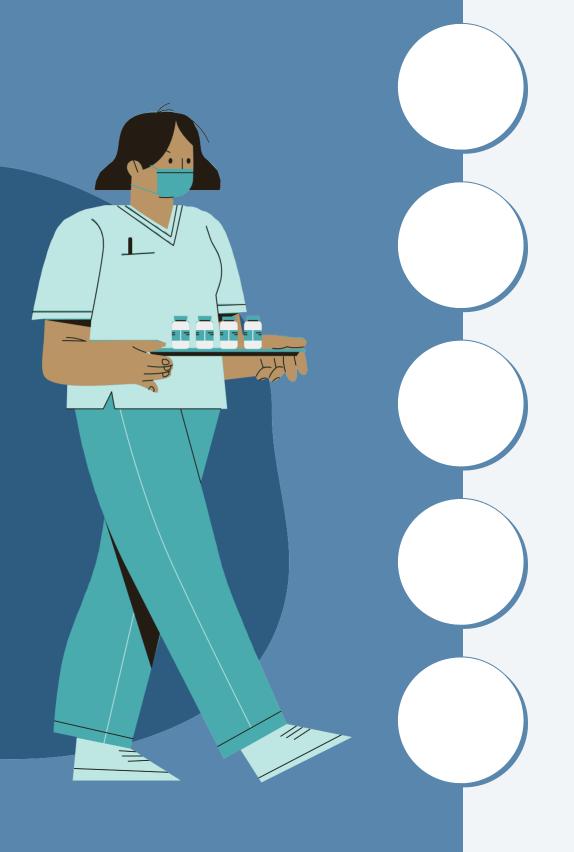
# PROJET IMAGE PROCESSING





## DETECTION OF SKIN CANCER WITH DEEP LEARNING

Réalisé par :

Soumaya El Amrani



## Tableau de matière

Introduction Description de la base données 3 Implémentation des modèles Tableau comparatif **Adversarial Attacks** 6 Conclusion



## Introduction

La peau forme une barrière protectrice entre l'organisme et l'environnement extérieur. Elle est formée de trois couches de tissus superposées qui sont, de l'extérieur vers l'intérieur : l'épiderme, le derme et l'hypoderme.

#### Les différentes formes de cancers de la peau

Il existe plusieurs formes de cancers de la peau, aussi appelés cancers cutanés :

- Le mélanome, la forme la plus grave mais la moins fréquente de cancer de la peau, qui provoque des métastases dans environ 20 % des cas. Principalement diagnostiqué après 50 ans, il peut apparaître sur n'importe quelle zone du corps mais est plus fréquent sur les zones exposées au soleil, à partir de la peau « normale » ou d'un grain de beauté.
- Le carcinome basocellulaire, qui se développe à partir des cellules dans la couche la plus profonde de l'épiderme.
- Le carcinome épidermoïde (appelé autrefois spinocellulaire), qui se développe au niveau de la couche intermédiaire de l'épiderme.

Certaines caractéristiques individuelles augmentent le risque de survenue de cancers cutanés. Toutefois, la plupart d'entre eux sont évitables ; les études ont montré que deux tiers des cancers de la peau seraient ainsi imputables à une exposition au soleil trop importante.

Ainsi, l'exposition aux rayons UV qu'ils soient d'origine naturelle (soleil) ou artificielle (les cabines de bronzage), le type d'exposition (prolongée ou intermittente mais intense avec coups de soleil), le phototype (peau et cheveux plus ou moins clairs), les grains de beauté, certaines maladies génétiques et antécédents familiaux sont des facteurs qui peuvent causer une augmentation du risque de développer un cancer de la peau.

## Description de la base données

Lien: Skin-cancer-malignant-vs-benign

Le cancer de la peau, la tumeur maligne la plus fréquente chez l'homme, est principalement diagnostiqué visuellement, en commençant par un dépistage clinique initial, suivi éventuellement d'une analyse dromoscopique, d'une biopsie et d'un examen histopathologique. La classification automatique des lésions cutanées à l'aide d'images est une tâche difficile en raison de la variabilité fine de l'apparence des lésions cutanées.

L'ensemble de données provient des archives ISIC (International Skin Image Collaboration). Il se compose de 1800 images de grains de beauté bénins et de 1497 images de grains de beauté malins classés. Les images ont toutes été redimensionnées en basse résolution (224x224x3) RGB. La tâche de ce noyau est de créer un modèle capable de classer visuellement un grain de beauté en bénin et malin.

Comme l'ensemble de données est assez équilibré, le modèle sera testé sur le score de précision, donc (TP + TN)/(ALL).

Il a 2 classes différentes de cancer de la peau qui sont listées ci-dessous :

- 1. Benign
- 2. Malignant



Dans ce projet, nous allons essayer de détecter 2 classes différentes de taupes en utilisant le réseau neuronal à convolution avec keras tensorflow en backend et ensuite analyser le résultat pour voir comment le modèle peut être utile dans un scénario pratique.

Nous avons suivi les étapes suivantes pour la construction du modèle et l'évaluation qui sont les suivantes :

**Step 1: Importing Essential Libraries** 

**Step 2: Loading pictures and making Dictionary of images and labels** 

**Step 3: Categorical Labels** 

**Step 4: Normalization** 

**Step 5: Train and Test Split** 

**Step 6: Model Building** 

**Step 7: Cross-validating model** 

**Step 8: Testing model** 

**Step 9: Classification avec ResNet50** 

**Step 10: Classification avec DenseNet121** 

**Step 11: Classification avec MobileNet** 

**Step 12: Prediction using CNN** 

**Step 13: Segmentation k-means** 

**Step 14: Adversarial attacks sur CNN + FGSM** 

## Implémentation des modèles :

#### Construction du modèle

Nous avons utilisé l'API Keras Séquentiel, où il suffit d'ajouter une couche à la fois, en partant de l'entrée.

La première est la couche convolutionnelle (Conv2D). C'est comme un ensemble de filtres apprenables. J'ai choisi de mettre 64 filtres pour les deux premières couches Conv2D. Chaque filtre transforme une partie de l'image (définie par la taille du noyau) en utilisant le filtre noyau. La matrice du filtre noyau est appliquée sur l'image entière. Les filtres peuvent être vus comme une transformation de l'image.

Le CNN peut isoler les caractéristiques qui sont utiles partout à partir de ces images transformées (cartes de caractéristiques).

La deuxième couche importante du CNN est la couche de pooling (MaxPool2D). Cette couche agit simplement comme un filtre de sous-échantillonnage. Elle examine les deux pixels voisins et choisit la valeur maximale. Ces filtres sont utilisés pour réduire le coût de calcul et, dans une certaine mesure, pour réduire l'overfitting. Nous devons choisir la taille de la mise en commun (c'est-à-dire la taille de la zone mise en commun à chaque fois). Plus la dimension de la mise en commun est élevée, plus le sous-échantillonnage est important.

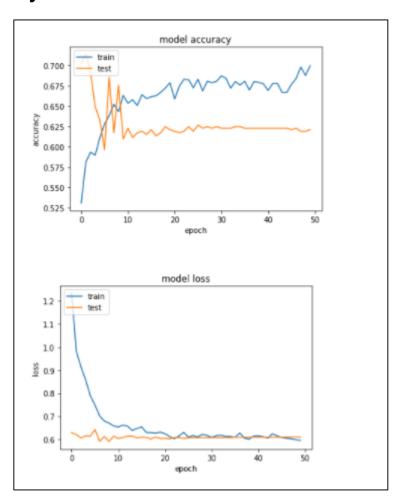
En combinant les couches de convolution et de mise en commun, les CNN sont capables de combiner les caractéristiques locales et d'apprendre des caractéristiques plus globales de l'image.

```
def build(input_shape= (224,224,3), 1r = 1e-3, num_classes= 2,
        init= 'normal', activ= 'relu', optim= 'adam'):
   model = Sequential()
   model.add(Conv2D(64, kernel_size=(3, 3),padding = 'Same',input_shape=input_shape,
                    activation= activ, kernel_initializer='glorot_uniform'))
   model.add(MaxPool2D(pool\_size = (2, 2)))
   model.add(Dropout(0.25))
   model.add(Conv2D(64, kernel_size=(3, 3),padding = 'Same',
                    activation =activ, kernel_initializer = 'glorot_uniform'))
   model.add(MaxPool2D(pool\_size = (2, 2)))
   model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer=init))
    model.add(Dense(num_classes, activation='softmax'))
    model.summary()
    if optim == 'rmsprop':
        optimizer = RMSprop(lr=lr)
    else:
        optimizer = Adam(lr=lr)
    model.compile(optimizer = optimizer ,loss = "binary_crossentropy", metrics=["accuracy"])
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=5.
                                            verbose=1,
                                            factor=0.5.
                                            min_lr=1e-7)
```

#### **Summary of Accuracies and Losses:**

	Training	Test
Accuracy	66.67%	62.67%
Loss	61.67%	63.67%

#### Training Vs. test Accuracy/Loss:



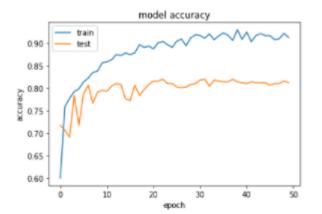
#### Modèle Resnet50:

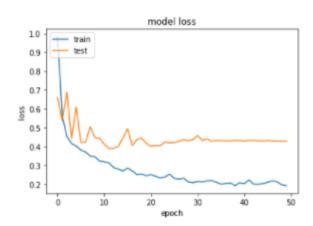
Le CNN ci-dessus n'est pas un modèle très sophistiqué, donc le resnet50, est aussi essayé

#### **Summary of Accuracies and Losses:**

	Training	Test
Accuracy	95.43%	81.96%
Loss	25.57%	45.67%

#### Training Vs. test Accuracy/Loss:





#### Train ResNet50 sur toutes les images de la base de données :

#### Test ResNet50 pour l'évaluation:

```
# Testing model on test data to evaluate
y_pred = model.predict(X_test)
print(accuracy_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1)))

# save model
# serialize model to JSON
resnet50_json = model.to_json()

with open("resnet50.json", "w") as json_file:
    json_file.write(resnet50_json)

# serialize weights to HDF5
model.save_weights("resnet50.h5")
print("Saved model to disk")

0.8196969696969697
Saved model to disk
```

#### Modèle MobileNet:

La classification pat MobileNet

#### **Training a simple Model with kfolds:**

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import StratifiedKFold
model_history = []
skf = StratifiedKFold(n_splits=FOLDS, random_state=None, shuffle=False)
for train_index, val_index in skf.split(X,y):
   print('*****Starting fold ', i+1, ' of ', FOLDS)
   filepath = 'best_mobilenet_fold_'+str(i)+'.h5'
   X_{train}, X_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}, Y_{train}
   y_train = to_categorical(y_train, num_classes=2)
   y_test = to_categorical(y_test, num_classes=2)
   datagen = ImageDataGenerator(shear_range=0.2,
                                  rotation_range=40,
                                  brightness_range=(0.5, 1.5),
                                  vertical_flip=True,
                                  horizontal_flip=True)
```

#### **Summary of Accuracies and Losses:**

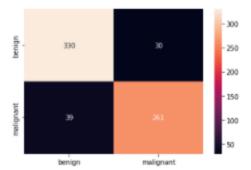
	Training	Test
Accuracy	65.67%	66.56%
Loss	45.78%	41.67%

#### Plotting model accuracy by fold:



#### Confusion Matrix :

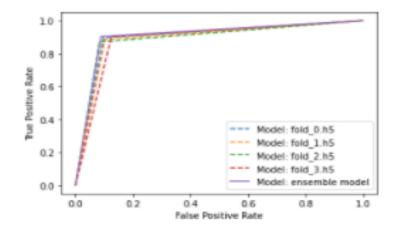
```
ROC AUC score: 0.8919444444444444
evaluating: best_mobilenet_fold_2.h5
Classification Report:
         precision recall f1-score support
    benign
             0.89
                   0.92
                           0.91
                                  360
                           0.88
  malignant
                    0.87
                                  300
                           0.90
                                  660
  accuracy
             0.90
                    0.89
                           0.89
                                  660
weighted avg
             0.90
                    0.90
                           0.90
                                  660
Confusion Matrix:
```



#### Plotting AUC Curves

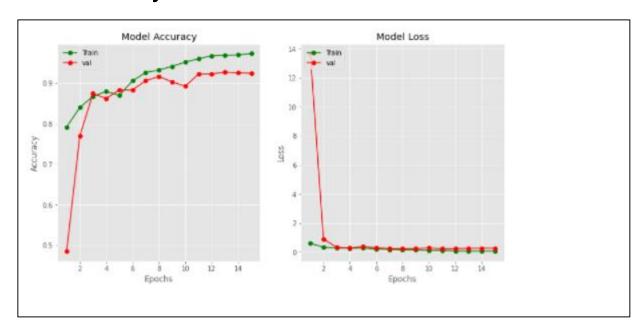
```
for i in range(FOLDS):
    fpr,tpr,_ = roc_curves[i]
    plt.plot(fpr, tpr, linestyle='--', label='Model: fold_'+str(i)+'.h5')
    # axis labels

plt.plot(fpr_ensemble, tpr_ensemble, linestyle='-', label='Model: ensemble model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```



#### Modèle DenseNet121:

#### Training Vs. test Accuracy/Loss:



#### **Summary of Accuracies and Losses:**

	Training	Test	
Accuracy	98.23%	91.24%	
Loss	1.23%	1.67%	



## Tableau Comparatif:

Un aperçu sur les performances impressionnantes atteintes de différents modèles exécutés sur notre dataset.

	Mesure de performance	Training	Test
Module from scratch	Accuracy	66.67%	62.67%
	Loss	61.67%	63.67%
ResNet50	Accuracy	95.43%	81.96%
	Loss	25.57%	45.67%
MobileNet	Accuracy	65.67%	66.56%
	Loss	45.78%	41.67%
DenseNet121	Accuracy	98.23%	91.24%
	Loss	1.23%	1.67%

## Segmentation k-means:

Consultez le notebook



Segmentation k-means

## Adversarial attacks:



Consultez le notebook

Adversarial attacks sur CNN + FGSM

## Notebooks:

Classification avec construction CNN et desnet

**Classification+ predictor** 

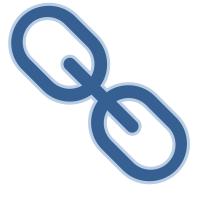
Skin cancer prediction using CNN

**Classification avec DenseNet121** 

**Classification avec MobileNet** 

Segmentation k-means

Adversarial attacks sur CNN + FGSM



### Conclusion:

Le cancer de la peau, la tumeur maligne la plus fréquente chez l'homme, est principalement diagnostiqué visuellement, en commençant par un dépistage clinique initial, suivi éventuellement d'une analyse dromoscopique, d'une biopsie et d'un examen histopathologique. La classification automatique des lésions cutanées à l'aide d'images est une tâche difficile en raison de la variabilité fine de l'apparence des lésions cutanées.

En somme notre projet s'inscrit dans la logique de deep learning qui fait appel à un ou plusieurs modèles. Le choix de ces derniers se base sur la performance et la précision que l'on veut afficher pour la thématique choisie. La détection de skin cancer à partir des images de rayon X, a nécessité l'implémentation et l'exécution du modèle from scrutch, ResNet50, MobileNet et DenseNet121 et respectivement. En addition, nous avons implémenté notre propre architecture bien adaptée à ce cas chose qui a donné des performances impressionnantes.