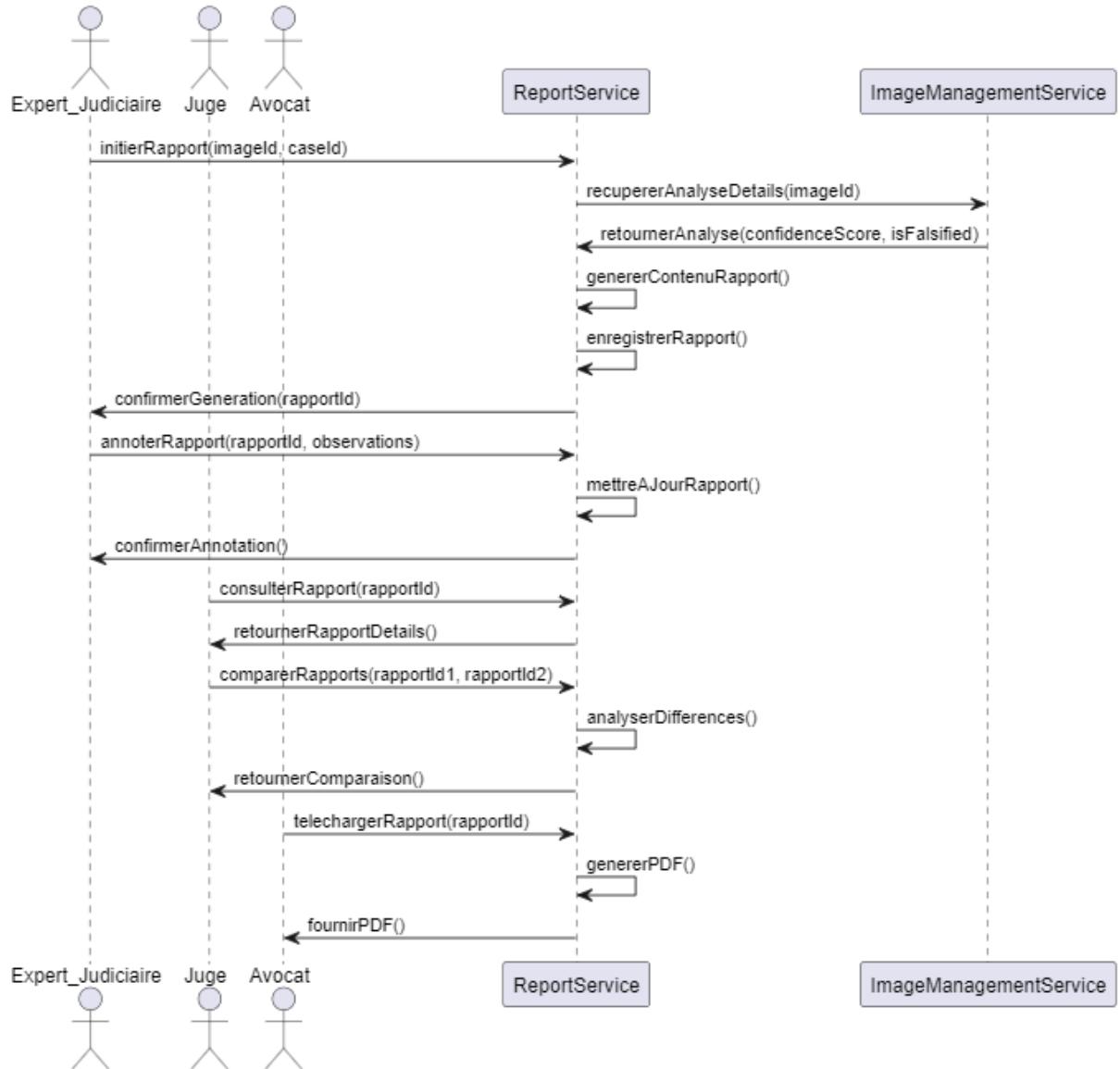


FIGURE 2.5 – Diagramme de séquence détaillé : Produire et gérer les rapports

Ce diagramme (figure 2.5) montre le processus de production et de gestion des rapports, impliquant l’Expert Judiciaire qui génère et注释 un rapport, le Juge qui consulte et compare les rapports, et l’Avocat qui télécharge le rapport final. Le service reportservice est utilisé pour générer et gérer les rapports.

4. Diagramme d’activité

Nous présentons ici (figure 2.6) le diagramme d’activité pour le cas d’utilisation **Gérer les preuves visuelles**. Ce diagramme modélise les flux de travail impliquant les acteurs.

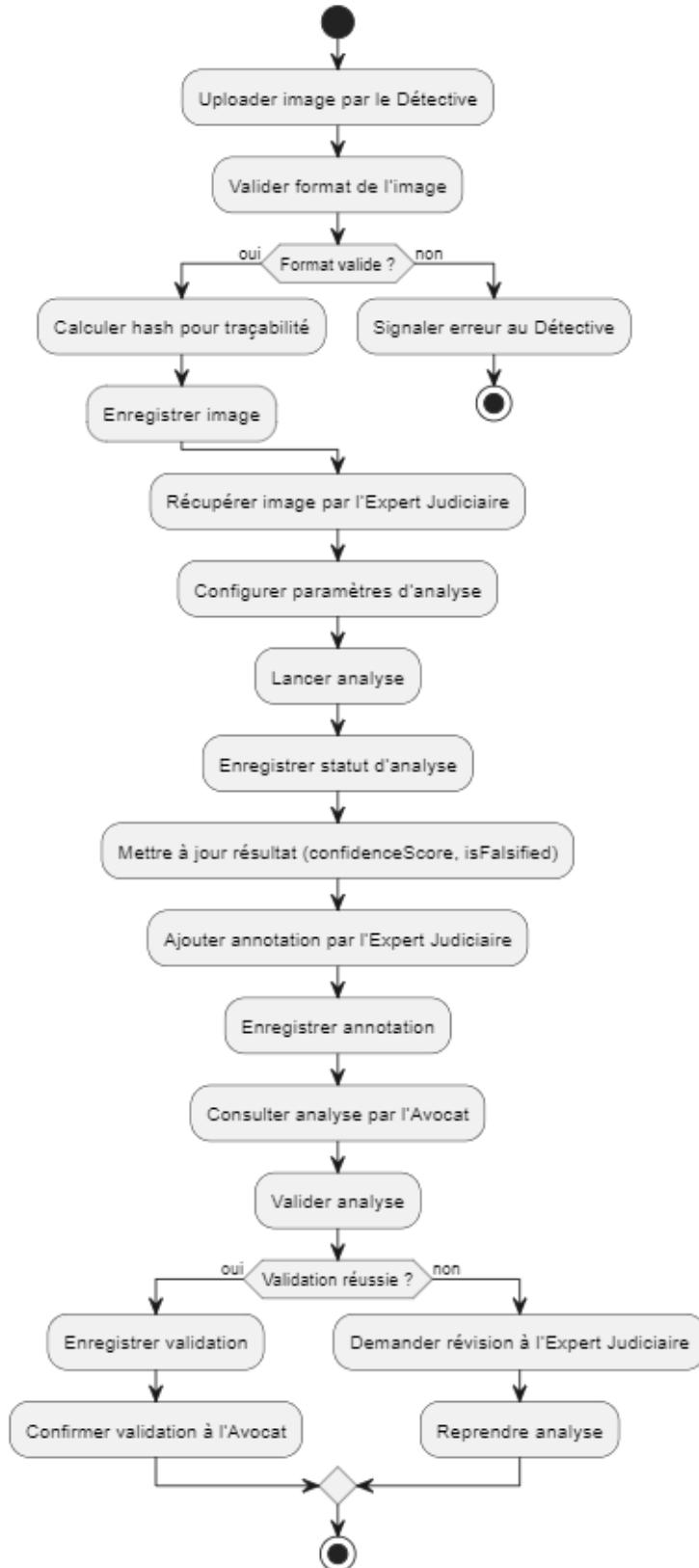


FIGURE 2.6 – Diagramme d'activité : Gérer les preuves visuelles

Le flux de diagramme commence par l'authentification du Détective, suivie de l'upload d'une image. L'imageAnalysis-service traite l'image, l'Expert Judi-

ciaire examine les résultats et ajoute des annotations, et l'Avocat valide le rapport final. Des points de décision assurent que seules les images valides sont analysées.

2.2 Conception de l'architecture

2.2.1 Vue d'ensemble

Nous avons opté pour une architecture microservices afin d'assurer modularité, scalabilité, et résilience, tout en répondant aux besoins de sécurité et de traçabilité. Voici les services constituant l'architecture :

- **config-server** : Gère les configurations centralisées des autres services, garantissant une gestion cohérente des paramètres.
- **discovery-service** : Permet la découverte et l'enregistrement dynamiques des services, facilitant la communication dans une architecture distribuée.
- **gateway-service** : Centralise les requêtes des utilisateurs, gère l'authentification via 2FA, et distribue les requêtes aux services appropriés, avec un Load Balancer intégré pour optimiser les performances.
- **user-service** : Gère l'authentification des utilisateurs (Déetective, Expert Judiciaire, Avocat, Juge, Administrateur) et leurs rôles.
- **imageManagement-service** : Responsable de la gestion des images, incluant leur stockage, récupération, et annotation.
- **imageAnalysis-service** : Effectue l'analyse des images pour détecter les falsifications, en utilisant des modèles d'intelligence artificielle.
- **report-service** : Gère la génération, l'annotation, et la consultation des rapports d'analyse.
- **notification-service** : Envoie des notifications aux utilisateurs (par exemple, confirmation d'upload ou validation d'analyse).

2.2.2 Diagramme de composants

Le diagramme de composant (Figure 2.7) illustre les interactions entre les services. Le API Gateway agit comme point d'entrée, routant les requêtes vers les services appropriés comme Report Service (génération de rapports), Image Analysis Service (analyse des images), Image Management Service (gestion des images), et Notification Service (alertes).

Ces services communiquent via Kafka pour les flux asynchrones (e.g., analysis-completed, fetchImage) et accèdent à PostgreSQL (utilisateurs) et MongoDB (images, rapports). Les services Discovery & Config Service assurent la configuration et la découverte des composants, garantissant une architecture scalable et robuste.

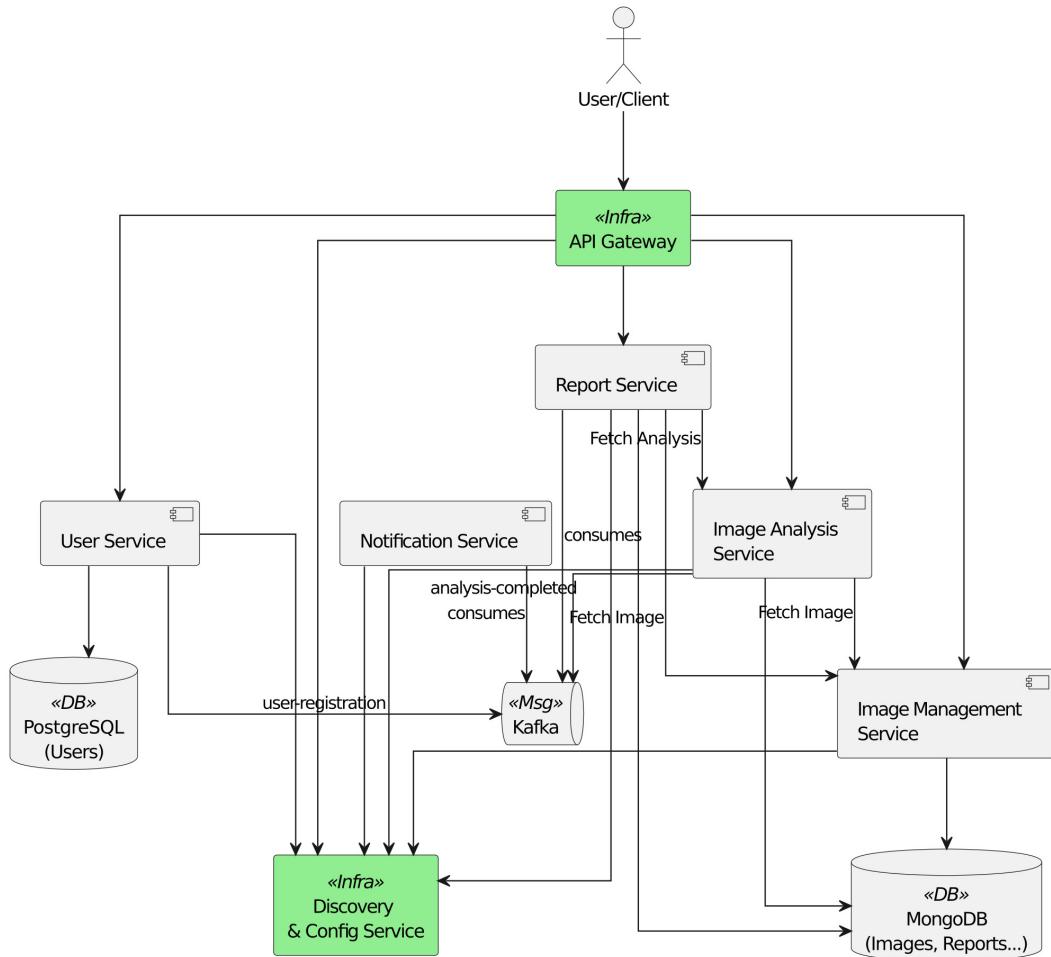


FIGURE 2.7 – Diagramme de composants

Conclusion

Ce chapitre a défini les besoins fonctionnels et non fonctionnels de la plateforme, identifié les acteurs, et conçu une architecture microservices modélisée par des diagrammes UML. Cette analyse pose les bases pour la méthodologie détaillée dans le chapitre suivant.

Chapitre 3

Méthodologie

Introduction

Dans ce chapitre, nous détaillons la méthodologie que nous avons adoptée pour concevoir et mettre en œuvre notre système de détection d'images falsifiées dans un contexte judiciaire.

3.1 Présentation générale

Notre démarche est structurée autour le développement de modèles de Transfer Learning, tout en intégrant des techniques d'explicabilité (XAI) pour analyser les images, qui seront utiliser lors de la création d'une infrastructure web répondant aux besoins judiciaires.

Cette méthodologie se décompose en plusieurs étapes clés, illustrées comme suit :

- **Collecte et préparation des données** : Constitution d'un ensemble de données adapté à notre cas d'étude, avec un prétraitement pour garantir la qualité des données.
- **Développement des modèles** : Sélection, adaptation et entraînement de modèles de Transfer Learning pour la classification binaire (authentique vs falsifié).
- **Évaluation des performances** : Analyse des résultats à l'aide de métriques standardisées pour sélectionner le modèle le plus performant.
- **Intégration de l'explicabilité** : Mise en place de méthodes XAI pour répondre aux exigences judiciaires de transparence et d'interprétabilité.

L'objectif est de combiner précision analytique et utilité pratique, en répondant aux exigences d'un contexte judiciaire où la fiabilité, la traçabilité et la rapidité sont essentielles.

3.2 Collecte et préparation des données

3.2.1 Collecte des données

Pour entraîner et tester nos modèles, nous avons sélectionné un ensemble de données disponible sur Kaggle, intitulé "Handwritten Signature Verification" (<https://www.kaggle.com/datasets/tienend/handwritten-signature-verification>). comprend 6,172 images de signatures manuscrites, réparties équitablement entre 3,188 signatures authentiques et 2,088 signatures falsifiées.

Il constitue une base idéale pour notre cas d'étude, car les signatures sont un type de preuve couramment analysé dans les procédures judiciaires.

3.2.2 Préparation des Données

Le processus de préparation des données inclut plusieurs étapes visant à garantir leur qualité et leur compatibilité avec les modèles :

1. **Organisation des données** : Nous avons divisé l'ensemble de données en trois sous-ensembles selon une répartition standard :
 - 70 % pour l'entraînement, afin de permettre aux modèles d'apprendre les caractéristiques des signatures.
 - 15 % pour la validation, utilisé pour ajuster les hyperparamètres pendant l'entraînement.
 - 15 % pour les tests, réservé pour évaluer les performances finalesNous avons vérifié l'équilibre entre les classes (authentique et falsifié) pour éviter les biais lors de l'entraînement. (*Section 6 du notebook [6]*)
2. **Vérification de l'intégrité** : Nous avons inspecté les images pour détecter d'éventuelles anomalies (ex. : fichiers corrompus, erreurs d'étiquetage) et assurer une cohérence entre les classes.
3. **Prétraitement des images** : À l'aide de la bibliothèque de prétraitement spécialisée (Albumentations), nous avons appliqué une série de transformations pour standardiser les images et enrichir les données :
 - **Redimensionnement** : Ajustement des images à des tailles compatibles avec les modèles (ex. : 224x224 pixels pour la majorité, adaptées si nécessaire).
 - **Normalisation** : Application des statistiques standards d'ImageNet (moyenne [0.485, 0.456, 0.406], écart-type [0.229, 0.224, 0.225]) pour uniformiser les valeurs des pixels.
 - **Augmentations** : Introduction de modifications comme des rotations aléatoires ($\pm 10^\circ$), des inversions horizontales, des ajustements de contraste ($\pm 15\%$) et des translations ($\pm 5\%$) pour simuler des variations naturelles et améliorer la généralisation des modèles.Ces transformations garantissent que les données sont adaptées aux exigences des modèles tout en réduisant les risques de surapprentissage.

(*Section 8 du notebook [6]*)

3.3 Étude et développement des modèles

Nous avons opté pour une approche de Transfer Learning, qui consiste à utiliser des modèles pré-entraînés sur de vastes ensembles de données génériques (comme ImageNet), puis à les adapter à notre tâche spécifique de classification binaire (authentique vs falsifié).

3.3.1 Sélection des modèles

Nous avons considéré un ensemble diversifié de modèles, disponibles via la bibliothèque timm, pour explorer une variété d'architectures et d'approches (CNNs et Transformers). Le choix de ces modèles repose sur leurs performances reconnues et leur adaptabilité à la détection des falsifications, comme détaillé dans l'état de l'art (Chapitre 1).

Le tableau suivant (Table 3.1) résume leurs caractéristiques principales.

TABLE 3.1 – Aperçu des modèles étudiés

Modèle	Type	Taille d'entrée	Paramètres (M)	Avantage principal
ResNetRS50	CNN	224x224	25	Stabilité dans l'extraction des caractéristiques
Xception	CNN	299x299	23	Capture multi-échelle des détails
MobileNetV3_Large	CNN	224x224	5,5	Efficacité computationnelle
ViT_Base	Transformer	224x224	86	Modélisation contextuelle
DeiT_Base	Transformer	224x224	86	Capture des relations globales
BEiT_Base	Transformer	224x224	307	Analyse fine des motifs locaux
EfficientNetV2-S	CNN	384x384	22	Scaling multi-échelle
ConvNeXt_Base	CNN	224x224	88	Robustesse aux variations spatiales

3.3.2 Adaptation des modèles

Pour chaque modèle, nous avons :

- Conservé les couches convolutives ou Transformer pré-entraînées pour extraire des caractéristiques générales.
- Remplacé la couche de classification finale par une nouvelle couche adaptée à la classification binaire.
- Gelé partiellement les premières couches pour préserver les connaissances acquises, tout en entraînant les couches supérieures sur nos données.

Cette approche, reposant sur une implémentation en PyTorch, nous permet de tirer parti des capacités des modèles pré-entraînés tout en les spécialisant pour la détection des signatures falsifiées.

3.3.3 Entraînement

L’entraînement des modèles a été conçu pour optimiser leurs performances tout en minimisant les risques de surapprentissage. Nous avons structuré ce processus en deux phases principales : une recherche automatisée des hyperparamètres et un entraînement final avec les configurations optimales.

1. Optimisation des hyperparamètres :

Pour chaque modèle, une recherche systématique des hyperparamètres sera réalisée. Nous avons employé la bibliothèque Optuna pour y faire tout en réalisant 20 essais (trials) par modèle et en explorant :

- Le **taux d’apprentissage** Exploration dans une plage de 10^{-3} à 5×10^{-6} , avec une décroissance exponentielle pour affiner la convergence.
- Le **type d’optimiseur** Comparaison entre Adam (adaptatif) et AdamW (avec découplage de la régularisation de poids) pour optimiser la stabilité.
- La **régularisation** : Test de dropout (valeurs de 0,1 à 0,6 avec pas égal à 0,05) pour prévenir le surapprentissage.
- La **taille des lots** (batch size, ex. : 16, 32, 64).
- Le **nombre d’Époques** : Détermination d’une plage initiale (10 à 30 époques) avec un critère d’arrêt précoce basé sur la perte de validation.

Cette optimisation sera effectuée sur l’ensemble de validation pour identifier les configurations optimales pour chaque modèle.

2. Entraînement final :

- À partir des hyperparamètres optimaux identifiés, nous avons entraîné chaque modèle sur l’ensemble d’entraînement complet.
- Une patience étendue à 5 époques a été appliquée pour stabiliser les performances.
- La fonction de perte utilisée est BCEWithLogitsLoss, adaptée à la classification binaire, combinant une activation sigmoïde et une perte d’entropie croisée.
- L’entraînement a été effectué sur un dispositif accéléré (GPU si disponible) pour réduire les temps de calcul.
- Les meilleurs checkpoints (basés sur l’accuracy de validation) ont été sauvegardés pour une réutilisation ultérieure.

Nous avons également surveillé les courbes de perte et de précision pendant l’entraînement pour détecter d’éventuels problèmes comme le surapprentissage ou une convergence insuffisante.

(Section 13 du notebook [6])

3.3.4 Comparaison des modèles

Une fois les hyperparamètres optimisés, une comparaison systématique sera menée :

- Évaluation des performances sur l'ensemble de test avec des métriques standard (détaillées dans la section 3.4).
- Analyse des forces et faiblesses de chaque modèle.
- Sélection des modèles les plus performants pour l'intégration finale dans le système.

3.4 Évaluation des performances

Nous avons réalisé l'évaluation des modèles sur l'ensemble de test pour mesurer leur capacité à généraliser sur des données non vues. Nous avons utilisé un ensemble de métriques adaptées à la classification binaire, calculées via des scripts Python.

(Section 14 du notebook [6])

TABLE 3.2 – Métriques quantitatives

Métriques	Description	Équation
Accuracy	Pourcentage de prédictions correctes, reflétant la performance globale.	$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$
F1-score	Moyenne harmonique de la précision et du rappel, utile pour équilibrer les classes.	$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
Précision	Proportion de prédictions positives correctes parmi toutes les prédictions positives.	$\text{Precision} = \frac{TP}{TP + FP}$
Rappel	Proportion d'images falsifiées correctement identifiées parmi toutes les images falsifiées.	$\text{Recall} = \frac{TP}{TP + FN}$

Avec **TP** , **FP** , **TN** , and **FN** sont les éléments de la matrice de confusion (Figure 3.1), représentant respectivement les prédictions vrais positives, faux positives, vrais négatives, and faux négatives.

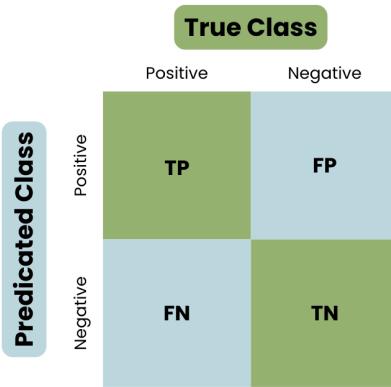


FIGURE 3.1 – Matrice de confusion

Toutes ces résultats seront exportés sous forme de :

- Fichiers JSON pour chaque modèle, contenant les métriques et les hyperparamètres associés.
- Des courbes de perte et de précision (entraînement et validation) sont tracées pour évaluer la convergence.
- Une archive (zip) regroupant les checkpoints et les résultats est créée pour une intégration future (*Section 15 du notebook [6]*) .

Cette évaluation rigoureuse nous permet de comparer les modèles et de sélectionner les plus adaptés pour le déploiement dans la plateforme web.

3.5 Intégration de l'explicabilité

3.5.1 Méthodes sélectionnées

Nous avons sélectionné trois méthodes d'explicabilité (XAI) adaptées à notre cas d'étude, en nous appuyant sur l'état de l'art (section 1.4). Ces méthodes permettent de visualiser et d'expliquer les décisions des modèles, en mettant en évidence les régions ou caractéristiques influençant la classification d'une image comme authentique ou falsifiée :

- **Grad-CAM** : Cette méthode utilise les gradients des couches convolutives du modèle pour générer des cartes de chaleur (heatmaps). Ces cartes mettent en évidence les régions de l'image, comme les discontinuités ou variations d'épaisseur dans une signature, qui ont le plus influencé la prédiction.
- **SHAP** : Basée sur la théorie des jeux, SHAP attribue une importance quantitative à chaque pixel ou groupe de pixels dans l'image, expliquant leur contribution à la décision finale. Par exemple, SHAP peut indiquer qu'un changement soudain de texture dans une signature a fortement contribué à la classification "falsifiée".

- **LIME** : LIME approxime localement le modèle complexe par un modèle interprétable pour expliquer les prédictions. Dans notre cas, LIME segmente l'image en superpixels et identifie ceux qui influencent le plus la décision, comme des traits incohérents dans une signature.

Ces trois méthodes ont été choisies pour leur complémentarité : Grad-CAM offre une visualisation spatiale, SHAP une analyse quantitative, et LIME une interprétation simplifiée, répondant ainsi aux besoins variés des acteurs judiciaires.

3.5.2 Objectifs

L'intégration de l'explicabilité dans notre système poursuit plusieurs objectifs, alignés avec les exigences judiciaires et les besoins fonctionnels (section 2.1.2).

- Fournir des visualisations claires (cartes de chaleur, graphiques de contribution) intégrées dans les rapports générés par le **report-service**.
- Assurer la transparence des décisions pour les Utilisateurs Judiciaires, en identifiant les zones suspectes.
- Répondre aux exigences légales, telles que le droit à l'explication.

Conclusion

Dans ce chapitre, nous avons énoncé la méthodologie adopté qui fournira un cadre structuré pour développer le système en étudiant les huit modèles avec une optimisation rigoureuse des hyperparamètres, en évaluant les performances de manière exhaustive, et en intégrant l'explicabilité. Les étapes décrites ici guideront l'implémentation dans le chapitre suivant.

Chapitre 4

Implémentation

Introduction

Dans ce chapitre, nous détaillons la mise en œuvre complète de notre solution, qui combine une approche d'intelligence artificielle pour détecter les images falsifiées avec une plateforme web basée sur une architecture microservices. Nous avons structuré l'implémentation en cinq axes principaux : le choix de l'environnement technique, le développement des modèles de Transfer Learning, la construction des microservices pour la partie web, l'intégration avec le déploiement final, et la conception des interfaces graphiques.

4.1 Choix de l'environnement technique

Nous avons sélectionné un ensemble d'outils pour couvrir à la fois les besoins de l'intelligence artificielle et de la plateforme web, en visant une synergie entre performance et facilité de maintenance. Voici les composants clés de notre environnement technique, avec leurs rôles spécifiques :

1. Frameworks et bibliothèques IA :

- **PyTorch** : Nous avons opté pour PyTorch comme framework principal pour le développement des modèles, en raison de sa flexibilité dans la gestion des graphes computationnels dynamiques et de son écosystème riche pour la vision par ordinateur. Cela nous a permis de personnaliser les couches des modèles et d'effectuer un fine-tuning précis.
- **timm (PyTorch Image Models)** : Cette bibliothèque nous a fourni un accès direct à une large gamme de modèles pré-entraînés, réduisant le temps nécessaire pour configurer des architectures complexes.
- **Albumentations** : Nous avons utilisé cette bibliothèque pour le prétraitement des images, appréciant sa rapidité et sa capacité à appliquer des transformations en temps réel, essentielles pour les augmentations de données.
- **Optuna** : Cet outil d'optimisation des hyperparamètres nous a permis d'automatiser la recherche des meilleures configurations, améliorant ainsi les performances des modèles.
- **tqdm** : Nous avons intégré cette bibliothèque pour suivre visuellement l'avancement des boucles d'entraînement, facilitant le débogage et la gestion des longues sessions.

2. Technologies Web (Backend - Microservices) :

- **Spring Boot** : Pour le backend, nous avons choisi Spring Boot pour développer nos microservices, en raison de sa robustesse, de son support pour les API REST, et de sa facilité d'intégration avec d'autres outils Java.

- **Spring Cloud (Config, Gateway, Netflix Eureka)** : Pour la gestion centralisée des configurations, la mise en place d'une API Gateway (routage, authentification, limitation de débit), et la découverte de services
- **PostgreSQL** : Ce système de base de données relationnelle a été sélectionné pour sa fiabilité et sa capacité à gérer des volumes importants de données structurées, comme les profils utilisateurs et les résultats d'analyse.
- **MongoDB (avec GridFS)** : Base de données NoSQL pour stocker les images (via GridFS pour les fichiers volumineux) et les résultats d'analyse (sorties XAI, métadonnées complexes).
- **Kafka** : Nous avons intégré Kafka pour gérer les notifications asynchrones entre microservices, assurant une communication rapide et fiable, même sous forte charge.
Par exemple, le `notification-service` utilise Kafka pour informer les utilisateurs de la disponibilité des rapports.
- **Maven/Gradle** : Outils de gestion de build et de dépendances pour les projets Java.

3. Développement Web (Frontend) :

- **Angular** : Framework JavaScript/TypeScript pour construire des interfaces utilisateurs modulaires, réactives et maintenables.
- **Bootstrap / Angular Material** : Pour les composants UI et le styling, assurant une expérience utilisateur cohérente et responsive.
- **Node.js / npm** : Environnement d'exécution et gestionnaire de paquets pour le développement frontend.

4. Outils de déploiement et infrastructure :

- **Docker** : Chaque composant, du modèle IA aux microservices, a été conteneurisé avec Docker pour garantir une exécution cohérente sur différents environnements.
- **Git / GitHub** : Pour le contrôle de version du code source et la collaboration.

5. Environnement matériel :

Les entraînements des modèles ont été réalisés sur des plateformes cloud offrant un accès GPU (NVIDIA Tesla T4 via Google Colab/Kaggle), avec 16 Go GDDR6 de VRAM, ce qui a permis de réduire significativement les temps de calcul par rapport à une exécution sur CPU.

4.2 Implémentation du Transfer Learning

Nous avons mis en œuvre les modèles de Transfer Learning conformément à la méthodologie définie au Chapitre 3. Cette implémentation s'est déroulée en plusieurs étapes, en

utilisant les outils décrits ci-dessus (section 4.1).

1. **Chargement des Modèles** : Nous avons utilisé la bibliothèque timm pour charger les modèles pré-entraînés étudiés (section 3.3.1), comme MobileNetV3_Large, ResNetRS50, et DeiT_Base.
2. **Préparation des Données** : Les données ont été préparées selon les étapes décrites dans la section 3.2.2. Nous avons utilisé Albumentations pour appliquer les transformations (redimensionnement à 224x224 pixels, normalisation avec les statistiques d'ImageNet, augmentations comme rotations et ajustements de contraste)..
3. **Fine-Tuning des Modèles** : Nous avons gelé les premières couches des modèles pour préserver les caractéristiques générales apprises sur ImageNet, tout en entraînant les couches supérieures sur notre dataset. Par exemple, pour MobileNetV3_Large, nous avons gelé les couches jusqu'au dernier bloc convolutif.
4. **Intégration de l'Explicabilité** : Nous avons intégré les méthodes XAI (Grad-CAM, SHAP, LIME) dans le pipeline d'analyse. Les résultats (heatmaps, valeurs SHAP, superpixels LIME) ont été encapsulés dans les métadonnées
5. **Évaluation et Sélection** : Après l'entraînement, nous avons évalué les modèles sur l'ensemble de test (section 3.4).

4.3 Implémentation des microservices

Nous avons implémenté l'architecture microservices définie dans la section 2.2, en utilisant Spring Boot pour chaque service. Voici les détails de l'implémentation des principaux services :

1. **config-server** : (port 8888)
 - **Rôle** : Gère les configurations centralisées (ex. : paramètres de connexion à PostgreSQL).
 - **Implémentation** : Utilise Spring Cloud Config pour une gestion dynamique.
2. **discovery-service** : (port 8761)
 - **Rôle** : Enregistre et découvre les services via Netflix Eureka, facilitant la communication dans un environnement distribué.
 - **Implémentation** : Chaque microservice s'enregistre automatiquement au démarrage.
3. **gateway-service** : (port 8222)
 - **Rôle** : Centralise les requêtes entrantes, distribue les requêtes vers les services Backend avec un Load Balancer, gère le partage des ressources entre les origines (CORS), et agit comme un premier point d'application de la sécurité.
 - **API** : GET /api/v1/general.

- **Sécurité** : Intègre un limiteur de débit (100 requêtes/minute).
- 4. **user-service** : (port 8892)
 - **Rôle** : Gère les profils utilisateurs et leurs rôles (DéTECTive, Juge, etc.) et l'authentification (génération/validation de JWT).
 - **API** : POST `api/v1/auth/register`, GET `api/v1/users/{id}`, etc.
 - **Implémentation** : Stocke les données dans PostgreSQL, avec des mots de passe hachés (Bcrypt).
- 5. **imageManagement-service** : (port 8050)
 - **Rôle** : Gère le stockage, la récupération et l'annotation des images.
 - **API** : POST `api/v1/images/upload`, GET `/images/{id}`, etc.
 - **Implémentation** : Stocke les images dans un système de fichiers sécurisé, avec métadonnées dans MongoDB(GridFS) et maintient la chaîne de traçabilité.
- 6. **imageAnalysis-service** : (port 8851)
 - **Rôle** : Analyse les images avec les modèles IA (MobileNetV3_Large), génère des visualisations XAI, et publie les événements d'analyse terminée avec Kafka.
 - **API** : POST `api/v1/analysis/{imageId}`, GET `api/v1/analysis/{analysisId}`, GET `api/v1/analysis/{analysisId}/xai-visualization/{type}`, etc.
 - **Implémentation** : Intègre PyTorch lors de l'exécution des scripts Python, et stocke les résultats d'analyse dans MongoDB.
- 7. **report-service** : (port 8053)
 - **Rôle** : Génère des rapports détaillés automatiques ou suivant des modèles personnalisables, annote et exporte les rapports d'analyse et écoute les événements de fin d'analyse via Kafka.
 - **API** : GET `api/v1/reports/auto-generated/{imageId}`, POST `api/v1/legal-documents/{documentId}/export`.
 - **Implémentation** : Utilise iTextPDF pour exporter les rapports en PDF et stocke les rapports dans MongoDB.
- 8. **notification-service** : (port 8668)
 - **Rôle** : Ecoute les événements Kafka et envoie des notifications (ex. : confirmation d'upload, rapport disponible) via Kafka.
 - **Implémentation** : Publie des messages dans un topic Kafka, consommés par les clients (email, interface).

Chaque service gère les erreurs via des codes HTTP standard (ex. : 400 pour une entrée invalide) et journalise les actions dans un système d'audit pour la traçabilité judiciaire.

4.4 Intégration et déploiement

Nous avons intégré les différents composants (modèles IA et microservices) dans un système unifié, puis déployé la solution sur une infrastructure docker. Les étapes d'intégration et de déploiement sont les suivantes :

1. Intégration des Modèles dans les Microservices :

- Le `imageAnalysis-service` intègre les modèles (`falsification_detector.pth`) en exécutant directement un script Python via `ProcessBuilder`.
- Les résultats de l'analyse (prédictions (`isFalsified`, `confidenceScore`), chemins vers les visualisations XAI comme GradCAM, LIME, SHAP) sont stockés dans MongoDB. Le `report-service` accède ensuite à ces résultats pour la génération de rapports.

2. Communication entre Microservices :

- Les requêtes REST synchrones transitent par le `gateway-service`, qui route vers les microservices appropriés, en utilisant Eureka pour la découverte de services. (ex. : `/api/v1/analysis/**` vers `imageAnalysis-service`, `/api/v1/images/**` vers `imageManagement-service`)
- Les notifications asynchrones via Kafka sont utilisées pour découpler les services (ex. : rapport généré)

3. Conteneurisation :

- Chaque service (`imageAnalysis-service`, `imageManagement-service`, `notification-service`, `report-service`, `user-service`) est destiné à être conteneurisé pour le déploiement.

4. Tests d'Intégration :

- Les tests unitaires pour les services Spring Boot sont réalisés avec JUnit (via `spring-boot-starter-test`)

4.5 Conception des interfaces graphiques

Nous avons développé des interfaces graphiques avec Angular pour offrir une expérience utilisateur intuitive et adaptée aux besoins des acteurs judiciaires (DéTECTIVE, EXPERT JUDICIAIRE, AVOCAT, JUGE, ADMINISTRATEUR). Les interfaces ont été conçues pour être responsives et accessibles sur différents appareils (desktop, tablette, mobile).

1. Conception de l'Interface :

- La conception s'est appuyée sur des principes modernes d'UX/UI, utilisant des bibliothèques reconnues comme Bootstrap et Angular Material pour assurer une cohérence visuelle et une facilité d'utilisation.

- L’interface est structurée autour des rôles utilisateurs définis (Admin, Expert, DéTECTive, Avocat, Juge), avec des tableaux de bord et des fonctionnalités spécifiques accessibles via un système de routage protégé par des gardes (‘AuthGuard’, ‘RoleGuard’).

2. Implémentation avec Angular :

- L’application front-end est construite sur le framework Angular, exploitant une architecture modulaire à composants réutilisables avec chargement paresseux (lazy loading) pour les sections dédiées à chaque rôle (Admin, Expert, DéTECTive, etc.).
- L’interaction avec les microservices back-end est gérée via ‘HttpClientModule’ et des intercepteurs (‘AuthInterceptor’ pour l’ajout de tokens JWT, ‘ErrorInterceptor’ pour la gestion centralisée des erreurs HTTP avec notifications ‘Toastr’, ‘BaseUrlInterceptor’ pour préfixer les appels avec l’URL de la gateway).

3. Visualisation des Résultats :

La présentation des données et des résultats d’analyse est un élément clé.

- **Tableaux de Bord** : Des dashboards dédiés (ex : ‘Admin Dashboard’) présentent des indicateurs clés et des graphiques pour visualiser les tendances d’activité.
- **Détails et Analyse** : Des vues dédiées permettent de consulter les informations spécifiques d’une image, d’une analyse, d’un rapport ou autre. Les résultats d’analyse d’image affichent notamment le statut (Falsified/Authentic) et le score de confiance.
- **Gestion des Fichiers** : L’interface d’upload utilise ‘ngx-dropzone’ pour une expérience glisser-déposer, et les images sont visualisables dans les détails des affaires ou des analyses.

Pour mieux illustrer ce qui précède, nous présentons ci-dessous des interfaces représentatives des principales fonctionnalités accessibles aux utilisateurs judiciaires (DéTECTive, Expert Judiciaire, Avocat, Juge, Administrateur).

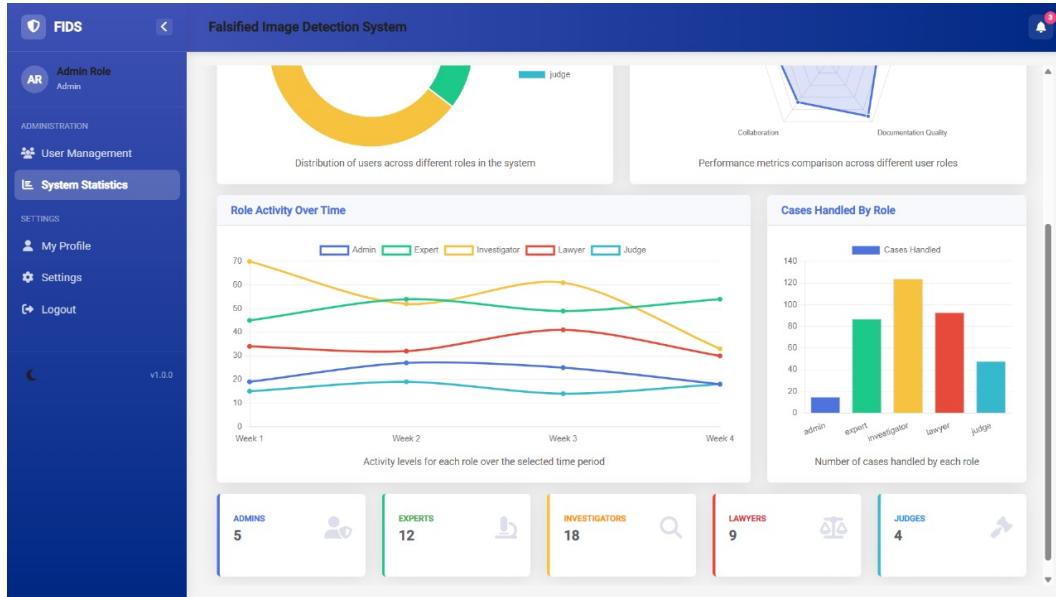


FIGURE 4.1 – Interface des Statistiques du Système (Vue Admin)

Destinée à un administrateur, cette interface présente des statistiques globales du système. Elle inclut un graphique circulaire montrant la distribution des rôles (juges, experts, etc.), un graphique d'activité par rôle sur le temps, une comparaison des performances via un radar chart, et un histogramme des cas traités par rôle, avec des compteurs (par exemple, 5 admins, 12 experts).

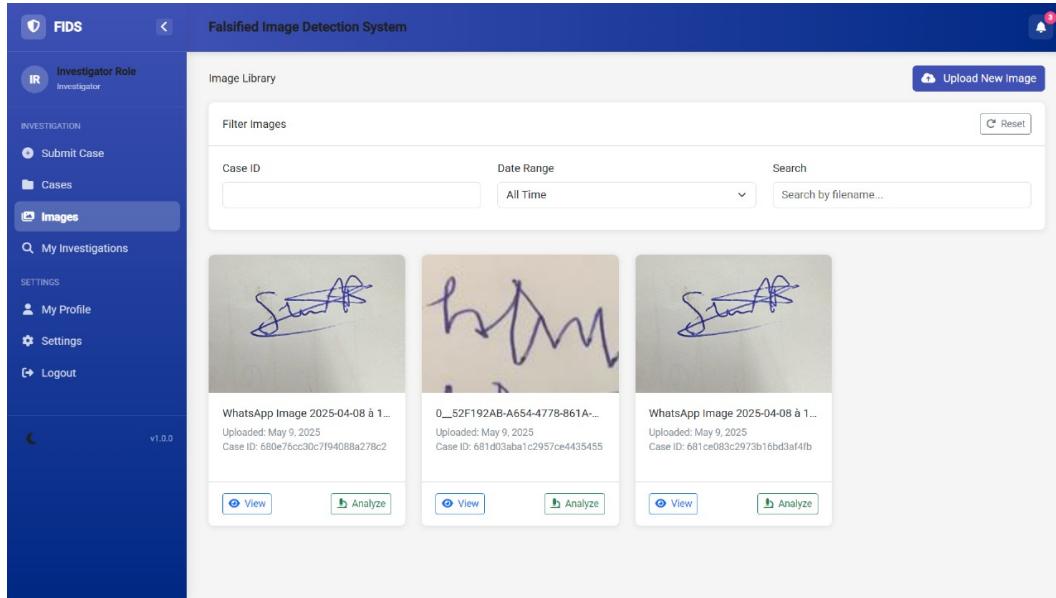


FIGURE 4.2 – Interface de Bibliothèque d'Images (Vue DéTECTIVE)

Cette interface permet à un DéTECTIVE de consulter et gérer la bibliothèque d'images. Elle affiche des images associées à des affaires judiciaires, avec des options de filtrage par ID de cas, date, et recherche par nom de fichier. Les actions "View" et "Analyze" sont disponibles pour chaque image.

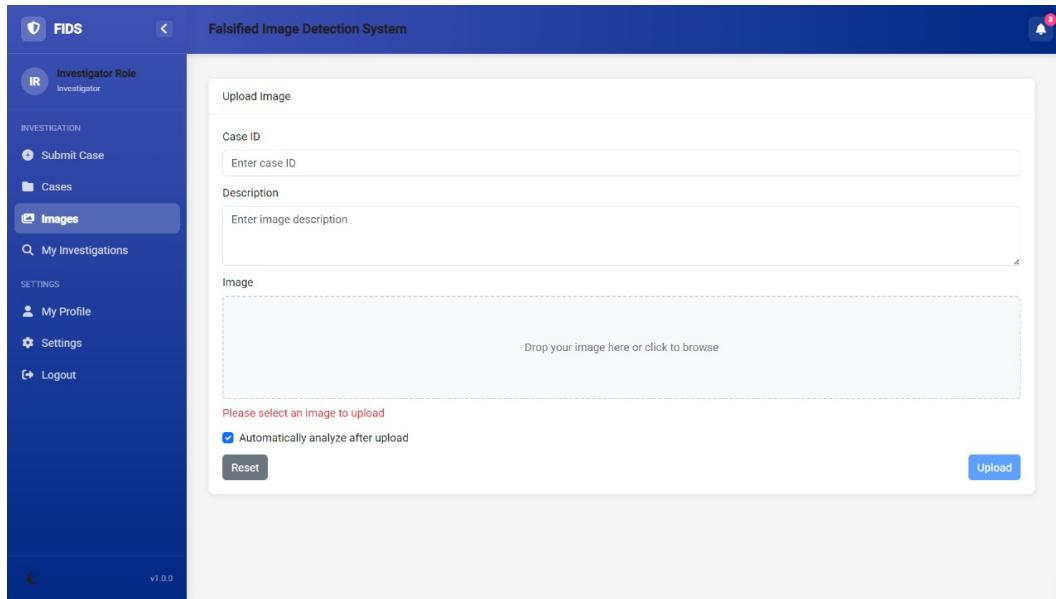


FIGURE 4.3 – Interface de Téléversement d’Image Initial (Vue DéTECTIVE)

Cette interface initiale permet à un DéTECTIVE de préparer le téléversement d’une image. Elle inclut des champs pour l’ID du cas et la description, une zone de dépôt d’image avec une invite ("Drop your image here or click to browse"), et une option d’analyse automatique, avec un message d’erreur si aucune image n’est sélectionnée.

Expert Reports						
Report ID	Case ID	Case Number	Title	Created Date	Status	Actions
681c42794ab57a07ee62b598	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Report	May 8, 2025	UNDER REVIEW	View Export PDF
681c432844b57a07ee62b599	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Report	May 8, 2025	UNDER REVIEW	View Export PDF
681c44bf1eda9d26f68ccf4d	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Reportxxxx	May 8, 2025	UNDER REVIEW	View Export PDF
681c44f41eda9d26f68ccf4e	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Reportzzzf	May 8, 2025	UNDER REVIEW	View Export PDF
681c45621eda9d26f68ccf4f	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Reportfx bvn	May 8, 2025	UNDER REVIEW	View Export PDF
681c46a6ecb417034adf3fc	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Report	May 8, 2025	UNDER REVIEW	View Export PDF
681c4861ecb417034adf3fd1	681c419044b57a07ee62b596	CASE-CC2A1A58	Expert Report	May 8, 2025	UNDER REVIEW	View Export PDF
681c4a34ecb417034adf3fd5	681c492cecb417034adf3fd2	CASE-3363CFB0	Expert Report	May 8, 2025	ASSIGNED	View Export PDF
681cdfd5c2973b16bd3af4f9	681c492cecb417034adf3fd2	CASE-3363CFB0	Expert Report	May 8, 2025	UNDER REVIEW	View Export PDF
681ce1b2c2973b16bd3af500	681ce083c2973b16bd3af4fb	CASE-031A6C68	Expert Report	May 8, 2025	ASSIGNED	View Export PDF

FIGURE 4.4 – Interface des Rapports d’Expert (Vue Expert)

Destinée à un Expert, cette interface liste les rapports générés (par exemple, pour CASE-CC2A1A58). Elle affiche l’ID du rapport, l’ID du cas, le titre, la date de création, le statut, et des actions comme "View" et "Export PDF". Une option "Create New Report" est également disponible.

Chapitre 4 : Implémentation

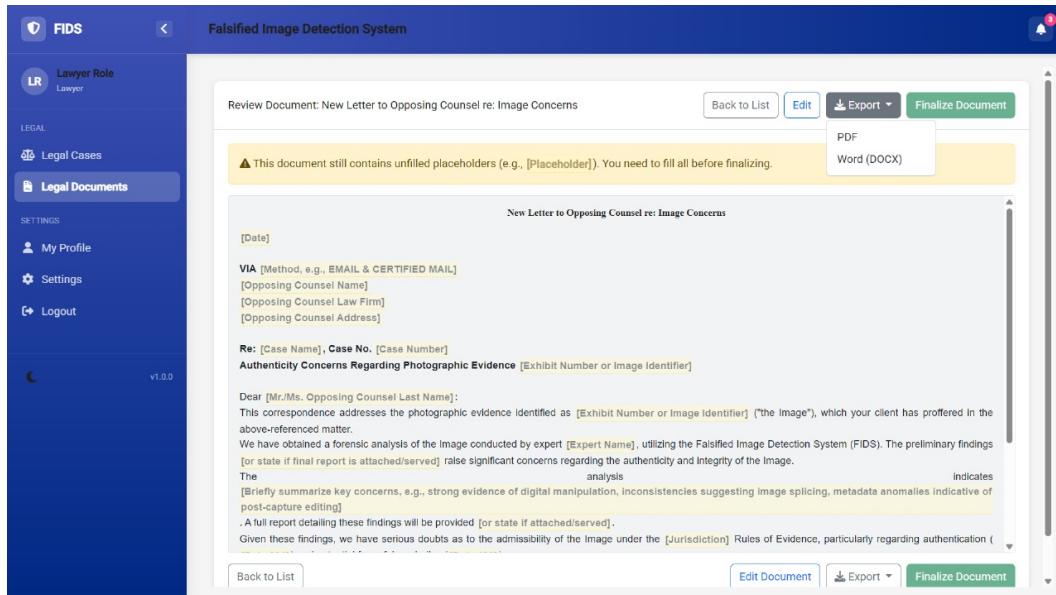


FIGURE 4.5 – Interface de Révision de Document Juridique (Vue Avocat)

Cette interface permet à un avocat de réviser un document juridique, comme une lettre à un conseil adverse concernant une image. Elle montre un modèle avec des espaces à remplir (date, VIA, nom, etc.), un avertissement sur les placeholders non remplis, et des options "Edit", "Export" (PDF/Word), et "Finalize Document".

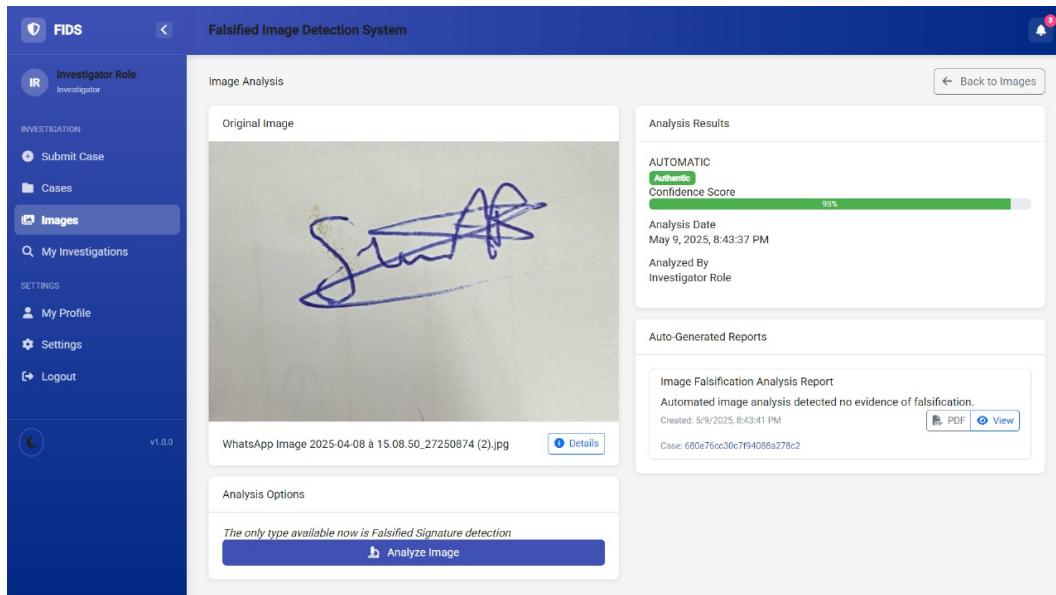


FIGURE 4.6 – Interface d'Analyse d'Image

Cette interface affiche l'analyse d'une image par un enquêteur, avec un résultat "Authentic" (95 % de confiance). Elle inclut un rapport PDF généré automatiquement et des options d'analyse supplémentaires.

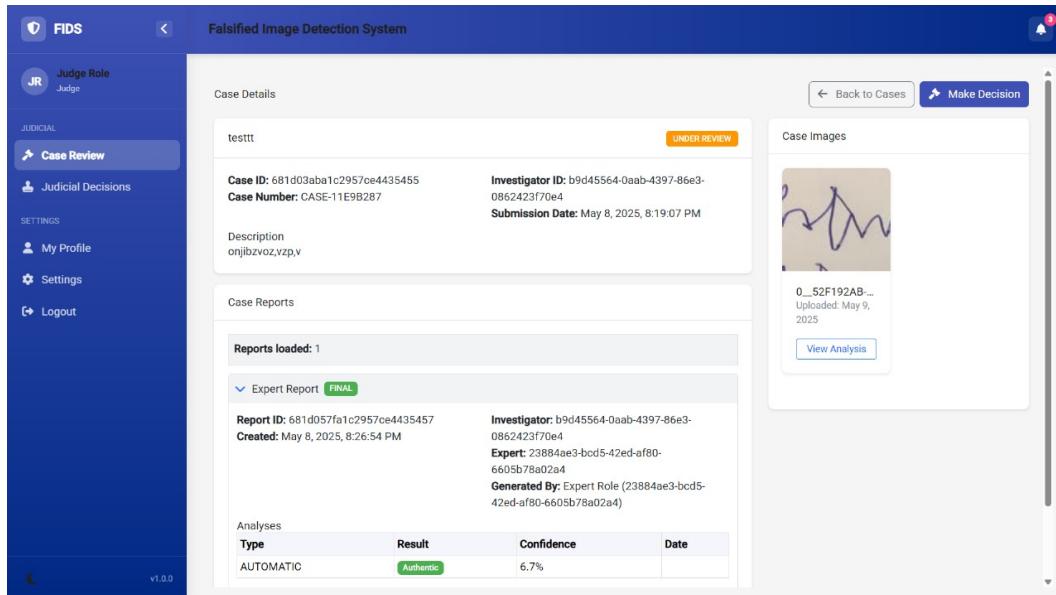


FIGURE 4.7 – Interface de Revue des affaires (Vue Juge)

Destinée à un juge, cette interface présente les détails d'une affaire judiciaires en statut UNDER REVIEW. Elle inclut des informations sur l'enquêteur, la date de soumission, une image associée, un rapport expert (FINAL), et des options comme "View Analysis" et "Make Decision".

4. **Tests Utilisateurs** : Des tests utilisateurs ont été menés pour évaluer l'intuitivité, l'ergonomie et l'adéquation des interfaces aux processus métiers de chaque rôle judiciaire. Les retours ont permis d'affiner la navigation, la disposition des éléments et la clarté des informations présentées.

Conclusion

Ce chapitre a détaillé l'implémentation de notre solution, couvrant l'environnement technique, le développement des modèles de Transfer Learning, la construction des micro-services, l'intégration et le déploiement, ainsi que la conception des interfaces graphiques.

Chapitre 5

Résultats et Analyse

Introduction

Dans ce chapitre, nous présentons les résultats obtenus à la suite de l'implémentation du système de détection d'images falsifiées, suivant la méthodologie décrite dans les chapitres précédents. Nous évaluons les performances des modèles de Transfer Learning étudiés. Après, nous menons une discussion approfondie expliquant les raisons des performances observées, suivie d'une critique des limites.

5.1 Résultats des modèles

Nous avons évalué les modèles entraînés pour détecter les images falsifiées en utilisant un ensemble de test représentatif de notre cas d'étude, conformément à la méthodologie décrite au chapitre 3. Les métriques retenues pour cette évaluation incluent l'accuracy, le F1-score, la précision et le recall, permettant une analyse multicritère des performances. Les résultats sont présentés dans le Tableau 5.1 et illustrés par un histogramme dans la Figure 5.1, permettant une comparaison visuelle des performances.

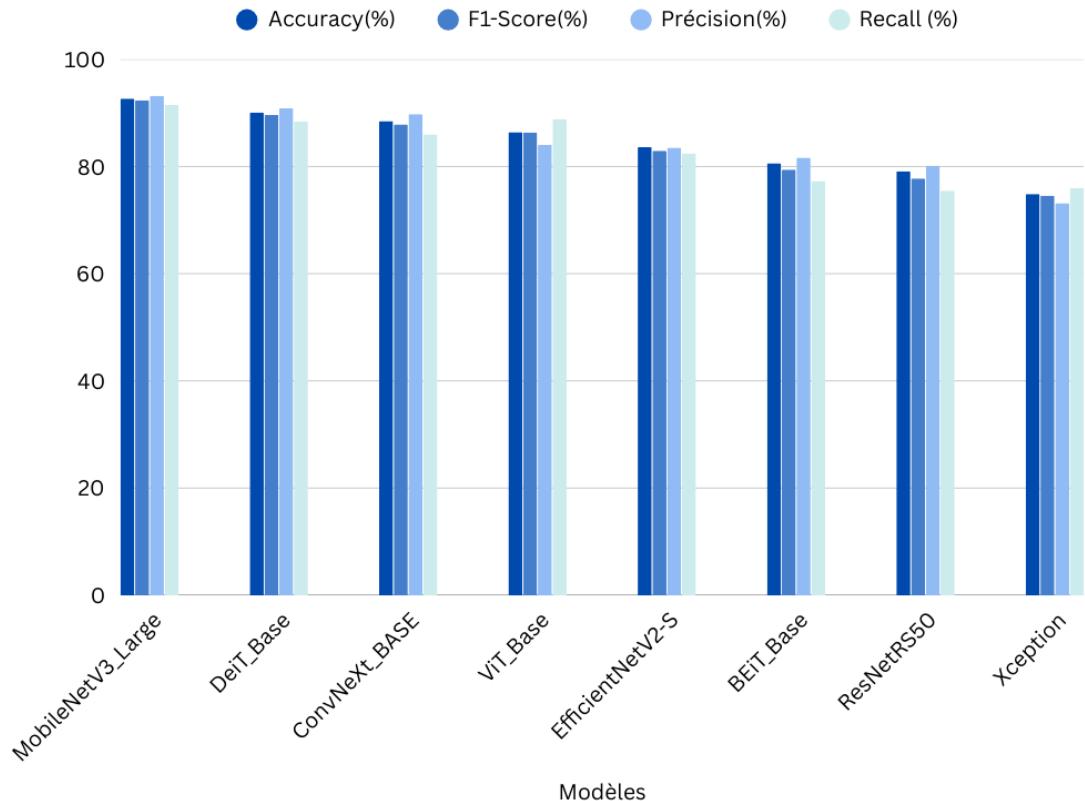


FIGURE 5.1 – Histogramme comparant les performances des modèles sur l'ensemble de test

TABLE 5.1 – Comparaison des performances des modèles sur l'ensemble de test

Modèle	Accuracy (%)	F1-Score (%)	Précision (%)	Recall (%)
MobileNetV3_Large	92.67	92.36	93.2	91.54
DeiT_Base	90.09	89.62	90.85	88.42
ConvNeXt_BASE	88.47	87.83	89.77	85.97
ViT_Base	86.42	86.36	84.00	88.86
EfficientNetV2-S	83.62	82.96	83.52	82.41
BEiT_Base	80.61	79.41	81.65	77.28
ResNetRS50	79.09	77.75	80.14	75.5
Xception	74.89	74.53	73.17	75.94

5.2 Discussion des résultats obtenus

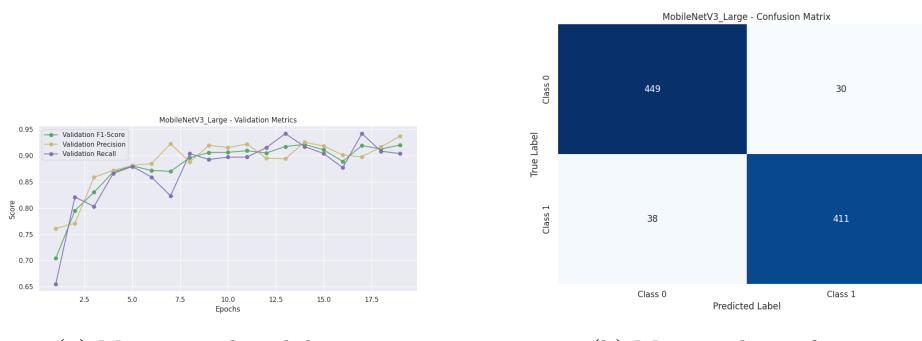
5.2.1 Analyse Comparative des Modèles

Nous avons comparé les performances des huit modèles étudiés (MobileNetV3_Large, DeiT_Base, ConvNeXt_Base, ViT_Base, EfficientNetV2-S, BEiT_Base, ResNetRS50, Xception) pour identifier celui offrant le meilleur compromis entre précision, robustesse et efficacité computationnelle.

Cette analyse s'appuie sur les métriques calculées (Tableau 5.1) et les visualisations générées des modèles. (Figures 5.2 à 5.8).

Nous remarquons une dominance du MobileNetV3_Large. En effet, ça se traduit à partir des critères suivants :

- **Exactitude (Accuracy)** : Avec **92.67%**, MobileNetV3_Large (Figure 5.2) surpassé son plus proche concurrent, DeiT_Base (Figure 5.3)(90.09%), de **2.58%**. Cette marge, bien que paraissant modeste, se traduit par une réduction notable des erreurs globales de classification. Par rapport à Xception (Figure 5.4) (74.89%), le moins performant, l'écart est de **17.78%**, illustrant une différence de capacité de généralisation considérable.



(a) Métriques de validation

(b) Matrice de confusion

FIGURE 5.2 – Résultat du modèle MobileNetV3_Large

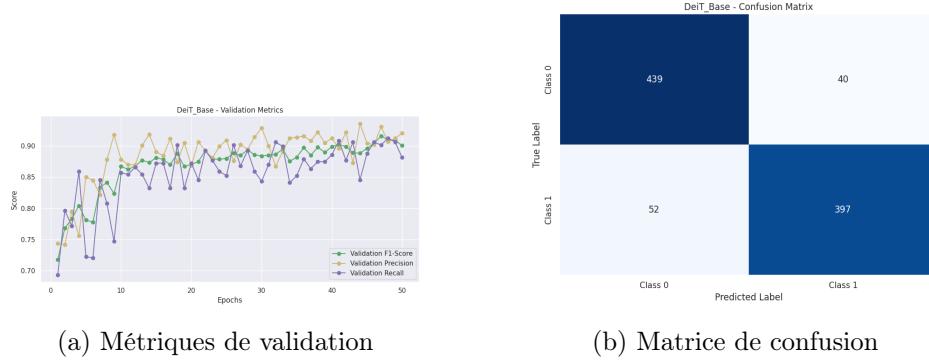


FIGURE 5.3 – Résultat du modèle Deit_Base

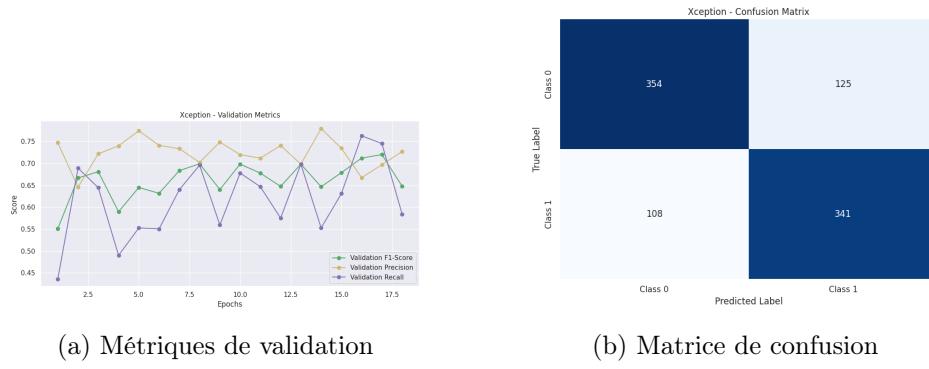


FIGURE 5.4 – Résultat du modèle Xception

- **Rappel (Recall)** : C'est ici que MobileNetV3_Large (91.54%) démontre un avantage crucial pour un usage judiciaire. Il identifie correctement **3.12%** plus de signatures falsifiées que DeiT_Base (88.42%) et **2.68%** de plus que ViT_Base (Figure 5.5) (88.86%). De manière plus critique, il surpassé Xception (75.94%) de **15.6%** en recall. Cela signifie que Xception *manque* près de 1 signature falsifiée sur 4 que MobileNetV3_Large *détecte*.

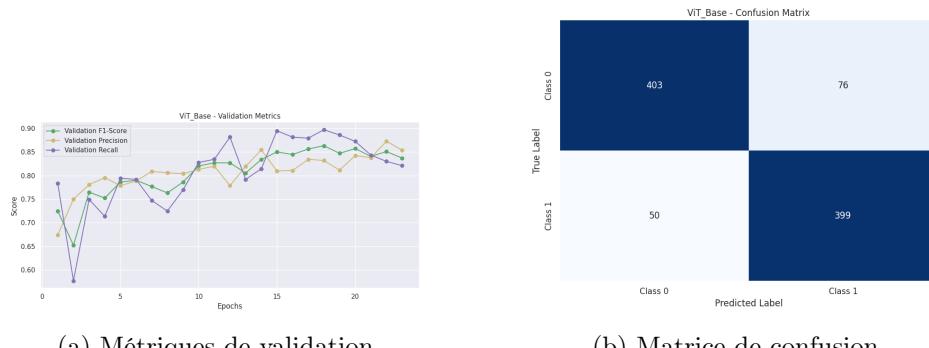
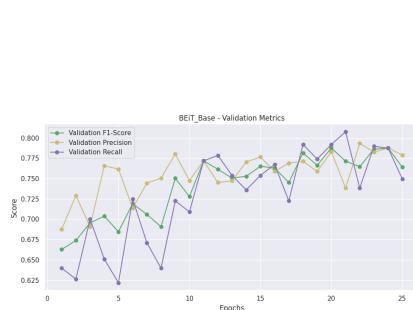
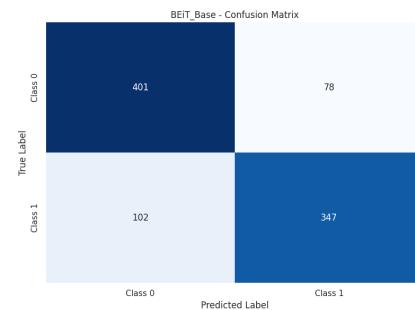


FIGURE 5.5 – Résultat du modèle ViT_Base

- **Précision** : MobileNetV3_Large (93.20%) est également en tête. Lorsqu'il classifie une signature comme falsifiée, il est correct dans 93.20% des cas, contre 90.85% pour DeiT_Base (une différence de **2.35%**). Cela réduit le nombre de "fausses alertes".
- **Score F1** : Le F1-score (92.36%) de MobileNetV3_Large, qui combine précision et rappel, est supérieur de **2.74%** à celui de DeiT_Base (89.62%), confirmant son meilleur équilibre global.
- **Efficacité Computationnelle** : Avec **5.5 millions de paramètres**, MobileNetV3_Large est extraordinairement léger comparé aux modèles Transformers comme DeiT_Base ou ViT_Base (86M paramètres, soit **plus de 15 fois plus de paramètres**) ou BEiT_Base (307M paramètres, soit **plus de 55 fois plus**). Cette différence massive a un impact direct sur la vitesse d'inférence, les besoins en ressources de déploiement et la consommation énergétique.



(a) Métriques de validation



(b) Matrice de confusion

FIGURE 5.6 – Résultat du modèle BEiT_Base

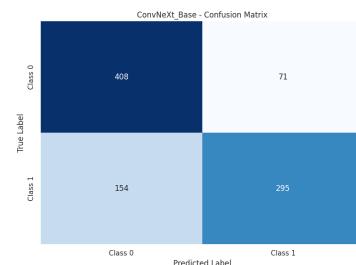
5.2.2 Analyse Comparative des Modèles CNN vs. Transformers

Modèles CNN performants :

- Outre MobileNetV3_Large, **ConvNeXt_Base** (Figure 5.7) (88.47% accuracy, 85.97% recall) est le deuxième CNN le plus performant. Il est toutefois en retrait de MobileNetV3_Large d'environ **4.2% en accuracy** et **5.57% en recall**.



(a) Métriques de validation



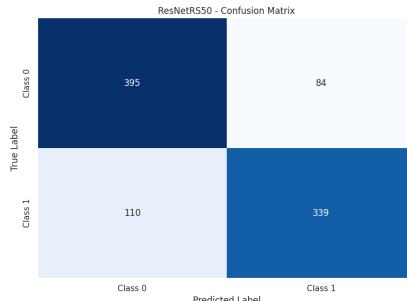
(b) Matrice de confusion

FIGURE 5.7 – Résultat du modèle ConvNeXt_Base

- **ResNetRS50** (Figure 5.8) (79.09% accuracy, 75.50% recall) et **Xception** (74.89% accuracy, 75.94% recall) sont significativement moins performants. Leur recall particulièrement bas (environ 75%) est problématique, indiquant qu'ils manquent une falsification sur quatre.



(a) Métriques de validation



(b) Matrice de confusion

FIGURE 5.8 – Résultat du modèle ResNetRS50

La performance supérieure de MobileNetV3_Large et, dans une moindre mesure, de ConvNeXt_Base parmi les CNNs suggère que leurs architectures spécifiques (convolutions séparables en profondeur, blocs modernes) sont mieux adaptées à la granularité fine des caractéristiques des signatures que les architectures CNN plus traditionnelles comme ResNet ou Xception pour ce dataset.

Modèles Transformers :

- **DeiT_Base** (90.09% accuracy, 88.42% recall) et **ViT_Base** (86.42% accuracy, 88.86% recall) sont les meilleurs Transformers. Bien que leur recall soit bon (proche de 88-89%), il reste inférieur à celui de MobileNetV3_Large. La différence de recall de **2.68 à 3.12%** par rapport à MobileNetV3 signifie que ce dernier détecte quelques pourcents de falsifications en plus, ce qui peut être décisif.
- **BEiT_Base** (80.61% accuracy, 77.28% recall) est le Transformer le moins performant, probablement en raison de sa très grande taille (307M paramètres) qui pourrait entraîner un sur-ajustement ou nécessiter un dataset beaucoup plus conséquent pour une convergence optimale. Son recall est inférieur de **14.26 %** à celui de MobileNetV3_Large.

Bien que les Transformers soient puissants, leur focalisation sur les relations globales pourrait être moins pertinente pour les signatures, où les altérations sont souvent des détails locaux subtils. L'avantage de MobileNetV3_Large suggère que la capacité à capturer efficacement ces micro-caractéristiques est primordiale.

Le **taux de faux négatifs** ($100\% - \text{Recall}\%$) est aussi une métrique critique.

- MobileNetV3_Large : **8.46%** de faux négatifs.

- DeiT_Base : **11.58%** de faux négatifs (**3.12 %** plus que MobileNetV3).
- ViT_Base : **11.14%** de faux négatifs (**2.68%** de plus).
- ConvNeXt_Base : **14.03%** de faux négatifs (**5.57%** de plus).
- Xception : **24.06%** de faux négatifs (**15.6%** de plus).

Un écart de 5 à 15 points dans le taux de faux négatifs est considérable. Dans un lot de 100 signatures falsifiées, Xception en manquerait 15 de plus que MobileNetV3_Large. Cette différence justifie à elle seule la préférence pour des modèles à recall élevé.

5.2.3 Facteurs influençant les performances

La hiérarchie des performances observée découle de l'adéquation entre l'architecture du modèle et la nature spécifique des données de signatures :

1. **Capture des Motifs Locaux vs. Globaux** : MobileNetV3_Large, avec ses convolutions séparables et ses modules Squeeze-and-Excitation, excelle dans l'extraction de caractéristiques locales fines. Les Transformers, bien que performants pour le contexte global, peuvent ne pas accorder autant d'importance à ces micro-détails, ce qui se reflète par un recall légèrement inférieur sur cette tâche.
2. **Efficacité et Complexité** : La légèreté de MobileNetV3 (5.5M paramètres) contraste fortement avec la complexité des Transformers (86M-307M). Pour un dataset de la taille utilisée (environ 6000 images), un modèle plus léger peut converger plus facilement vers une bonne solution généralisable, tandis que les modèles très complexes risquent le sur-ajustement ou nécessitent des techniques de régularisation plus poussées ou davantage de données.
3. **Optimisation Architecturale Spécifique** : La conception de MobileNetV3 via la recherche d'architecture neuronale (NAS) l'a spécifiquement optimisé pour un bon ratio performance/coût sur des tâches de vision mobile, ce qui se traduit par une efficacité remarquable sur notre cas d'usage qui partage des contraintes similaires (besoin de rapidité, importance des détails locaux).

En conclusion, l'analyse comparative démontre que MobileNetV3_Large n'est pas seulement marginalement meilleur, mais offre un avantage substantiel en termes de recall, de précision et d'efficacité globale par rapport aux autres modèles testés pour la détection de falsification de signatures manuscrites. Ces différences quantitatives renforcent son choix comme solution privilégiée pour le système développé.

5.3 Résultats et analyse de l'explicabilité (XAI)

Au-delà des métriques de performance quantitatives présentées précédemment, la compréhension des décisions prises par les modèles d'apprentissage profond est essentielles.

C'est pourquoi notre système intègre des techniques d'Intelligence Artificielle Explicable (XAI) visant à visualiser les régions d'une image de signature qui ont le plus influencé la prédiction du modèle, qu'il s'agisse de classer la signature comme authentique ou falsifiée.

La Figure 5.9 présente un exemple des visualisations XAI produites par notre système.

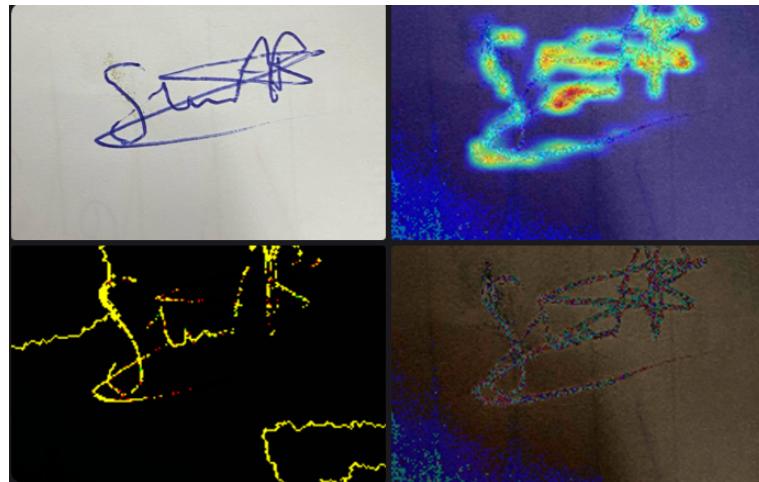


FIGURE 5.9 – Visualisations XAI

L'analyse de la Figure 5.9 révèle plusieurs aspects :

- La signature originale (en haut à gauche) sert de référence.
- La carte de chaleur (en haut à droite), générée par une méthode de type Grad-CAM, met en évidence les zones de la signature sur lesquelles le modèle a concentré son "attention" pour prendre sa décision. Les couleurs plus chaudes (rouge, jaune) indiquent une plus forte activation. Cela peut aider un expert à identifier si le modèle se focalise sur des caractéristiques pertinentes ou potentiellement sur des artefacts non pertinents.
- L'attribution d'importance des caractéristiques (en bas à gauche), issue de méthodes comme SHAP ou LIME, quantifie la contribution de différentes parties de l'image (pixels ou groupes de pixels) à la prédiction finale. Les zones colorées en jaune ou rouge signalent une forte influence sur la probabilité d'une classe donnée.
- La superposition (en bas à droite) combine l'information visuelle de la signature avec les zones d'importance, facilitant une interprétation directe des éléments graphiques spécifiques qui ont conduit à la classification.

Ces visualisations XAI, intégrées dans les rapports générés par le système, offrent un outil précieux aux acteurs judiciaires. Elles permettent non seulement de renforcer la confiance dans les résultats du système en rendant le processus de décision moins opaque, mais aussi d'orienter l'analyse humaine.

5.4 Critiques et limites

Nous avons identifié ici les principales limites de notre solution et celles que nous avons rencontré lors du développement, qui pourraient être abordées dans de futures itérations.

1. Ressources Computationnelles Limitées :

- L'entraînement des modèles et l'optimisation des hyperparamètres ont été contraints par l'accès à des ressources GPU limitées (par exemple, utilisation de versions gratuites de plateformes cloud comme Google Colab ou Kaggle Kernels).

Bien que nous ayons pu mener nos expérimentations à terme, des ressources plus conséquentes auraient permis d'explorer des plages d'hyperparamètres plus larges, d'entraîner les modèles sur un plus grand nombre d'époques si nécessaire, ou de tester des architectures encore plus complexes. Cette contrainte est un facteur à considérer pour la reproductibilité ou l'extension du travail sur des datasets beaucoup plus volumineux.

2. Qualité des Données :

- Les erreurs de classification sont souvent dues à des falsifications subtiles ou à des images bruitées (ex. : faible contraste, artefacts de numérisation). Cela souligne la dépendance du modèle à la qualité des données d'entrée, un défi courant en vision par ordinateur.

3. Généralisation :

- Les modèles ont été entraînés sur un dataset spécifique de signatures scannées. Leur performance pourrait diminuer sur d'autres types d'images falsifiées (ex. : photos, documents imprimés), limitant la généralisation.

4. Méthodes d'Explicabilité :

- Bien que les méthodes XAI soient efficaces, SHAP et LIME présentent des faiblesses sur les images bruitées ou à faible contraste (couverture de 90-92 %). Cela limite leur fiabilité dans certains cas complexes.
- Les Experts Judiciaires peuvent demander des descriptions textuelles automatiques plus détaillées pour accompagner les *heatmaps*, ce qui n'a pas été implémenté dans cette version.

5. Performance et Scalabilité :

- Bien que le système soit performant, des erreurs réseau temporaires peuvent affecter la disponibilité. Cela pourrait poser problème dans un contexte judiciaire où une disponibilité de 100 % est souvent attendue.

Conclusion

Dans ce chapitre nous avons détaillé les résultats quantitatifs des modèles utilisés pour la détection d’images falsifiées (les signatures manuscrites), tout en les comparant . Cette analyse comparative a démontré l’efficacité supérieure de MobileNetV3_Large pour cette tâche spécifique, soulignant sa pertinence pour l’intégration dans une plateforme web.

Conclusion et perspectives

Au terme de ce projet, nous avons conduit une étude approfondie de la détection d'images falsifiées dans un contexte judiciaire, avec un accent particulier sur l'intelligence artificielle, à travers l'étude et la comparaison de modèles de Transfer Learning, tout en développant une plateforme adaptée aux exigences judiciaires.

Ce que nous avons fait repose principalement sur l'analyse des huit modèles de Transfer Learning – ResNetRS50, Xception, MobileNetV3_Large, ViT_Base, DeiT_Base, BEiT_Large, EfficientNetV2-S, et ConvNeXt Base. Nous avons optimisé leurs hyperparamètres pour maximiser leurs performances sur un dataset de signatures authentiques et falsifiées, tout en intégrant des techniques d'explicabilité comme GradCAM et SHAP afin de répondre aux besoins de transparence judiciaire. La comparaison des modèles a révélé que MobileNetV3_Large se distingue avec accuracy respectives de 92.67 %, étant particulièrement efficace grâce à sa légèreté et sa robustesse face aux images bruitées. Par ailleurs, nous avons conçu une plateforme modulaire basée sur une architecture microservices, avec une interface graphique pour faciliter l'interaction des utilisateurs judiciaires, assurant sécurité et traçabilité.

Dans l'ensemble, suite à la réalisation de ce projet, nous avons non seulement conclut la nécessité d'adapter les modèles d'IA aux spécificités du domaine judiciaire, où la précision et l'explicabilité sont primordiales, mais aussi les défis liés aux perturbations et à la généralisation des données.

Dans le futur, nous envisageons de poursuivre l'amélioration des modèles en renforçant leur robustesse face aux attaques adversariales, notamment pour les Transformers, à travers un entraînement adversatif. Une diversification du dataset permettra de mieux généraliser les performances.

Nous prévoyons également d'enrichir l'explicabilité en ajoutant des descriptions textuelles automatiques aux résultats, facilitant leur interprétation.

Enfin, une exploration de modèles hybrides combinant, par exemple, deux modèles pourrait optimiser l'équilibre entre précision et efficacité.

Bibliographie

- [1] Abdulrahman Abbas, Belal Al-Khateeb, and Mazin Mohammed. Breast cancer images classification using a new transfer learning technique. 01 2023.
- [2] Saleh AlTakrouri, Norliza Noor, Norulhusna Ahmad, Taghreed Justinia, and Sahnus Usman. Image super-resolution using generative adversarial networks with efficientnetv2. *International Journal of Advanced Computer Science and Applications*, 14, 01 2023.
- [3] Hangbo Bao, Li Dong, and Furu Wei. Beit : BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021.
- [4] Shenglong Chen, Yoshiki Ogawa, and Yoshihide Sekimoto. Large-scale individual building extraction from open-source satellite imagery via super-resolution-based instance segmentation approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 195 :129–152, 01 2023.
- [5] Francois Chollet. Xception : Deep learning with depthwise separable convolutions. pages 1800–1807, 07 2017.
- [6] DaLight. Falsified image detection, 2025. <https://github.com/DaL1ght1/PCD-2025-40/blob/main/AI/falsified-images-pytorch.ipynb>.
- [7] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai) : A survey, 2020.
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words : Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.
- [9] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch-Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. *Microservices : yesterday, today, and tomorrow*. 01 2017.
- [10] Hany Farid. Image forgery detection. *IEEE Signal Processing Magazine*, 26(2) :16–25, 2009.
- [11] M. Fowler and J. Lewis. Microservices, 2014. <https://martinfowler.com/articles/microservices.html>, consulté le 13/03/2025.

- [12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [14] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. pages 1314–1324, 10 2019.
- [15] Yuezun Li, Ming-Ching Chang, Hany Farid, and Siwei Lyu. In ictu oculi : Exposing ai generated fake face videos by detecting eye blinking. 06 2018.
- [16] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. Convnext : A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022.
- [17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. 12 2017.
- [18] Sam Newman. *Building Microservices*. O'Reilly Media, Inc., 2nd edition edition, August 2021.
- [19] Oracle. Learn about architecting microservices-based applications on oracle cloud, 10 2024. <https://docs.oracle.com/en/solutions/learn-architect-microservice/>, consulté le 12/03/2025.
- [20] Sivylla Paraskevopoulou. Explainable ai (xai) : Are we there yet ?, 2023. "<https://blogs.mathworks.com/deep-learning/2023/05/08/explainable-ai-xai-are-we-there-yet/>", consulté le 19/04/2025".
- [21] A.C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 53(2) :758–767, 2005.
- [22] Santhosh Raminedi, S. Shridevi, and Daehan Won. Multi-modal transformer architecture for medical image analysis and automated report generation. *Scientific Reports*, 14, 08 2024.
- [23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you ?" : Explaining the predictions of any classifier, 2016.
- [24] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, 2019.
- [25] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam : Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2) :336–359, October 2019.

- [26] Saisha Shetty, Naman Garg, Gayathri Mani, Malathy Chidambaranathan, Vineet Batta, and A Ramanathan. Automated identification of make and model of total wrist replacement implants using deep learning. 08 2023.
- [27] Mingxing Tan and Quoc V. Le. Efficientnetv2 : Smaller models and faster training. In *International Conference on Machine Learning*, 2021.
- [28] Chongyangzi Teng, Pengwei Yang, and Mengshen Guo. Multimodal in multi-label classification : A report, 06 2023.
- [29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.
- [30] Luisa Verdoliva. Media forensics and deepfakes : An overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5) :910–932, 2020.
- [31] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei Efros. Cnn-generated images are surprisingly easy to spot... for now. pages 8692–8701, 06 2020.
- [32] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2019.