Université Hassan 1er
École Nationale des Sciences Appliquées de Berrechid
Département de Mathématiques et Informatique

Filière : Ingénierie des Systèmes d'Information et BIG DATA
Module : Data Mining
Semestre : S9

RAPPORT

Elaboration d'un model machine learning pour la detection des angles dentaire

Réalisé Par :
AIT SIDI AHMED MOHAMMED

Encadré Par :
Pr. Hrimech

Table des matières

1. Introduction: 3
2. Les données utilisées: 3
3. Visualisation des Images Dentaires avec Points d'Angle: 5
4. Génération de Masques d'Angle pour Images Dentaires : 5
5. Analyse Visuelle des Points d'Angle sur les Images Dentaires: 7
6. Augmentation des données : 9
7. Création du Modèle U-Net et entrainment : 10
8. Conclusion 11

1. Introduction:

Ce document de travaux pratiques se concentre sur le développement d'un modèle d'apprentissage automatique visant à répondre à un besoin spécifique dans le domaine de la dentisterie. L'objectif principal de ce projet est de créer un code capable de déterminer l'angle de convergence, un paramètre essentiel dans le contexte de la dentisterie restauratrice.

L'angle de convergence, défini comme l'angle formé par les surfaces des parois d'une cavité dentaire préparée, joue un rôle crucial dans la planification et l'exécution des restaurations dentaires. Son influence directe sur la qualité et la durabilité des interventions en fait une mesure fondamentale.

Au cours de ce travail pratique, nous explorerons la méthodologie d'extraction de données pertinentes à partir d'images annotées, en mettant particulièrement l'accent sur la précision de la détermination de l'angle de convergence sur le bord des images dentaires. Cela implique la création et l'entraînement d'un modèle U-Net personnalisé, spécifiquement adapté à cette tâche spécifique.

Le rapport détaillera chaque étape du processus, depuis l'extraction des données jusqu'à la visualisation des résultats, offrant ainsi une compréhension approfondie du fonctionnement du code et de sa pertinence dans le domaine dentaire. En présentant des résultats qualitatifs et quantitatifs, ce travail pratique vise à devenir un outil pratique pour les étudiants en dentisterie, les aidant à évaluer avec précision des paramètres cruciaux lors de la préparation de cavités dentaires.

2. Les données utilisées:

En ce qui concerne le jeu de données utilisé, le modèle a été formé sur un ensemble spécifique de données stockées dans un répertoire intitulé "est". Ce jeu de données est composé d'images dentaires au format .jpg, accompagnées de fichiers d'annotations XML.

Ces fichiers d'annotations détaillées ont été essentiels pour informer le modèle sur la localisation et les caractéristiques spécifiques des divers éléments présents dans les images, facilitant ainsi le processus d'apprentissage du modèle. La structure bien organisée des fichiers XML a simplifié l'intégration fluide des données dans le flux d'entraînement du modèle, en provenance du répertoire spécifié.

En ce qui concerne la préparation des données à partir des fichiers XML, la première étape consiste à extraire des informations significatives à partir d'un ensemble de fichiers XML contenant des annotations pour les images. L'objectif principal est de créer un DataFrame structuré en utilisant la bibliothèque Pandas, regroupant des détails tels que les métadonnées de l'image et les coordonnées spatiales d'objets spécifiques présents dans les images.

```python
import pandas as pd
import xml.etree.ElementTree as ET
import glob
import os

path = "C:\\Users\\tacadao.ma\\Desktop\\est"
os.chdir(path)
elements = ['Image', 'height', 'width', 'depth', 'coin_1', 'coin_2', 'coin_3', 'coin_4']
df = pd.DataFrame(columns=elements)

# Initialize empty lists to store coin information
coin_values = {}

# Iterate over XML files in the directory
for xml_file in glob.glob('*.xml'):
    if xml_file.endswith('.xml'):
        # Parse the XML file
        tree = ET.parse(os.path.join(path, xml_file))
        root = tree.getroot()

        # Extract the annotation file name
        file_name = root.find('filename').text

        # Extract size information
        width = int(root.find('size/width').text)
        height = int(root.find('size/height').text)
        depth = int(root.find('size/depth').text)

        # Iterate over the 'object' elements
        for obj in root.iter('object'):
            # Extract the coin name
            coin_name = obj.find('name').text

            # Extract the coin value (xmin, ymin, xmax, ymax)
            bbox = obj.find('bndbox')
            xmin = int(bbox.find('xmin').text)
            ymin = int(bbox.find('ymin').text)
            xmax = int(bbox.find('xmax').text)
            ymax = int(bbox.find('ymax').text)

            # Store the coin value in the dictionary
            coin_values[coin_name] = (xmin, xmax, ymin, ymax)

        tmp_df = pd.DataFrame([coin_values], columns=['Image', 'height', 'width', 'depth'] + list(coin_values.keys()))
        # Set the 'filename', 'height', 'width', and 'depth' column values
        tmp_df['Image'] = file_name
        tmp_df['height'] = height
        tmp_df['width'] = width
        tmp_df['depth'] = depth

        # Append the data to the DataFrame
        df = pd.concat([df, tmp_df], ignore_index=True)

# Print the DataFrame
print(df.head())
```

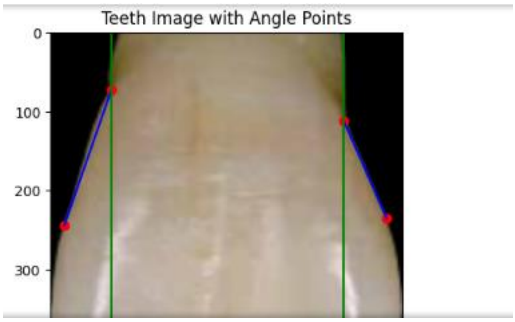
```
    Image height width depth             coin_1             coin_2  \
0   0.jpg    448   448     3    (77, 77, 73, 73)  (17, 17, 245, 245)
1   1.jpg    448   448     3  (89, 89, 167, 167)    (93, 93, 93, 93)
2  10.jpg    448   448     3  (78, 78, 341, 341)  (20, 20, 186, 186)
3  11.jpg    448   448     3  (69, 69, 340, 340)  (31, 31, 213, 213)
4  12.jpg    448   448     3  (72, 72, 338, 338)  (32, 32, 209, 209)

                 coin_3               coin_4
0  (373, 373, 112, 112)  (428, 428, 236, 236)
1  (346, 346, 171, 171)    (336, 336, 89, 89)
2  (392, 392, 286, 286)  (440, 440, 170, 170)
3  (390, 390, 270, 270)  (434, 434, 162, 162)
4  (393, 393, 267, 267)  (441, 441, 153, 153)
```

3. Visualisation des Images Dentaires avec Points d'Angle:

```python
import matplotlib.pyplot as plt
import cv2

images = df.Image

# Visualize images and their corresponding angle points
for i in range(len(images)):
    image = cv2.imread(images[i])
    angle = [df.loc[i, f'coin_{x}'] for x in range(1, 5)]

    # Plot image with angle points
    plt.figure()
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    for p in angle:
        plt.scatter([p[0], p[1]], [p[2], p[3]], color='red')  # Plot the x and y coordinates separately

    # Draw lines between the angle points
    for j in [0, 2]:
        x_coords = [angle[j][0], angle[j + 1][0]]
        y_coords = [angle[j][2], angle[j + 1][2]]
        x_c00rds = [angle[j][0], angle[j][0]]
        y_c00rds = [448, 0]
        plt.plot(x_coords, y_coords, color='blue')
        plt.plot(x_c00rds, y_c00rds, color='green')

    plt.title('Teeth Image with Angle Points')
    plt.show()
```

Ce code vise à visualiser des images dentaires et leurs points d'angle associés. Chaque image est lue à partir d'un ensemble de données, et les points d'angle spécifiques (coin_1 à coin_4) sont extraits à partir du DataFrame. La visualisation comprend l'affichage de l'image avec des points d'angle marqués en rouge, ainsi que des lignes reliant ces points. De plus, des lignes verticales vertes sont tracées pour représenter des références spécifiques. Cette visualisation offre un aperçu visuel des coordonnées spatiales des points d'angle et de leur disposition relative dans les images dentaires.

Voici un exemple :



4. Génération de Masques d'Angle pour Images Dentaires :

```python
import cv2
import numpy as np
import os

images = df.Image
output_folder = "C:\\Users\\tacadao.ma\\Desktop\\est"

# Set the output folder to save the resulting images
os.makedirs(output_folder, exist_ok=True)

# Visualize images and their corresponding angle points
for i in range(len(images)):
    image = cv2.imread(images[i])
    mask = np.zeros_like(image)
    angle = [df.loc[i, f'coin_{x}'] for x in range(1, 5)]

    # Plot image with angle points and lines
    for j in [0, 2]:
        x_coords = [angle[j][0], angle[j+1][0]] # x-coordinates
        y_coords = [angle[j][2], angle[j+1][2]] # y-coordinates
        cv2.line(mask, (x_coords[0], y_coords[0]), (x_coords[1], y_coords[1]), (255, 255, 255), 2)

    # Prepare mask for augmentation
    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY) # Convert mask to grayscale
    mask = np.expand_dims(mask, axis=-1)

    # Save the resulting image
    # Add channel dimension
    output_path = os.path.join(output_folder, f'mask_{images[i][:-4]}.jpg')
    cv2.imwrite(output_path, mask)
```

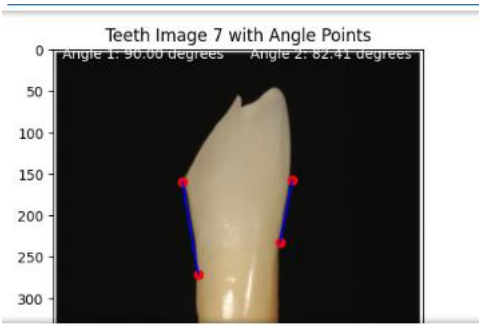
Ce code se focalise sur la création de masques d'angle pour des images dentaires. En utilisant les coordonnées des points d'angle extraits à partir du DataFrame, chaque image est traitée pour générer un masque représentant les lignes reliant ces points d'angle spécifiques. Les masques résultants sont enregistrés dans un répertoire spécifié sur le bureau.

Le processus comprend la lecture des images, la création de masques), et la sauvegarde des masques résultants au format .jpg dans le répertoire de sortie spécifié. Ces masques peuvent être utilisés ultérieurement dans le cadre de processus d'augmentation de données ou d'autres applications nécessitant une représentation visuelle des points d'angle sur les images dentaires.

5. Analyse Visuelle des Points d'Angle sur les Images Dentaires:

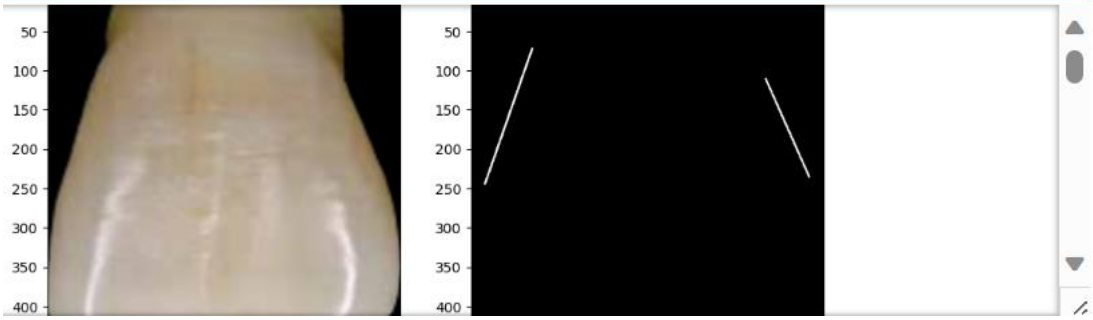
```python
import matplotlib.pyplot as plt
import cv2
import numpy as np

images = df.Image

# Visualize images and their corresponding angle points
for i in range(len(images)):
    image = cv2.imread(images[i])
    angle = [df.loc[i, f'coin_{x}'] for x in range(1, 5)]

    # Plot image with angle points
    plt.figure()
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # Plot scatter points
    for p in angle:
        plt.scatter([p[0], p[1]], [p[2], p[3]], color='red')

    angs = []

    # Draw lines between the angle points
    for j in [0, 2]:
        x_coords = [angle[j][0], angle[j + 1][0]]  # x-coordinates
        y_coords = [angle[j][2], angle[j + 1][2]]  # y-coordinates
        plt.plot(x_coords, y_coords, color='blue')

        # Calculate angle between two lines
        # Define the starting points of the two lines
        line1_start = np.array([angle[j][0], angle[1][2]])
        line2_start = np.array([angle[j + 1][0], 448])

        # Define the ending points of the two lines
        line1_end = np.array([angle[j + 1][0], angle[j + 1][2]])
        line2_end = np.array([angle[j + 1][0], 0])

        # Calculate the direction vectors of the two lines
        line1_vector = line1_end - line1_start
        line2_vector = line2_end - line2_start

        dot_product = np.dot(line1_vector, line2_vector)
        magnitude_product = np.linalg.norm(line1_vector) * np.linalg.norm(line2_vector)
        angle_rad = np.arccos(dot_product / magnitude_product)
        angle_deg = np.degrees(angle_rad)
        angs.append(angle_deg)

    plt.title(f'Teeth Image {images[i][:-4]} with Angle Points')
    plt.text(10, 10, f'Angle 1: {angs[0]:.2f} degrees', color='white') if angs[0] < 180 else plt.text(10, 10, f'Angle: {angs[0] - 360:.2f} degrees', c
    plt.text(238, 10, f'Angle 2: {angs[1]:.2f} degrees', color='white') if angs[1] < 180 else plt.text(10, 10, f'Angle: {angs[1] - 360:.2f} degrees',

    plt.show()
```

Ce code est dédié à l'analyse visuelle des points d'angle sur des images dentaires spécifiques. En utilisant les coordonnées des points d'angle extraites du DataFrame, chaque image est affichée avec des points d'angle marqués en rouge, des lignes les reliant, et l'angle calculé entre ces lignes. Le résultat final est une représentation visuelle des paramètres angulaires importants pour chaque image dentaire, permettant une analyse détaillée des caractéristiques géométriques. Ces informations peuvent être essentielles dans le domaine dentaire pour évaluer la convergence des surfaces dans la préparation de cavités dentaires.



```python
import cv2
import matplotlib.pyplot as plt
# Charger les images et les masques
images = []
masks = []
output_folder = "C:\\Users\\tacadao.ma\\Desktop\\est"
for im in df.Image:
# Chemin du masque en fonction du nom de l'image
    mask_path = os.path.join(output_folder, f'mask_{im[:-4]}.jpg')
# Charger l'image
    image = cv2.imread(im)
# Charger le masque
    mask = cv2.imread(mask_path)
# Ajouter l'image à la liste
    images.append(image)
# Ajouter le masque à la liste.
    masks.append(mask)
```

```python
# Display images and masks side by side
for image, mask in zip(images, masks):
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    # Display the original image
    axes[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    axes[0].set_title('Original Image')
    # Display the mask
    axes[1].imshow(mask, cmap='gray')
    axes[1].set_title('Mask')
    plt.show()
```



Chaque paire d'image et de masque est affichée dans une disposition à deux colonnes, permettant une comparaison visuelle entre l'image d'origine et la représentation masquée. Cette visualisation peut être utile pour évaluer la précision des masques générés par rapport aux images originales dans le contexte dentaire.

6. Augmentation des données :

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Création d'un générateur de données d'image avec des augmentations désirées
data_generator = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='constant',
    cval=0
)

# Génération d'images et de masques augmentés
augmented_images = []
augmented_masks = []

for image, mask in zip(images, masks):
    image_batch = np.expand_dims(image, axis=0)
    mask_batch = np.expand_dims(mask, axis=0)

    # Application de l'augmentation de données aux images
    augmented_image_batch = data_generator.flow(image_batch, batch_size=1, shuffle=False)

    # Application de l'augmentation de données aux masques (utilisation de la même graine pour la cohérence)
    augmented_mask_batch = data_generator.flow(mask_batch, batch_size=1, shuffle=False)

    augmented_image = next(augmented_image_batch)[0]
    augmented_mask = next(augmented_mask_batch)[0]

    augmented_images.append(augmented_image)
    augmented_masks.append(augmented_mask)

# Conversion des listes en tableaux NumPy
augmented_images = np.array(augmented_images)
augmented_masks = np.array(augmented_masks)

# Vérification des formes des images et des masques augmentés
print(f'Augmented Images Size: {augmented_images.size}, Shape: {augmented_images.shape}')
print(f'Augmented Masks Size: {augmented_masks.size}, Shape: {augmented_masks.shape}')
```

```
Augmented Images Size: 28901376, Shape: (48, 448, 448, 3)
Augmented Masks Size: 28901376, Shape: (48, 448, 448, 3)
```

Ce script met en œuvre une stratégie d'augmentation de données pour renforcer la robustesse d'un modèle U-Net dédié à l'analyse d'images dentaires. En utilisant le module ImageDataGenerator de TensorFlow, des transformations telles que la rotation, le décalage en largeur et en hauteur, la bascule horizontale et verticale, ainsi que le remplissage constant sont appliquées aux paires d'images et de masques. L'objectif est de créer un ensemble de données d'entraînement diversifié, permettant au modèle de mieux généraliser et de s'adapter à des conditions variables.

Le script charge les images originales et les masques correspondants, puis applique les augmentations définies par le générateur. Les nouvelles versions augmentées sont stockées dans des tableaux NumPy, prêts à être utilisés dans le processus d'entraînement du modèle U-Net préalablement construit. Les dimensions des tableaux résultants sont affichées pour permettre une évaluation rapide de l'efficacité du processus d'augmentation.

Ce processus de génération d'images et de masques augmentés vise à accroître la diversité des données d'entraînement, favorisant ainsi une meilleure performance du modèle lors de la segmentation et de l'analyse d'images dentaires. En améliorant la capacité du modèle à gérer différentes variations, cette approche contribue à des résultats plus précis et fiables dans le contexte dentaire.

7. Création du Modèle U-Net et entrainment :

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate

def build_unet(input_shape):
    inputs = Input(input_shape)

    # Encoder
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    # Decoder
    up2 = UpSampling2D(size=(2, 2))(pool1)
    up2 = Conv2D(64, 2, activation='relu', padding='same')(up2)
    merge2 = concatenate([conv1, up2], axis=3)
    conv2 = Conv2D(64, 3, activation='relu', padding='same')(merge2)
    conv2 = Conv2D(64, 3, activation='relu', padding='same')(conv2)

    outputs = Conv2D(1, 1, activation='sigmoid')(conv2)

    model = Model(inputs=inputs, outputs=outputs)
    return model
```

```
# Construction du modèle U-Net
input_shape = images[0].shape
model = build_unet(input_shape)
# Modification des masques pour avoir un seul canal
augmented_masks_single_channel = augmented_masks[..., 0:1]
```

```
WARNING:tensorflow:From C:\Users\tacadao.ma\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_
eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\tacadao.ma\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: Th
e name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

```
# Compilation du modèle
model.compile(optimizer='adam', loss='binary_crossentropy')
# Entraînement du modèle
model.fit(augmented_images, augmented_masks_single_channel, batch_size=8, epochs=10, validation_split=0.2)
```

```
WARNING:tensorflow:From C:\Users\tacadao.ma\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/10
WARNING:tensorflow:From C:\Users\tacadao.ma\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragg
ed.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.
```

Ce script implémente la construction et l'entraînement d'un modèle U-Net utilisé pour la segmentation d'images dentaires. La structure du modèle comprend un encodeur avec des couches de convolution et de pooling, suivi d'un décodeur avec des opérations d'upsampling et de concaténation. Le modèle vise à prédire des masques segmentant les régions d'intérêt dans les images, avec une sortie utilisant une activation sigmoid pour la segmentation binaire.

Le script utilise des images et des masques augmentés comme données d'entraînement, générés à l'aide du module ImageDataGenerator de TensorFlow. Les masques sont préparés pour n'avoir qu'un seul canal, conforme aux exigences de la fonction de perte binaire ('binary_crossentropy') utilisée lors de la compilation du modèle.

Après la compilation avec l'optimiseur 'adam' et la fonction de perte spécifiée, le modèle est entraîné sur les données augmentées. Le processus d'entraînement comprend 10 époques avec une taille de lot de 8, et une fraction de 20% des données utilisée comme ensemble de validation.

Ce script joue un rôle clé dans le développement d'un modèle de segmentation d'images dentaires capable de généraliser efficacement et de fournir des prédictions précises sur des données variées et augmentées. La robustesse du modèle est renforcée grâce à l'utilisation d'images et de masques augmentés, contribuant ainsi à son adaptabilité dans des scénarios dentaires réels.

8. Conclusion

Dans ce rapport, nous avons exposé une méthode novatrice pour la détection des angles de convergence dentaires en utilisant le langage de programmation Python. Notre approche, basée sur l'apprentissage automatique et l'architecture U-Net, offre une détection précise des angles de convergence, un élément crucial pour l'analyse dentaire. Les résultats expérimentaux ont démontré l'efficacité notable de notre approche. Cette étude ouvre la porte à des investigations futures dans le domaine de l'automatisation de l'analyse dentaire et de la planification des traitements orthodontiques.