Edvantis Higher Education Group
Institut Supérieur d'Ingénierie
des Affaires
de Marrakech

Rapport de Stage de Spécialité

De la 2ème année du cycle Ingénieur

Filière : Ingénierie des Systèmes Informatiques

Spécialité : Big Data & IA

Sujet de stage :

CLASSIFICATION D'IMAGES POUR L'AIDE AU DIAGNOSTIC DU CANCER CUTANÉ

Rédigé par :

FINGOUE Estelle Danielle

Encadrant académique :

M. AIT IBOUREK Lahcen

Encadrants professionnels :

Dr. IGUERNAISSI Rabah

Dr. MONNIER Jilliana

Année universitaire :

2023/2024

RESUME

Le mélanome est un type de cancer de la peau particulièrement agressif, responsable de la majorité des décès liés aux cancers cutanés dans le monde. La détection précoce est cruciale pour améliorer les taux de survie des patients, mais elle reste un défi complexe en raison de la difficulté à distinguer les lésions malignes des lésions bénignes. Ce projet vise à développer un système de diagnostic assisté par ordinateur (DAO) pour la détection du mélanome en s'appuyant sur des techniques de classification d'images à la pointe de la technologie et des stratégies de fusion de modèles. Deux approches principales ont été explorées : une approche classique utilisant des modèles tels que le K-Nearest Neighbors (KNN) et le Support Vector Machine (SVM), et une approche par apprentissage profond faisant appel à des réseaux de neurones convolutifs (CNN) et un modèle pré-entraîné EfficientNetB0. Le jeu de données utilisé se compose d'images dermoscopiques non segmentées, avec un ensemble d'entraînement de 1088 images de mélanomes et 5036 images de nævus, ainsi qu'un ensemble de test comprenant 273 mélanomes et 1260 nævus. Afin d'améliorer la précision du diagnostic, plusieurs techniques de fusion de modèles ont été appliquées, notamment la moyenne simple, la moyenne pondérée et la fusion basée sur la régression logistique. Les résultats de l'approche classique ont montré que KNN et SVM produisaient des performances comparables, avec 74 % de sensibilité, 72 % de spécificité et une AUC de 82 % pour KNN, et respectivement 80 %, 73 % et 83 % pour SVM. Cependant, les méthodes d'apprentissage profond ont nettement surpassé les modèles classiques, EfficientNetB0 atteignant une AUC de 88 %, contre 78 % pour le CNN de base. Parmi les approches de fusion, la fusion par moyenne pondérée a montré les meilleures performances globales, avec une sensibilité de 79 %, une spécificité de 81 % et une AUC de 88 %. La fusion par régression logistique, bien qu'efficace pour réduire les faux positifs avec une spécificité de 94 %, a montré une sensibilité plus faible, à 50 %. La fusion par moyenne simple a donné des résultats plus modestes. Ces résultats soulignent le potentiel des techniques de fusion de modèles pour améliorer la précision des systèmes de diagnostic automatisé du mélanome, notamment en intégrant des modèles d'apprentissage profond avec des approches classiques.

Mots clés : mélanome, diagnostic assisté par ordinateur, KNN, SVM, CNN, EfficientNetBO, fusion

ABSTRACT

Melanoma is a particularly aggressive type of skin cancer, responsible for the majority of deaths related to skin cancer worldwide. Early detection is crucial for improving patient survival rates, yet remains a complex challenge due to the difficulty in distinguishing between malignant and benign lesions. This project aims to develop a computer-aided diagnosis (CAD) system for melanoma detection by leveraging state-of-the-art image classification techniques and model fusion strategies. Two primary approaches were explored: a classical machine learning approach using models such as K-Nearest Neighbors (KNN) and Support Vector Machine (SVM), and a deep learning approach employing Convolutional Neural Networks (CNN) and a pre-trained EfficientNetB0 model. The dataset used consisted of non-segmented dermoscopic images, with a training set of 1088 melanoma images and 5036 nevus images, and a test set of 273 melanoma and 1260 nevus images. To enhance diagnostic accuracy, several model fusion techniques were applied, including simple averaging, weighted averaging, and logistic regression-based fusion. The results from the classical approach revealed that both KNN and SVM produced comparable performances, with 74% sensitivity, 72% specificity, and an AUC of 82% for KNN, and 80%, 73%, and 83% respectively for SVM. However, deep learning methods significantly outperformed the classical models, with EfficientNetB0 achieving an AUC of 88%, compared to 78% for the baseline CNN. Among the fusion approaches, weighted average fusion demonstrated the best overall performance, with a sensitivity of 79%, specificity of 81%, and an AUC of 88%. Logistic regression fusion, while highly effective at reducing false positives with a specificity of 94%, showed lower sensitivity at 50%. Simple averaging fusion yielded more modest results. These findings underscore the potential of model fusion techniques in boosting the accuracy of automated melanoma diagnosis systems, particularly when integrating deep learning models with classical approaches.

Keywords: melanoma, computer-aided diagnosis, KNN, SVM, CNN, EfficientNetB0, fusion

REMERCIEMENTS

Nous remercions tout d'abord DIEU Tout-Puissant pour ses bénédictions, la vie, la santé et la force qu'il nous a accordées tout au long de cette période d'étude.

Nous tenons également à exprimer notre profonde gratitude à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail, que ce soit par leur soutien moral, leur expertise, leur encouragement, ou leur précieuse collaboration. Mes pensées vont particulièrement à :

- Monsieur Rabah IGUERNAISSI notre encadrant au LIS pour sa patience, son orientation, ses conseils et sa disponibilité tout au long de ce projet.

- Madame Jilliana MONNIER notre deuxième encadrant au LIS pour son attention, ses remarques et conseils.

- Monsieur Lahcen AIT IBOUREK notre tuteur pédagogique pour tous ses conseils et notre formation.

- Mes amis et camarades de promotion pour leur soutien et encouragements.

- Mon chéri Arthur pour sa présence continue, ses conseils et encouragements.

- Ma maman Virginie, mon frère Franck et ma sœur Rose pour leur soutien.

Et finalement à tous ceux et celles que nous n'avons pas mentionnés, qu'ils trouvent ici l'expression notre profonde gratitude.

SOMMAIRE

LISTE DE FIGURES

LISTE DE TABLEAUX

ABBREVIATIONS

IA : Intelligence Artificielle

AMU : Aix-Marseille Université

CNRS : Centre National de la Recherche Scientifique

INSERM : Institut National de la Santé et de la Recherche Médicale

IRD : Institut de Recherche pour le Développement

P2RI : Pôle de Recherche intersectoriels et interdisciplinaires

LIS : Laboratoire d'Informatique et de Systèmes

ABCD : Asymmetry- Border- Color- Diameter

GLCM : Gray-Level Co-occurrence Matrix

RBF : Radial Basis Function

CNN : Convolutionnal Neural Network (Réseau de neurones convolutif)

KNN : K-Nearest Neighbors

SVM : Support Vector Machine

DT : Decision Tree (arbre de décision)

BT : Boost Tree (Arbre Boosté)

AUC : Area Under Curve

ROC : Receiver Operating Characteristic

INTRODUCTION

Le cancer de la peau représente un enjeu majeur de santé publique en raison de son incidence croissante et de son taux de mortalité élevé. Les types de tumeurs les plus fréquents de la peau sont le carcinome basocellulaire, le carcinome épidermoïde et le mélanome. Ces derniers temps, le type de tumeur cutanée le plus mortel qui apparaît fréquemment est le mélanome [1]. Ce type de cancer de la peau est responsable de 75 % des décès liés au cancer de la peau aux États-Unis [2]. Une détection précoce du mélanome peut considérablement améliorer le taux de survie des patients. Cependant, identifier les mélanomes à un stade précoce reste un défi pour les dermatologues en raison de la complexité de différenciation entre les lésions bénignes et malignes. Face à ces difficultés, l'utilisation de l'intelligence artificielle (IA) et des techniques d'apprentissage automatique offre des solutions prometteuses pour automatiser et améliorer le diagnostic.

C'est dans ce contexte que notre projet de stage vise à développer un système de classification pour l'aide au diagnostic de cancer de la peau notamment le mélanome. Nous avons exploré deux approches : une approche classique utilisant les classifieurs K-Nearest Neighbors (KNN) et Support Vector Machine (SVM) et une approche par apprentissage profond avec des réseaux de neurones convolutifs (CNN) et EfficientNetB0. En plus, nous avons implémenté des méthodes de fusion de modèles pour améliorer les performances globales en combinant les forces de chaque modèle.

Pour organiser notre travail, nous commencerons par une revue littéraire afin de comprendre les méthodes automatiques de détection du mélanome. Ensuite, nous développerons nos différents modèles. Ce rapport est structuré en trois chapitres : le premier présente le cadre et le contexte global de notre projet ; le deuxième décrit la méthodologie que nous avons mise en œuvre ; et le troisième expose les résultats obtenus ainsi que leur interprétation.

Chapitre 1 : Présentation du cadre du projet

1.1 Présentation de l'organisme d'accueil

1.1.1 À propos de l'Université d'Aix- Marseille

L'université d'Aix Marseille (AMU) est l'une des plus grandes universités francophones pluridisciplinaires, elle a été créée en 2012 à la suite de la fusion de trois universités de la région du sud de la France. Elle est une université de recherche intensive et est présente dans neuf villes et dans quatre départements de la région. Classée parmi les premières universités françaises au classement de Shangaï, elle est structurée autour de cinq secteurs disciplinaires tels que :

- Arts, Lettres, Langues et Sciences Humaines
- Droit et Science politique
- Economie et Gestion
- Santé
- Sciences et Technologies

Repartie sur 17 composantes notamment les facultés, instituts et écoles. Elle est reconnue pour son excellence en recherche grâce à ses nombreuses unités et fédérations de recherche, ainsi qu'à ses partenariats avec des organismes majeurs tels que le CNRS, l'Inserm, l'IRD et le CEA. Elle a développé une stratégie de recherche et d'innovation structurée autour de ses cinq pôles de recherche intersectoriels et interdisciplinaires (PR2I) tels que : l'Énergie, l'Environnement, Humanités, Santé & Sciences de la Vie, et Sciences & Technologies Avancées. Parmi ses unités de recherche figure le Laboratoire d'Informatique et Systèmes (LIS), où nous avons effectué notre stage.

1.1.2 Le LIS UMR 7020

Le Laboratoire d'Informatiques et des Systèmes (LIS) est une Unité Mixte de Recherche (UMR) placée sous la double tutelle du Centre National de la Recherche Scientifique via l'Institut des sciences informatiques et de leurs interactions, et des universités d'Aix-Marseille (AMU) et de Toulon (UTLN). Il regroupe des activités de recherche qui sont réparties sur les campus de Saint-Jérôme et Luminy à Marseille, ainsi que sur celui de l'Université de Toulon.

Il regroupe plus de 375 membres, dont 190 permanents, chercheurs et enseignants-chercheurs, ainsi que 20 personnels techniques et administratifs. Les travaux du LIS couvrent des recherches tant fondamentales qu'appliquées dans les domaines de l'informatique, de

L'automatique, ainsi que du traitement du signal et de l'image. Le laboratoire est organisé en 20 équipes de recherche, elles-mêmes regroupées en 4 pôles principaux telles que décrites dans la Figure 1 suivante :

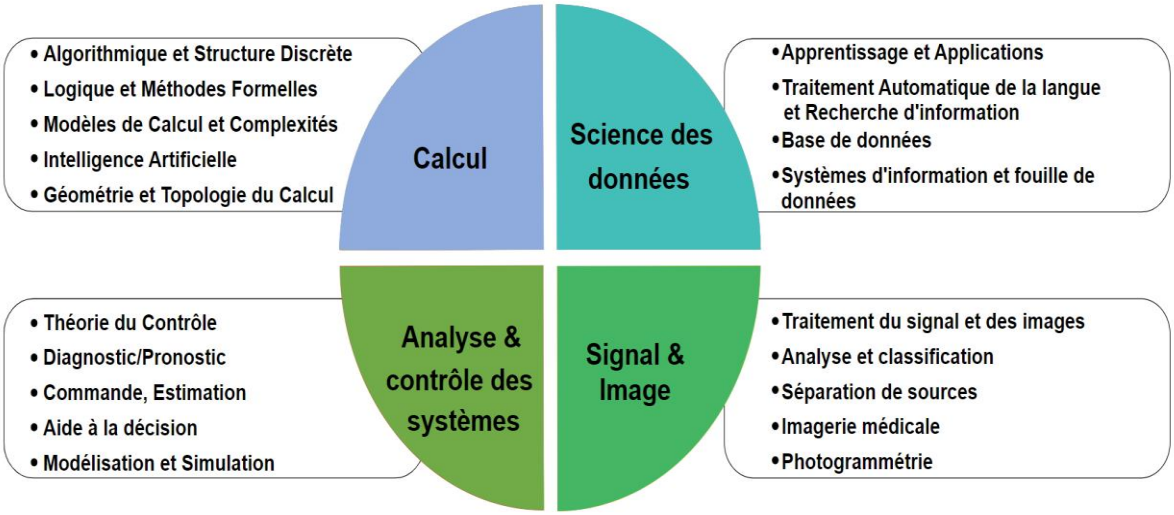

Figure 1:pôles de recherches du LIS

❖ **L'équipe**

Dans le cadre de notre stage, nous étions rattachés à l'équipe Image & Modèle constituée de 22 membres du pole Signal & Image donc les axes de recherches se focalise sur :

- L'aide au diagnostic
- Planification préopératoire
- Analyse comportementale
- Morphométrie
- Systèmes d'informations

1.2 Planification du projet (Diagramme de Gantt)

Notre stage s'est déroulé sur une période de huit semaines, durant lesquelles les différentes tâches ont été planifiées de manière structurée afin de respecter les délais et d'assurer une progression régulière du projet. Le diagramme de Gantt, illustré à la Figure 2 ci-dessous, présente la répartition des tâches et leur enchaînement tout au long de ces huit semaines.

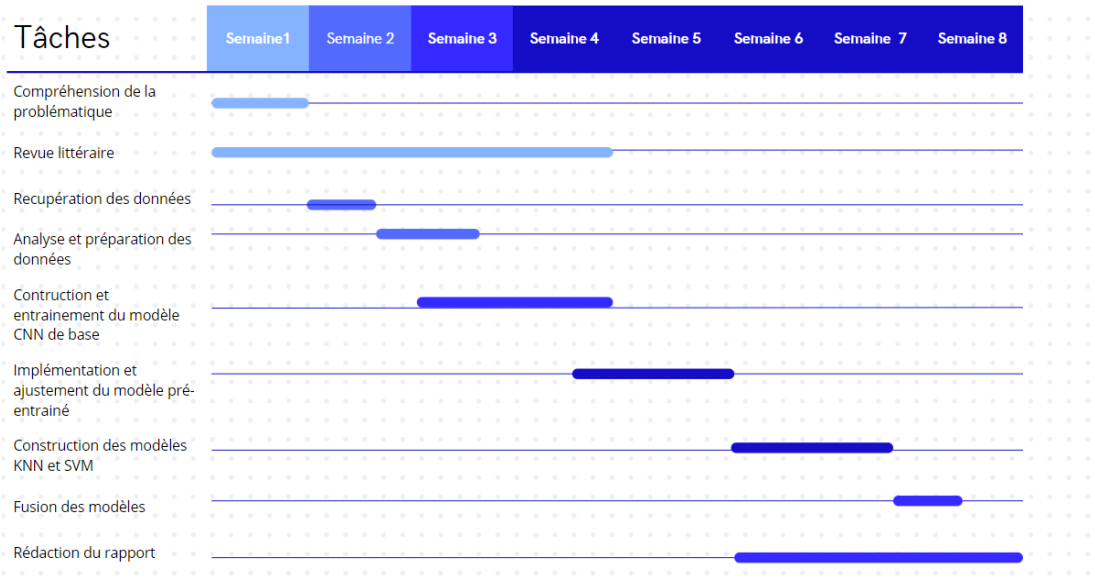

Figure 2:Diagramme de Gantt

1.3 Etat de l'art

La détection et la classification précoces des cancers cutanés, notamment du mélanome, sont des enjeux majeurs en dermatologie, permettant d'améliorer significativement les taux de survie. Les méthodes traditionnelles comme la biopsie, bien que précises, sont invasives et peuvent entraîner des retards dans le diagnostic. Ces limitations ont conduit au développement de méthodes non invasives utilisant des techniques d'intelligence artificielle (IA) et de traitement d'images. Cette section présente une revue des méthodes automatiques de détection et de classification des lésions cutanées, en mettant en lumière les approches, les algorithmes, et les résultats obtenus dans la littérature récente.

Alenezi [3] a étudié les limites des méthodes traditionnelles, comme la biopsie, pour la détection du mélanome. Il a exploré l'utilisation des CNNs et des SVMs comme alternatives non invasives et précises. Son étude a appliqué des modèles CNNs tels qu'AlexNet, LeNet et VGG-16 sur des images dermoscopiques, tout en utilisant un classificateur SVM avec un noyau RBF pour la classification. Les résultats montrent que le modèle CNN a obtenu une précision de 94 %, avec une perte de 17 % sur le jeu de données de test, tandis que le SVM a affiché une précision de 86,6 %.

Akila Victor [4] a réalisé une étude axée sur la classification des lésions cutanées, mettant l'accent sur la différenciation entre les formes bénignes et malignes. Pour ce faire, il a utilisé des techniques de traitement d'images en combinaison avec des algorithmes d'apprentissage automatique. Le processus de classification débute par un pré-traitement des images, qui vise à éliminer le bruit à l'aide d'un filtre médian, suivi d'une amélioration de la qualité des images par l'égalisation de l'histogramme. La segmentation est ensuite effectuée grâce à la méthode de seuillage d'Otsu [5], permettant de distinguer la région d'intérêt de l'arrière-plan. Les caractéristiques extraites, telles que la surface, la moyenne, la variance et l'écart-type des images

segmentées, sont utilisées pour la classification à l'aide de différents algorithmes : SVM, KNN, arbre de décision (DT) et arbre boosté (BT). Les résultats indiquent que le SVM a atteint une précision de 93,70 %, suivi du KNN avec 92,70 %, de l'arbre de décision avec 89,50 %, et de l'arbre boosté avec 84,30 % [3].

Une autre étude s'est focalisée sur la détection des mélanomes à partir d'images de peau, en utilisant divers algorithmes de classification, notamment SVM, Random Forest et KNN [6]. Ce processus commence par un pré-traitement avec un filtre médian pour réduire le bruit, suivi d'une segmentation par la méthode de bassin versant. Les caractéristiques extraites comprennent des paramètres basés sur la règle ABCD [7] et la matrice de cooccurrence des niveaux de gris (GLCM), ainsi que des caractéristiques de forme et la distance de la lésion. Les résultats de la classification révèlent que, pour la règle ABCD, le SVM a atteint une précision de 89,43 %, une sensibilité de 91,15 %, et une spécificité de 87,71 %, surpassant ainsi les performances du Random Forest (qui sont de : 76.87%, 78.43%, 75.31% respectivement) et du KNN (69.54%, 71.32%, 67.76%). Des résultats similaires ont été observés pour les caractéristiques GLCM et de forme, avec le SVM se distinguant comme l'algorithme le plus performant parmi ceux testés avec des performances de 85.72%, 87.68%, 83.76%.

D’autres chercheurs ont cherché à améliorer cette détection en en combinant des méthodes traditionnelles de traitement d'image avec l'apprentissage profond dans un modèle fusionné [8]. Ils ont utilisé trois modules manuels pour identifier les caractéristiques des lésions cutanées, comme le réseau pigmentaire atypique, la distribution des couleurs et les vaisseaux sanguins, tout en tenant compte de données cliniques comme l'âge, le sexe et l'emplacement de la lésion. En parallèle, un réseau de neurones ResNet-50 a été employé pour extraire des caractéristiques via l'apprentissage profond. En fusionnant ces deux approches à l'aide d'une régression logistique, ils ont atteint une aire sous la courbe ROC (AUC) de 0,94, surpassant les performances de chaque modèle pris séparément (0,87 pour ResNet-50 seul et 0,90 pour le traitement d'image manuel).

Chapitre2 : Méthodologie

De nombreuses recherches ont été menées sur la conception de systèmes automatiques pour l'analyse et la classification des lésions cutanées. Dans ce travail, nous avons exploré deux approches, dont une méthode classique utilisant un algorithme KNN, un algorithme SVM et une approche par apprentissage profond avec des CNN. L'objectif principal est de développer trois modèles distincts : les algorithmes KNN et SVM basés sur l'extraction de caractéristiques spécifiques, un CNN conçu « from scratch », ainsi qu'un modèle EfficientNetB0 pré-entraîné. Chaque modèle sera entraîné et évalué individuellement, avant que leurs probabilités de classification ne soient fusionnées pour améliorer les performances globales, la Figure 3ci-dessous illustre le processus général de classification par les différents modèles. Ce chapitre détaille les étapes du processus, allant de la préparation des données au développement et à la validation des modèles, ainsi qu'à l'évaluation des résultats obtenus par l'approche de fusion.

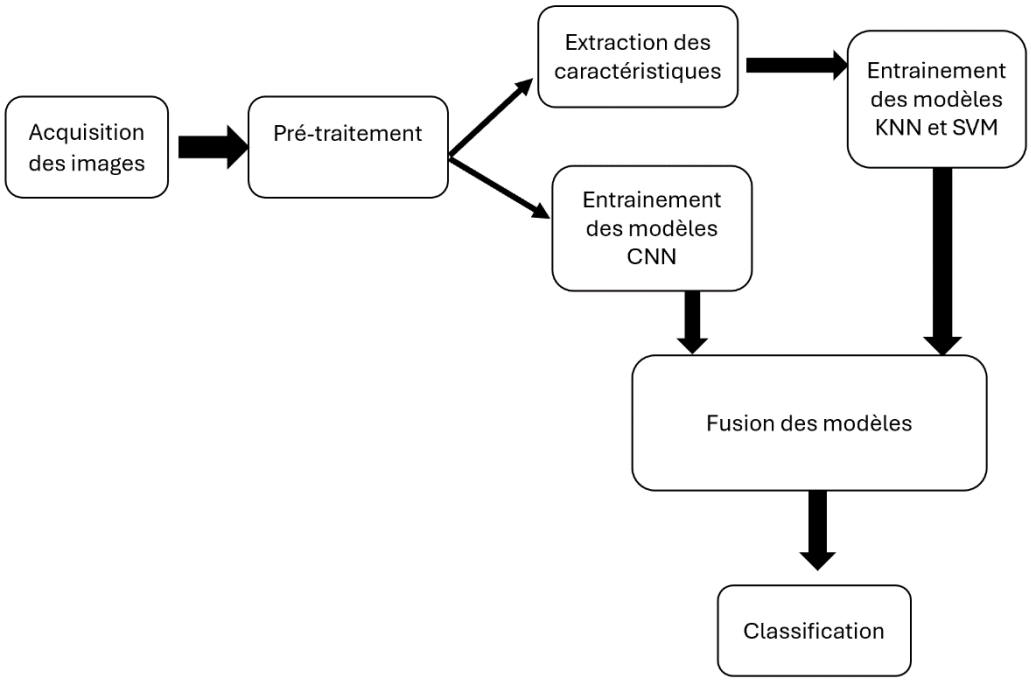

Figure 3: schéma global

2.1 Matériels et outils utilisés

Dans le cadre de notre projet nous avons utilisés plusieurs technologies et outils notamment un environnement de développement et des bibliothèques tels que :

- **Python :**

  Pour l'ensemble du projet nous avons utilisé ce langage de programmation, en raison de sa polyvalence et de la richesse de ses bibliothèques dédiées au traitement d'images et à l'apprentissage automatique.

Figure 4:Python

- **Spyder :** Nous avons utilisé cet environnement de développement intégré (IDE) pour écrire, exécuter et déboguer le code.

- **NumPy** : Utilisée pour les opérations mathématiques et le traitement des données sous forme de tableaux multidimensionnels.

Figure 5:Numpy

- **Pandas** : c'est une bibliothèque Python qui permet l'analyse et la manipulation des données, nous l'avons utilisée notamment pour la gestion des fichiers CSV contenant les caractéristiques extraites et les résultats de classification des modèles.

Figure 6:Pandas

- **Matplotlib** et **Seaborn** : Ces bibliothèques ont servi à la visualisation des données, permettant de créer des graphiques et des courbes illustrant les performances des modèles.

(a) 

(b) 

Figure 7:(a) MatPlotlib, (b) Seaborn,

- **Scikit-learn** : Nous l'avons utilisé principalement pour les modèles de l'approche classique tels que le KNN et le SVM, ainsi que pour les tâches de prétraitement des données et d'évaluation des performances.

Figure 8:Scikit-learn

- **TensorFlow** et **Keras** : Utilisées pour la conception, l'entraînement et l'évaluation des modèles de réseaux de neurones convolutifs (CNN) et pour l'implémentation du modèle pré-entraîné EfficientNetB0.

Figure 9:TensorFlow, Keras

2.2 Base de données utilisée

2.2.1 Description de la base

La base de données utilisée dans cette étude provient du programme ISIC. Elle se compose d'images non segmentées dermoscopiques de lésions cutanées, comprenant des cas de

mélanomes et de nævus. Cette base est divisée en deux ensembles : un ensemble d'entraînement de 4900 images réparties entre les deux classes « MEL » et « NEV », et un ensemble de test de 1533 images représentant également ces deux classes.

Dataset	Nombre d'images de la classe MEL	Nombre d'images de la classe NEV
Base d'entrainement	1088	5036
Base de Test	273	1260

Tableau 1:répartition de la base de données

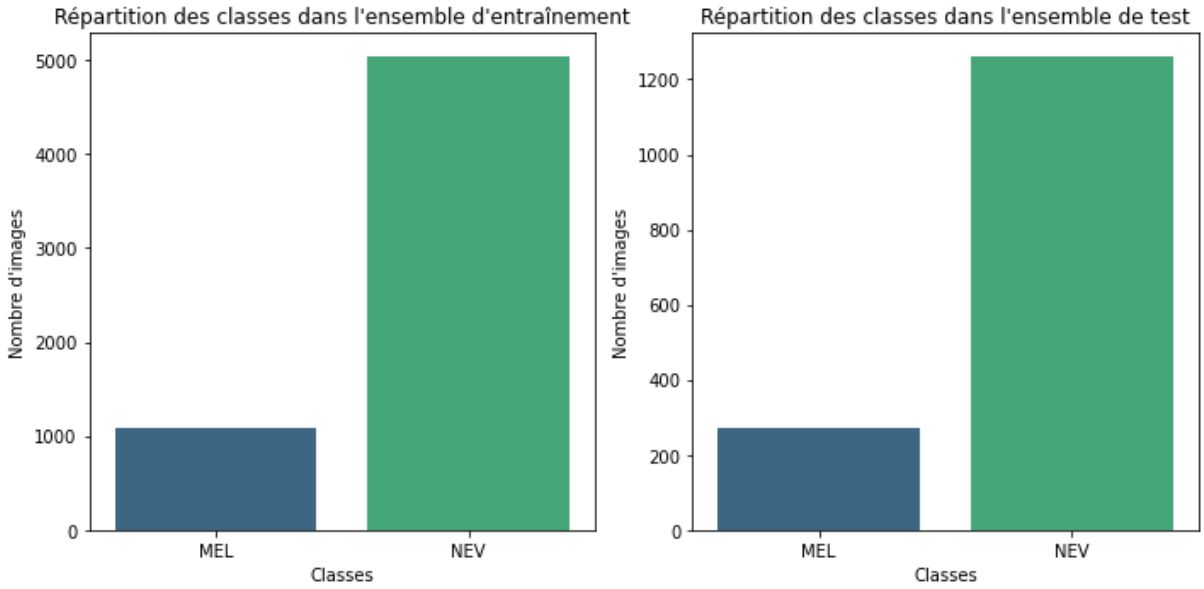

Figure 10: répartition des classes dans les bases de test et d'entrainement

Les images de la base sont au format de couleur RGB et possèdent des dimensions variées. La Figure 11 suivante illustre quelques exemples annotés tirés de cette base de données.

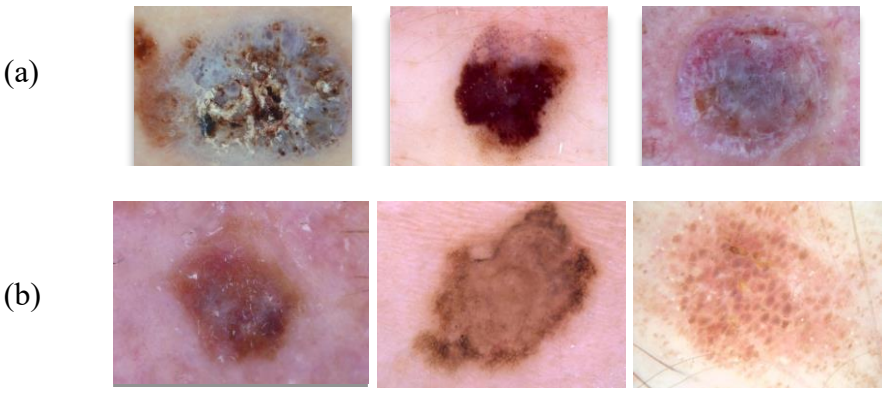

Figure 11: exemples d'images de la base de données (a) Mélanome, (b) Naevus

2.3 Approche classique

2.3.1 Développement des modèles KNN et SVM

2.3.1.1 Extraction des caractéristiques

Cette étape indispensable pour la classification, elle permet de transformer les images brutes en un ensemble de caractéristiques numériques qui capturent l'essence de l'information contenue dans les images, facilitant ainsi la discrimination entre les classes. Dans notre étude, pour chaque image, des caractéristiques préalablement calculées notamment l'entropie, l'écart-type, le kurtosis, et le skewness telles que illustrées dans la Figure 12 ci-dessous ont été extraites à partir de ses composantes RGB (Rouge, Vert, Bleu) et HSV (Teinte, Saturation, Valeur) afin de capturer des variations subtiles de teinte, de saturation et de luminosité qui pourraient être pertinentes pour la classification.

- Entropie : Mesure de l'incertitude ou du désordre dans la distribution des valeurs des pixels.
- Écart-type : Mesure de la dispersion des valeurs des pixels autour de la moyenne.
- Kurtosis : Mesure de la concentration des valeurs des pixels par rapport à la moyenne, indiquant la présence d'éventuelles valeurs extrêmes.
- Skewness : Mesure de l'asymétrie de la distribution des valeurs des pixels.

Ces caractéristiques seront stockées dans un fichier CSV qui servira d'entrée pour l'entrainement que ce soit du modèle KNN ou le SVM.

```python
def calculate_features(self, img, img_hsv):
      features = {}
      #  Pour les composants RGB
      for i, color in enumerate(['R', 'G', 'B']):
          channel = img[:, :, i]
          features[f'{color}_entropy'] = entropy(channel.ravel())
          features[f'{color}_std'] = np.std(channel)
          features[f'{color}_kurtosis'] = kurtosis(channel.ravel(), fisher=False)
          features[f'{color}_skewness'] = skew(channel.ravel())


      # Pour les composants HSV
      for i, color in enumerate(['H', 'S', 'V']):
          channel = img_hsv[:, :, i]
          features[f'{color}_entropy'] = entropy(channel.ravel())
          features[f'{color}_std'] = np.std(channel)
          features[f'{color}_kurtosis'] = kurtosis(channel.ravel(), fisher=False)
          features[f'{color}_skewness'] = skew(channel.ravel())



      return features

def extract_features(self, image_dir_hsv, label):
     image_paths = glob.glob(image_dir_hsv + '/*/*.jpg')
     features_list = []

     for path in image_paths:
         print(f"Extracting features from {path}")  # vérifier les chemins des images converties
         img_hsv = cv2.imread(path)
         if img_hsv is not None:
             features = self.calculate_features(img_hsv, img_hsv)
             features['image_path'] = path
             features['nom'] = path.split("\\")[-1]
             features['label'] = label
             features_list.append(features)
         else:
             print(f"Impossible de charger l'image : {path}")

     return features_list
```

Figure 12: processus d'extraction des caractéristiques

2.3.1.2 Entrainement des modèles

Afin d'optimiser l'entrainement, nous avons implémenté des techniques pour mieux préparer les données notamment :

- Le calcul du rapport de classe afin de déterminer le poids de chacune des deux classes (MEL et NEV) pour pouvoir les ajuster du fait du déséquilibre afin que les modèles apprennent bien les deux lésions.

- L'équilibrage des données par la méthode *SMOTE (Synthetic Minority Over-sampling Technique)* de la librairie **IMBALANCED-LEARN** qui génère des exemples supplémentaires de la classe minoritaire (mélanomes) en créant des points synthétiques, ce qui permet d'obtenir un ensemble de données équilibré.

- La normalisation des données par *StandarScaler() de la* bibliothèque **scikit-learn** ce qui permet de centrer les caractéristiques autour de leur moyenne afin qu'elles soient à une même échelle.

Après cette préparation des données, nous avons entrainé les modèles :

- Le modèle KNN a été entrainé en utilisant un nombre de voisins que nous avons fixés (n=100) il évaluera la classe d'une image en fonction des classes des images les plus proches.

- Le modèle SVM lui a été entrainé en utilisant le paramètre noyau RBF car il permet de capturer les relations complexes et non linéaire des caractéristiques entre les deux classes ce qui permettra au classifieur de séparer les efficacement.

2.4 Approche par apprentissage profond

2.4.1 Pré-traitement des images

Nous avons conçu une méthode spécifique pour préparer les données avant l'entraînement et la validation des modèles par apprentissage profond. Les images sont d'abord redimensionnées à la taille 224x224 pixels pour garantir une uniformité dans les données d'entrée. Ensuite, des transformations d'augmentation par la classe *ImageDataGenerator* de la librairie **Keras** telles que le cisaillement, le zoom, le retournement horizontal, et la rotation sont appliquées pour enrichir la base de données et améliorer la robustesse des modèles. Les données sont divisées en ensembles d'entraînement, de validation et de test, avec une fraction des données d'entraînement réservée à la validation.

2.4.2 Développement du modèle CNN

2.4.2.1 Qu'est-ce qu'un CNN ?

Un CNN est un algorithme d'apprentissage profond utilisé pour traiter les images, Il fonctionne en passant une image à travers une série de filtres pour extraire progressivement des motifs complexes à travers des couches convolutives suivies de couches de « pooling », qui réduisent la dimensionnalité tout en conservant les caractéristiques essentielles. Ces informations sont ensuite aplaties en un vecteur par une couche de « flattening », puis transmises à une couche entièrement connectée qui permet de classer les images dans différentes catégories. La Figure 13 ci-dessous illustre le schéma d'un CNN adapté à notre projet.

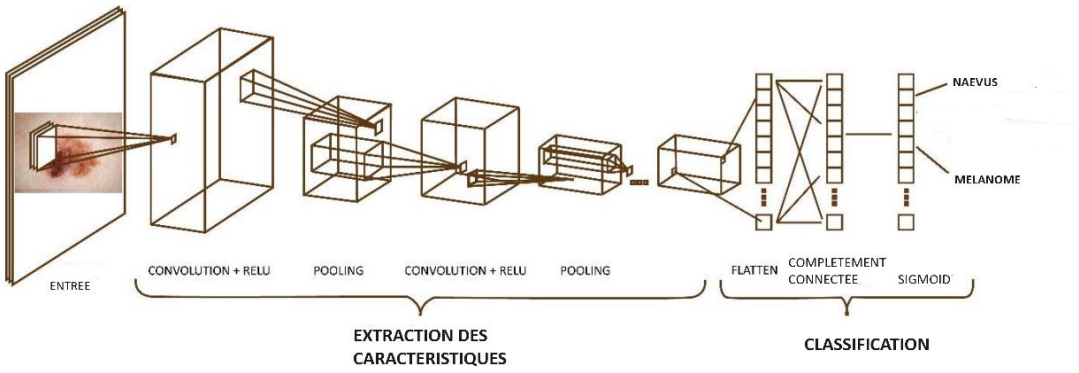

Figure 13:schéma illustratif d'un CNN

2.4.2.2 Architecture du modèle

Pendant notre étude, nous avons testé différents nombres de couches et tailles d'entrée des images nous avons conservé comme architecture de base celle décrite dans le Tableau 2ci-dessous

Nom de la couche(type)	**Taille de sortie**	**Paramètres**
Conv2d_5 (Conv2D)	(None, 128, 128, 16)	448 Activation= Relu
Batch_normalization_6 (BatchNormalization)	(None, 128, 128, 16)	64
Max_pooling2d_5 (MaxPooling2D)	(None, 64, 64, 16)	0
conv2d_6 (Conv2D)	(None, 64, 64, 32)	4,640 Activation= Relu
batch_normalization_7 (BatchNormalization)	(None, 64, 64, 32)	128
max_pooling2d_6 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_7 (Conv2D)	(None, 32, 32, 64)	18,496 Activation= Relu
batch_normalization_8 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d_7 (MaxPooling2D)	(None, 16, 16, 64)	0

conv2d_8 (Conv2D)	(None, 16, 16, 128)	73,856 Activation= Relu
batch_normalization_9 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 8, 8, 128)	0
conv2d_9 (Conv2D)	(None, 8, 8, 256)	295,168 Activation= Relu
batch_normalization_10 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_9 (MaxPooling2D)	(None, 4, 4, 256)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_4 (Dense)	(None, 64)	262,208 Activation= Relu
dropout_4 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 1)	65 Activation= Sigmoïde
Paramètres entrainables : **Paramètres non-entrainables** **Total paramètres :**	**655,873** **992** **656,865**	

Tableau 2: Architecture de base du modèle CNN

2.4.2.3 Ajustement des hyperparamètres

Dans notre étude, nous avons constaté que certains hyperparamètres influent grandement sur la capacité de notre modèle à apprendre, c'est pourquoi un ajustement optimal est indispensable pour la bonne performance du modèle, nous avons notamment :

-Réduit le nombre de couches convolutives à 5 car un modèle trop long met plus de temps à apprendre

- Varié le nombre de filtres par couche, en commençant par 16 pour la première couche, 32 pour la deuxième, puis en doublant progressivement jusqu'à atteindre 256 filtres dans la cinquième couche.

- Nous avons également appliqué un « padding » [1]afin de préserver les dimensions des images à chaque étape de la convolution, ce qui permet de simplifier l'architecture du modèle et d'en faciliter l'apprentissage.

2.4.2.4 Entrainement du modèle

L'entraînement consiste à présenter au modèle les données d'entraînement à travers plusieurs itérations regroupées en époques, afin qu'il apprenne à différencier les caractéristiques des lésions. Au cours de cet apprentissage, le modèle ajuste progressivement ses poids en minimisant l'erreur entre ses prédictions et les valeurs réelles ce qui lui permet de généraliser correctement et d'obtenir des performances optimales sur les nouvelles images. La Figure 14 ci-dessous illustre les méthodes et configurations utilisées pour entraîner notre modèle, Nous avons calculé des poids de classes pour gérer le déséquilibre entre les classes de mélanomes et de nævus. Ensuite, le modèle a été compilé avec un optimiseur Adam, un taux d'apprentissage (Learning rate =1%) et une fonction de perte de type entropie croisée binaire. De plus, des mécanismes de régulation, tels que l'« Early Stopping » et la réduction du taux d'apprentissage sur plateau, ont été mis en place pour éviter le surajustement et améliorer l'efficacité modèle.

```
def train(self, model_build, training_set, validation_set, numb_epochs):
    # Calculer les poids des classes pour le jeu d'entraînement
    class_labels = training_set.classes
    class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(class_labels), y=class_labels)
    class_weights = dict(enumerate(class_weights))

    # Callbacks
    early_stopping = EarlyStopping(monitor='auc', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=0.0001)
    model_checkpoint = ModelCheckpoint('best_model.keras', save_best_only=True, monitor='val_auc', mode='max')



    # Compiler le modèle avec les poids de classes

    optimizer = Adam(learning_rate=0.01)
    model_build.compile(optimizer=optimizer,
                loss='binary_crossentropy',
                metrics=['accuracy', AUC(name='auc')])

    # Entraîner le modèle avec les poids de classes
    history = model_build.fit(
        training_set,
        epochs=numb_epochs,
        validation_data=validation_set,
        class_weight=class_weights,
        callbacks=[early_stopping, reduce_lr, model_checkpoint]
    )


    return model_build, history
```

[1] Le padding est une technique en apprentissage profond qui consiste à ajouter des pixels autour des bords d'une image pour préserver ses dimensions lors des opérations de convolution

Figure 14:Methode pour entrainer le modèle

2.4.3 Implémentation du modèle EfficientNetB0

2.4.3.1 Architecture

Nous avons conservé l'architecture de base du modèle EfficientNetB0 avec ses poids pré entrainés sur ImageNet, ensuite nous avons apporté une légère modification pour avoir une architecture optimale adapté à notre base de données. Nous avons dans un premier temps initialisé le modèle sans sa dernière couche afin de pouvoir ajouter des couches pour l'ajuster à notre tâche, le tableau ci-dessous décrit les couches ajoutées sur le modèle.

Nom de la couche(type)	**Taille de sortie**	**Paramètres**
avg_pool(GlobalAveragePooling2D	128	0
batch_normalization(BatchNormalization)	128	512
top_dropout(Dropout)	1280	0 Taux = 0.2
pred(Dense)	1	1281 Activation = sigmoid

Tableau 3:paramètres des couches ajoutées au modèle EfficentNetB0

2.4.3.2 Ajustement des hyperparamètres

Pour optimiser les performances du modèle pré-entrainé EfficientNetB0 nous avons ajuster certains paramètres notamment :

- Premièrement, nous avons **geler les couches** en initialisant le modèle pour lui permettre d'utiliser les caractéristiques générales basiques telles que les bordures, texture, formes etc... Qu'il a déjà apprises sans toutefois modifier les poids ce qui pourra stabiliser ces caractéristiques lors de l'entrainement initial sur notre propre jeu de données.

- Ensuite nous avons **partiellement dégeler les derniers couches** (18couches) pour pouvoir adapter le modèle à notre jeu de données en lui permettent de réapprendre des caractéristiques plus fines qui sont pertinentes à notre classification(**fine-tunning**) tout ceci sans perturber les caractéristiques basiques déjà apprises.

2.4.3.3 Entrainement

L'entrainement de ce modèle s'est fait en deux parties ; premièrement, l'entrainement initial avec les couches gelées et par la suite entrainement avec les couches dégelées ; nous avons appliqué les mêmes paramètres que dans le modèle CNN et les mêmes mécanismes de régulation. La Figure 15ci-dessous illustre les méthodes pour l'entrainement du modèle.

```python
#Degeler une partie des couches
def unfreeze_model(self, model):
# on degele les 18 dernieres couches sauf les couches de bacthnormlisation
    for layer in model.layers[-18:]:
        if not isinstance(layer, layers.BatchNormalization):
            layer.trainable = True

    optimizer = keras.optimizers.Adam(learning_rate=1e-5)
    model.compile(
        optimizer=optimizer, loss="binary_crossentropy", metrics=["accuracy", AUC(name='auc')]
    )

    return model


def train(self, model_build, training_set, validation_set, numb_epochs):
    class_labels = training_set.classes
    class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(class_labels), y=class_labels)
    class_weights = dict(enumerate(class_weights))

    early_stopping = EarlyStopping(monitor='val_auc', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=0.0001)
    model_checkpoint = ModelCheckpoint('best_model.keras', save_best_only=True, monitor='val_auc', mode='max')


    history = model_build.fit(
        training_set,
        epochs=numb_epochs,
        validation_data=validation_set,
        class_weight=class_weights,
        callbacks=[early_stopping, reduce_lr, model_checkpoint]
    )

    return model_build, history
```

Figure 15: illustration des méthodes pour l'entrainement du modèle EffecientNetB0

2.5 Fusion des modèles

Après avoir évalué séparément les différents modèles de chaque approche, nous avons abordé cette technique de fusion pour pouvoir tirer parti de leurs forces respectives et à compenser leurs faiblesses tout ceci dans le but d'améliorer les performances globales de notre système de classification des lésions mélanome et naevus. Premièrement nous avons récupérer ces probabilités que nous avons stocker dans des fichiers Excel avec le nom des images et de la classes vraies et prédites correspondantes ensuite faire fusionner tous les fichiers dans un seul qui correspondra à celui de la fusion

Nous avons notamment testé trois méthodes de fusion à savoir la fusion par moyenne, par moyenne pondérée et la fusion par régression logistique.

2.5.1 Fusion par moyenne

Nous avons dans un premier temps testé cette méthode qui vise à simplement de faire une moyenne des probabilités des différents modèles telle que décrite par l'équation (1)

$$P_{finale} = \frac{1}{N}\sum_{i=1}^{N} P_i(M_i) \qquad (1)$$

Où P_i est la probabilité prédite par le modèle M_i et N le nombre de modèles.

2.5.2 Fusion par moyenne pondérée

Nous avons remarqué dans la fusion par moyenne que les performances de chaque modèle ne sont pas vraiment prises en compte, donc les modèles moins performants auront tendance à baisser les résultats. Nous avons exploré la moyenne pondérée car elle améliore la fusion par moyenne en ce sens qu'elle attribue des poids différents aux modèles en fonction de leurs performances donc les modèles plus performants influenceront davantage sur les résultats finaux. Ici les probabilités P_i prédites par chaque modèle sont multipliées par un poids w_i associé soit manuellement soit par validation croisée et la somme pondérée des probabilités est calculée telle que décrit dans l'équation (2) ci-dessous :

$$P_{finale} = \frac{\sum_{i=1}^{N} w_i * P_i}{\sum_{i=1}^{N} w_i} \qquad (2)$$

2.5.3 Fusion par régression Logistique

La régression logistique est une méthode statistique utilisée pour les problèmes de classification binaire. Elle vise à prédire la probabilité qu'un événement appartienne à l'une des deux catégories possibles. Dans notre contexte de la détection des lésions cutanées, elle est employée pour déterminer si une lésion est maligne (mélanome) ou non (naevus) en utilisant la fonction logistique(sigmoïde) pour modéliser cette probabilité. Cette fonction transforme les valeurs réelles en une probabilité comprise entre 0 et 1, facilitant ainsi la classification des observations en deux classes distinctes.

La fusion par régression logistique vise donc à combiner ces différentes probabilités prédites par chacun des modèles en pondérant leur contribution en fonction de leur pertinence et performance individuelle pour pouvoir obtenir une probabilité finale plus fiable pour chaque classe. Cette probabilité est donnée par l'équation (3).

$$P(y = 1|X) = \frac{1}{1+exp\,(-(\beta_0+\beta_1 p_1+\beta_2 p_2+\cdots\beta_n p_n))} \qquad (3)$$

Où P(y=1|X) est la probabilité que la lésion soit un mélanome (classe 1) donnée par la régression logistique,

X représente les probabilités prédites par les différents modèles.

β_0 Le terme d'interception, β_1 , β_2 , β_n les coefficients associés à chaque modèle, appris durant l'entraînement de la régression logistique et l'exponentielle, qui permet de transformer la combinaison linéaire en une probabilité dans l'intervalle [0,1].

Chapitre 3 : Evaluation et résultats des modèles

3.1 Evaluation des modèles

Nous avons vu dans la littérature plusieurs critères d'évaluation des modèles de détection automatique du cancer de la peau, dans notre projet, nous prendrons trois critères jugés pertinentes du point de vue du dermatologue notamment la sensibilité, la specificité et la courbe ROC AUC (Receiver Operating Characteristic - Area Under the Curve).

❖ La sensibilité (SE) : C'est la capacité du modèle à identifier correctement les lésions malignes (taux de vrai positif). Sa formule est donnée par l'équation (1).

$$SE = \frac{\text{VP}}{\text{VP+FN}} \qquad (1)$$

❖ La spécificité (SP) : C'est la capacité du modèle à identifier correctement les lésions bénignes (taux de vrai négatif). Sa formule est donnée par l'équation (2).

$$SP = \frac{\text{VN}}{\text{VN+FP}} \qquad (2)$$

❖ Le ROC AUC : c'est une métrique qui évalue la performance d'un modèle de classification en mesurant sa capacité à distinguer entre les classes positives et négatives. Elle combine la courbe ROC, qui trace le taux de vrais positifs contre le taux de faux positifs pour différents seuils, avec l'aire sous cette courbe (AUC) qui indique la capacité de discrimination du modèle.

3.2 Résultats des modèles et discussion

3.2.1 Approche classique

Après avoir ajuster tous les hyperparamètres des modèles, nous avons obtenu les résultats sur l'ensemble des données de test tels que décrit dans le Tableau 4 ci-dessous

Modèle	Sensibilité	Spécificité	AUC
KNN	74%	72%	**82%**
SVM	80%	73%	**83%**

Tableau 4:Résultats des modèles de l'approche classique

Nous observons que, globalement, les valeurs d'AUC des deux modèles sont très proches, ce qui révèle une capacité similaire à différencier les mélanomes des nævus. En plus des mesures de sensibilité et de spécificité, la

Figure 16 ci-dessous montre les courbes ROC des deux modèles. On peut y voir que la courbe du SVM est légèrement au-dessus de celle du KNN, ce qui confirme leur similarité en termes de performance.

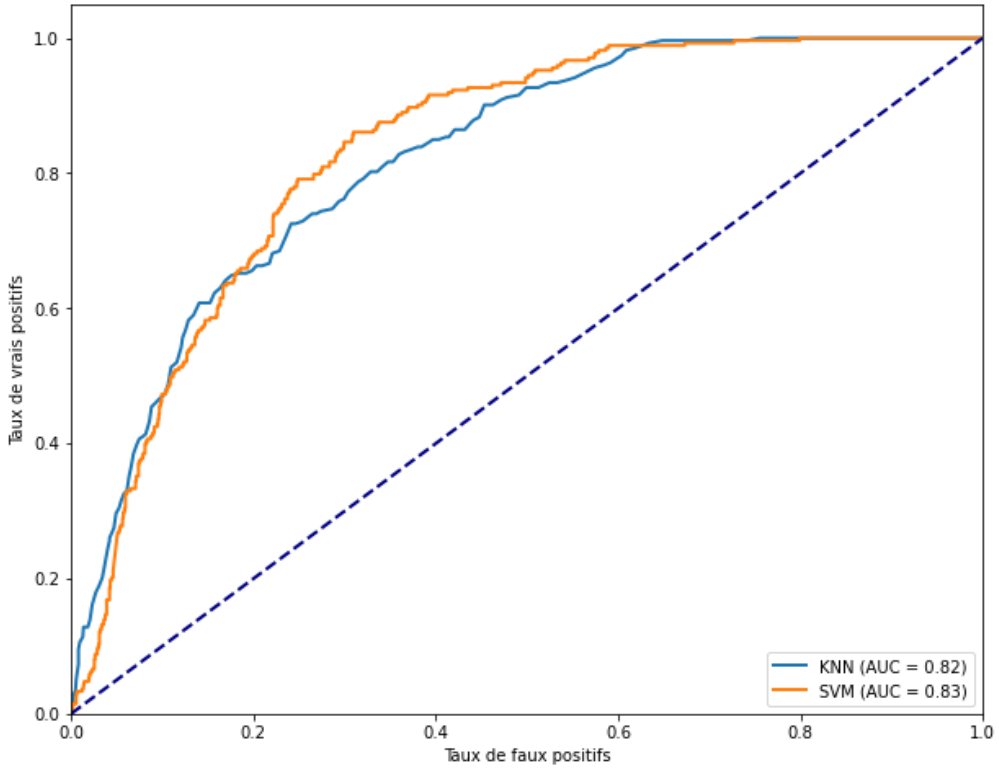

Figure 16:Courbes ROC des modèles classiques

3.2.2 Approche par apprentissage profond

Nous observons que le modèle CNN de base est très performant pour détecter les mélanomes avec une sensibilité de 98%. Cependant, sa spécificité est plutôt faible à 41%, ce qui signifie qu'il a tendance à confondre de nombreux nævus avec des mélanomes. En comparaison, le modèle EfficientNetB0 offre un meilleur équilibre : il détecte les mélanomes avec une sensibilité de 88% tout en étant plus précis pour identifier les nævus, avec une spécificité de 73%. Le score AUC du modèle EfficientNetB0 est également plus élevé, atteignant 89%, ce qui indique une meilleure performance globale pour distinguer les deux types de lésions. Ces résultats sont présentés dans le Tableau 5 et illustrés dans la Figure 17 ci-dessous. On y voit que l'aire sous la courbe ROC de EfficientNetB0 est plus grand que celle du CNN de base, ce qui confirme son avantage en termes de précision.

Modèle	Sensibilité	Spécificité	AUC
CNN de base	**98**%	41%	77%
EfficientNetB0	88%	**73**%	**88**%

Tableau 5:Resultats des modèles de l'approche par apprentissage profond

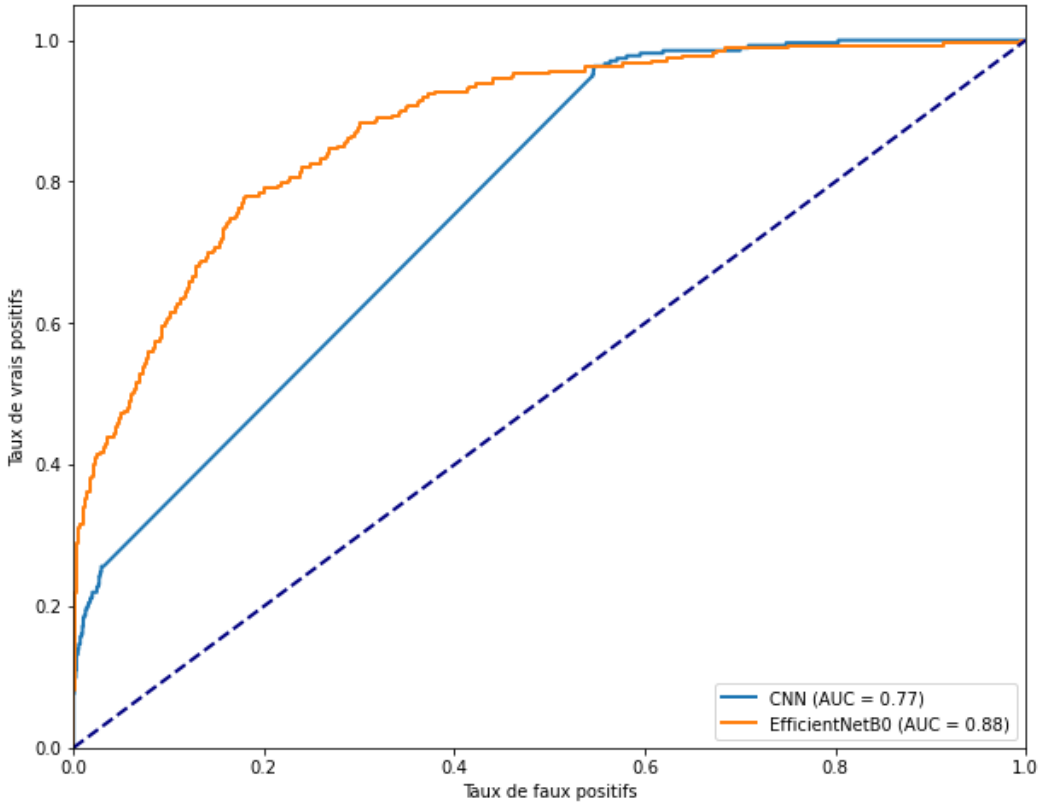

Figure 17:Courbes ROC des modèles profonds

3.2.3 Fusion des modèles

En analysant les performances individuelles des modèles telles que présentées précédemment, nous avons observé que celles du modèle CNN de base étaient nettement inférieures à celles des autres modèles. Malgré plusieurs ajustements des hyper paramètres, il n'a pas réussi à atteindre des résultats proches des autres modèles en termes de spécificité (41%) et de score AUC qui était également en dessous de celui des autres modèles (73%). Par conséquent, pour maximiser l'efficacité de la classification, nous avons donc décidé de fusionner uniquement les modèles KNN, EfficientNetB0, et SVM car intégrer le modèle CNN de base dans la fusion aurait pu nuire aux performances globales du système en introduisant des prédictions moins fiables. Le Tableau 6 ci-dessous présente les résultats obtenus par les différentes méthodes de fusion des modèles.

Méthode de fusion	Sensibilité	Spécificité	AUC
Fusion par moyenne	44%	76%	73%
Fusion par moyenne pondérée	**79%**	81%	88%
Fusion par Régression Logistique	50%	**94%**	**88%**

Tableau 6: Résultats de l'approche par fusion

La méthode de fusion par moyenne a montré des résultats plutôt modestes, avec une sensibilité de 44% et une spécificité de 76%. Cela signifie qu'elle détecte les mélanomes de manière limitée tout en restant assez bonne pour éviter les faux positifs. En revanche, la fusion par moyenne pondérée a significativement amélioré les résultats, atteignant une sensibilité de 79% et une spécificité de 81%, avec un AUC de 88%. Cette approche semble offrir un meilleur équilibre entre la détection des mélanomes et la réduction des faux positifs. La fusion par régression logistique, quant à elle, a obtenu une sensibilité de 50% et une spécificité très élevée de 94%, tout en affichant également un AUC de 88%. Bien qu'elle soit très efficace pour éviter les faux positifs, sa sensibilité un peu plus faible par rapport à la moyenne pondérée indique qu'elle pourrait être un peu moins réactive pour détecter les mélanomes. La Figure 18 ci-dessous illustre ces résultats.

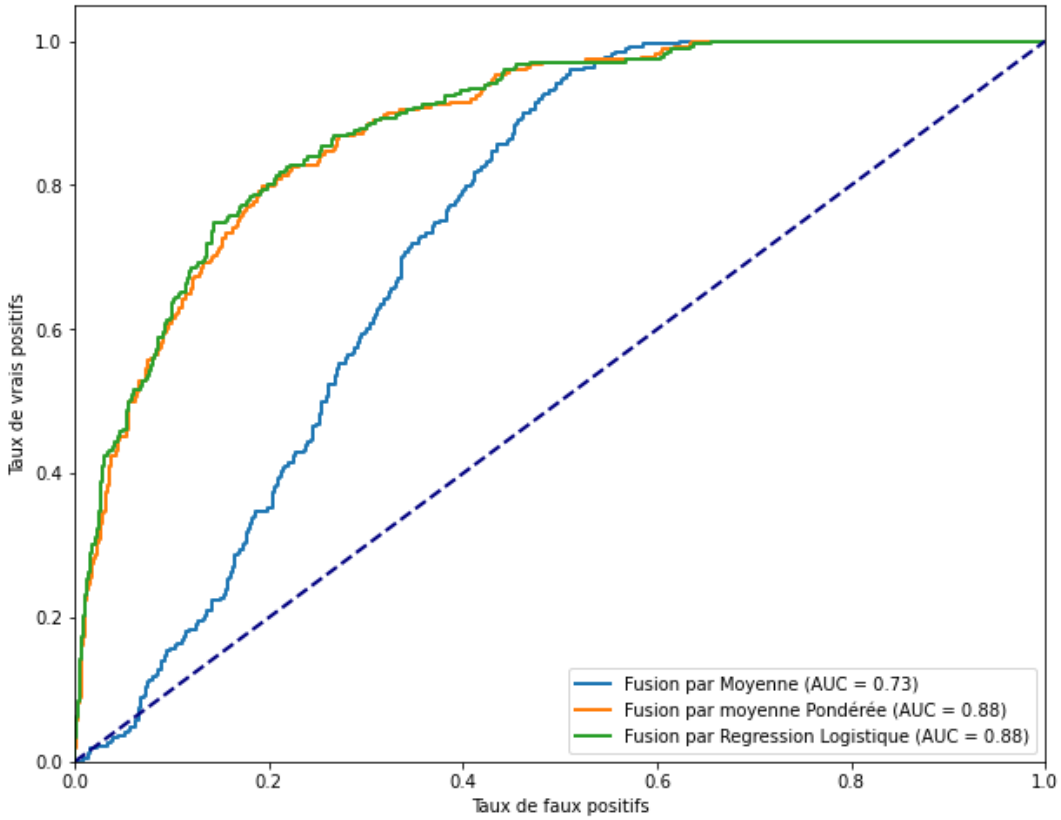

Figure 18: Courbes ROC des fusions des modèles

CONCLUSION

Ce projet avait pour principal objectif de développer un système de classification d'images visant à améliorer le diagnostic du mélanome, un type de cancer cutané particulièrement agressif. À travers différentes approches, allant des modèles classiques comme K-Nearest Neighbors (KNN) et Support Vector Machine (SVM) à l'apprentissage profond avec des réseaux de neurones convolutifs (CNN) et EfficientNetB0, nous avons exploré plusieurs méthodes pour évaluer leur efficacité dans la détection de cette maladie.

Les résultats obtenus montrent que l'apprentissage profond, notamment avec le modèle EfficientNetB0, a offert des performances nettement supérieures aux méthodes classiques, avec une AUC atteignant 88 %. De plus, la fusion de modèles s'est révélée être une stratégie prometteuse pour améliorer les résultats, la méthode de moyenne pondérée offrant un bon compromis entre sensibilité et spécificité.

En conclusion, ce travail a mis en évidence l'importance de l'intelligence artificielle dans l'amélioration des outils de diagnostic assisté par ordinateur, particulièrement dans le domaine médical. Les perspectives d'amélioration incluent l'exploration d'autres méthodes de fusion de modèles et l'augmentation des jeux de données pour affiner davantage la précision du système.

REFERENCES BIBLIOGRAPHIQUES

[1] R. J. Hemalatha, B. Babu et al., «A comparison of filtering and enhancement methods in malignant melanoma images,» *IEEE International Conference on Power, Control,Signals and Instrumentation Engineering (ICPCSI),* pp. 2704-2710, 2017.

[2] V. L. REVATHI et A. S. CHITRA, «A Review on Segmentation Techniques in Lesion on skinImages,» *International Research Journal of Engineering and Technology (IRJET),* vol. 2, n° %109, 2015.

[3] ALENEZI, FAYADH et al., «A novel multi-task learning network based on melanoma segmentation and classification with skin lesion images,» *Diagnostics,* vol. vol. 13, n° %1no 2, p. p. 262., 2023.

[4] GHALIB, V. A et al., «Automatic detection and classification of skin cancer.,» *Int. J. Intell. Eng. Syst.,* vol. 10, n° %13, 2017.

[5] P. YANG, W. SONG et al., «An improved Otsu threshold segmentation algorithm.,» *International Journal of Computational Science and Engineering,* pp. 146-153., 22(1).

[6] A. Murugan, S. A. H. Nair et al., «Detection of skin cancer using SVM, random forest and kNN classifiers,» *J. Med. Syst,* vol. 43, n° %18, p. 269, 2019.

[7] F. NACHBAR, W. STOLZ et al., «The ABCD rule of dermatoscopy: high prospective value in the diagnosis of doubtful,» *Journal of the American Academy of Dermatology,* vol. 30, n° %14, pp. 551-559, 1994.

[8] HAGERTY, R. JASON et al., «handcrafted method fusion: higher diagnostic accuracy for melanoma dermoscopy images,» *journal of biomedical and health informatics,* vol. 23, n° %14, pp. 1385-1391, 2019.