UNIVERSITE HASSAN PREMIER

ECOLE NATIONALE DES SCIENCES APPLIQUEES

- BERRECHID -

L'ECOLE NATIONAL DES SCIENCES APPLIQUEES DE BERRECHID

Filière : « Ingénierie des Systèmes d'Information et Big Data»

Réalisé par

RAJAE HESSANI

Rapport de TP

Encadré par | Pr. Hamid HRIMECH

Année Universitaire : 2023/2024

Table de matière :

Introduction générale

La dentisterie restauratrice, en constante évolution, s'appuie de plus en plus sur les avancées technologiques pour améliorer la qualité et la précision des interventions dentaires. Parmi les paramètres cruciaux, l'angle de convergence, défini comme l'angle formé par les surfaces des parois d'une cavité préparée dans une dent, joue un rôle essentiel dans la planification et la réalisation des restaurations dentaires. Ce travail pratique se consacre à l'exploration d'une approche novatrice en utilisant l'apprentissage automatique pour déterminer de manière précise l'angle de convergence sur des images dentaires.

Le processus complexe débute par l'extraction minutieuse de données pertinentes à partir d'images annotées, en mettant particulièrement l'accent sur la détermination précise de l'angle de convergence le long des bords des images dentaires. Pour ce faire, nous avons élaboré et entraîné un modèle U-Net sur mesure, spécifiquement adapté à cette tâche spécifique.

Ce rapport détaillera chaque étape du processus, de l'extraction des données à la visualisation des résultats, offrant ainsi une vision complète du fonctionnement du code et de son applicabilité dans le domaine dentaire. Les résultats obtenus, qu'ils soient qualitatifs ou quantitatifs, visent à fournir aux étudiants en dentisterie un outil pratique pour évaluer avec précision des paramètres cruciaux dans la préparation des cavités dentaires, inscrivant ainsi cette approche innovante dans une perspective d'amélioration continue des pratiques dentaires.

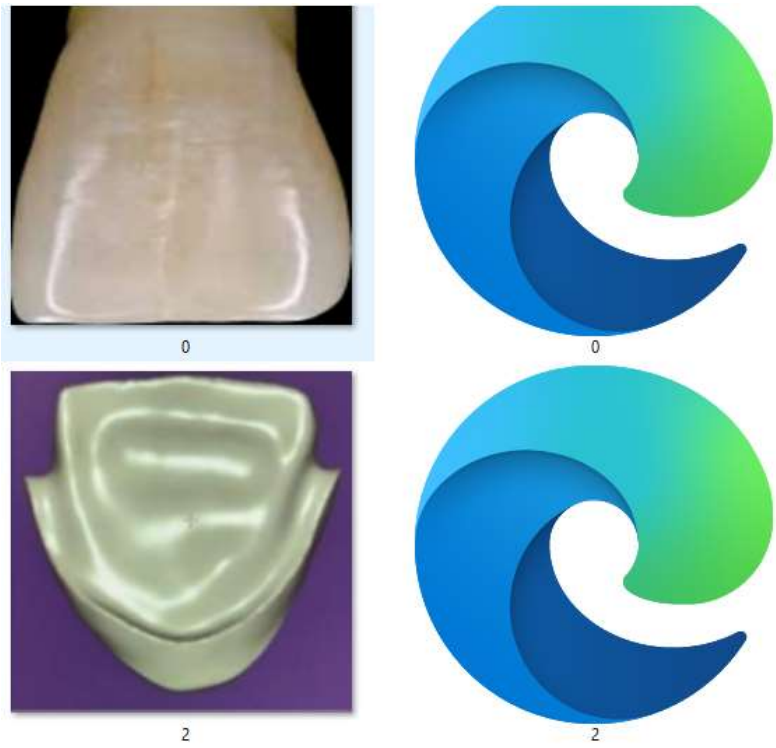
I. L’objectif du TP :

L'objectif central de ce travail pratique est de concevoir et mettre en œuvre un modèle d'apprentissage automatique capable de prédire avec précision les angles de convergence dans le contexte de la dentisterie restauratrice. En se focalisant sur cette tâche spécifique, notre démarche vise à exploiter les techniques avancées de l'apprentissage automatique pour automatiser et optimiser la détermination de cet angle crucial. À travers l'utilisation d'un modèle U-Net personnalisé, spécialement conçu pour cette application dentaire, nous aspirons à fournir un outil efficace et précis aux professionnels de la dentisterie. Cette approche innovante vise à simplifier le processus de planification des interventions dentaires en fournissant des prédictions fiables sur les angles de convergence, contribuant ainsi à l'amélioration des résultats cliniques et à l'efficacité des procédures de restauration dentaire. En mettant l'accent sur la pertinence clinique de la prédiction des angles dentaires, ce travail pratique s'inscrit dans une perspective d'optimisation des pratiques dentaires par le biais de solutions technologiques innovantes.

II. Le dataset :

Le modèle conçu a été formé sur un ensemble de données spécifique stocké dans un répertoire intitulé "est". Cette collection de données comprend des images dentaires au format .jpg, accompagnées de fichiers d'annotations XML. Ces annotations détaillées ont servi à instruire le modèle quant à la localisation et aux caractéristiques des différentes composantes des images, des informations cruciales pour son processus d'apprentissage. La structure organisée des fichiers XML a grandement facilité l'intégration des données dans le flux d'entraînement du modèle, en partant du répertoire spécifié.

Pour la préparation des données issues des fichiers XML, la première étape se concentre sur l'extraction d'informations pertinentes à partir d'un ensemble de fichiers XML qui contiennent des annotations pour les images. L'objectif premier est de construire un DataFrame organisé à l'aide de la bibliothèque Pandas. Ce DataFrame regroupe des détails essentiels tels que les métadonnées de l'image et les coordonnées spatiales d'objets spécifiques présents dans les images.



III. Prétraitement des données :

1. Extraction de quelques informations des images à partir des fichiers xml :

La phase de prétraitement des données s'avère essentielle pour organiser les informations extraites des fichiers XML et les préparer de manière adéquate pour l'entraînement du modèle. Dans le script fourni, nous utilisons des bibliothèques telles que Pandas, xml.etree.ElementTree, glob, et os pour réaliser cette tâche.

Tout d'abord, un DataFrame vide est initialisé avec les colonnes nécessaires, telles que 'Image', 'height', 'width', 'depth', 'coin_1', 'coin_2', 'coin_3', et 'coin_4'.

```
elements = ['Image','height','width','depth','coin_1','coin_2','coin_3','coin_4']
```

Ensuite, nous utilisons une boucle pour parcourir tous les fichiers XML du répertoire spécifié. Chaque fichier XML est analysé pour extraire des informations telles que le nom du fichier, les dimensions de l'image (largeur,

hauteur, profondeur), et les coordonnées spatiales des objets (coins) présents dans l'image.

Les informations extraites sont stockées dans un dictionnaire, coin_values, avec le nom du coin en tant que clé et les coordonnées (xmin, ymin, xmax, ymax) en tant que valeurs. Un DataFrame temporaire, tmp_df, est ensuite créé à partir de ce dictionnaire, et les colonnes 'Image', 'height', 'width', 'depth', ainsi que les colonnes spécifiques à chaque coin, sont remplies avec les valeurs appropriées.

```
df=pd.DataFrame(columns=elements)

coin_names = []
coin_values = {}

for xml_file in glob.glob('*.xml'):
    if xml_file.endswith('.xml'):

        tree = ET.parse(os.path.join(_dir, xml_file))
        root = tree.getroot()
        file_name = root.find('filename').text
        width = int(root.find('size/width').text)
        height = int(root.find('size/height').text)
        depth = int(root.find('size/depth').text)

        for obj in root.iter('object'):
            coin_name = obj.find('name').text
            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)
            coin_values[coin_name] = (xmin, xmax, ymin, ymax)
```

Le DataFrame temporaire est ensuite concaténé avec le DataFrame principal, df, pour agréger progressivement les données extraites de tous les fichiers XML. En fin de compte, le DataFrame complet offre une structure organisée, prête à être utilisée dans le flux d'entraînement du modèle.

```
tmp_df=pd.DataFrame([coin_values], columns=['Image','height','width','depth'] + list(coin_values.keys()))

tmp_df['Image'] = file_name
tmp_df['height']=height
tmp_df['width']=width
tmp_df['depth']=depth

df = pd.concat([df,tmp_df], ignore_index=True)
```

Le dataframe résultant est le suivant :

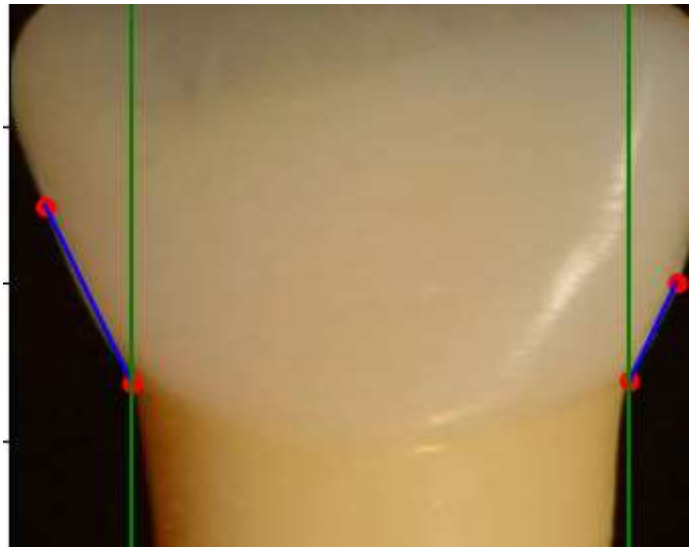
```
<bound method NDFrame.describe of      Image height width depth              coin_1              coin_2  \
0    0.jpg    448    448     3     (77, 77, 73, 73)    (17, 17, 245, 245)
1    1.jpg    448    448     3    (89, 89, 167, 167)      (93, 93, 93, 93)
2   10.jpg    448    448     3    (78, 78, 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)
5   13.jpg    448    448     3    (74, 74, 277, 277)    (26, 26, 163, 163)
6   14.jpg    448    448     3  (100, 100, 284, 284)    (42, 42, 169, 169)
7   15.jpg    448    448     3    (87, 87, 256, 256)    (35, 35, 158, 158)
8   16.jpg    448    448     3  (101, 101, 249, 249)    (39, 39, 147, 147)
9   17.jpg    448    448     3    (66, 66, 262, 262)    (19, 19, 171, 171)
10  18.jpg    448    448     3    (81, 81, 287, 287)    (28, 28, 196, 196)
11  19.jpg    448    448     3    (72, 72, 314, 314)    (17, 17, 233, 233)
12   2.jpg    448    448     3    (56, 56, 131, 131)      (61, 61, 41, 41)
13  20.jpg    448    448     3    (72, 72, 243, 243)    (23, 23, 149, 149)
14  21.jpg    448    448     3    (92, 92, 311, 311)    (44, 44, 247, 247)
15  22.jpg    448    448     3  (113, 113, 265, 265)    (61, 61, 138, 138)
16  23.jpg    448    448     3    (22, 22, 325, 325)    (44, 44, 182, 182)
17  24.jpg    448    448     3    (97, 97, 355, 355)    (19, 19, 207, 207)
18  25.jpg    448    448     3    (92, 92, 128, 128)    (64, 64, 296, 296)
19  26.jpg    448    448     3    (84, 84, 290, 290)    (23, 23, 166, 166)
20  27.jpg    448    448     3    (72, 72, 330, 330)    (15, 15, 181, 181)
21  28.jpg    448    448     3    (77, 77, 264, 264)    (22, 22, 151, 151)
22  29.jpg    448    448     3    (68, 68, 291, 291)    (29, 29, 195, 195)
23   3.jpg    448    448     3    (39, 39, 138, 138)      (40, 40, 70, 70)
24  30.jpg    448    448     3    (79, 79, 364, 364)    (29, 29, 247, 247)
```

2. Détection des coordonnées des cornes des images :

À travers une boucle itérative, chaque image est chargée et les coordonnées spatiales des points d'angle sont extraites du DataFrame. Ces coordonnées sont ensuite utilisées pour afficher l'image avec des points d'angle représentés par des cercles rouges. De plus, des lignes bleues sont tracées entre ces points pour illustrer la géométrie des angles. La figure résultante est affichée avec le titre "Teeth Image with Angle Points". Ces visualisations graphiques permettent une inspection visuelle approfondie des données, offrant une compréhension claire des caractéristiques spatiales des angles dans le contexte des images dentaires.

```python
images = df.Image
output_folder = 'D:/output_folder'

os.makedirs(output_folder, exist_ok=True)

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)]

    for j in [0, 2]:
        x_coords = [angle[j][0], angle[j + 1][0]]
        y_coords = [angle[j][2], angle[j + 1][2]]
        cv2.line(mask, (x_coords[0], y_coords[0]), (x_coords[1], y_coords[1]), (255, 255, 255), 2)

    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
    mask = np.expand_dims(mask, axis=-1)

    output_path = os.path.join(output_folder, f'mask_{images[i][:-4]}.jpg')
    cv2.imwrite(output_path, mask)
```

Un exemple de résultat est ci-dessous :



3. Génération de masque pour les points d'angle des images dentaires :

On a utilisé les coordonnées spatiales des points d'angle extraits de chaque image dentaire pour générer des masques visuels. Ces masques, représentant graphiquement la géométrie des angles, sont créés à l'aide de lignes blanches tracées entre les points d'angle sur un fond noir. Les masques ainsi obtenus sont ensuite convertis en niveaux de gris et adaptés pour être intégrés dans le flux d'entraînement des modèles d'apprentissage automatique. Cette approche s'avère essentielle pour enrichir notre ensemble de données en fournissant des informations visuelles supplémentaires, contribuant ainsi à renforcer la robustesse et la qualité de notre modèle de prédiction des angles dentaires. Les masques générés sont sauvegardés dans un dossier dédié, prêts à être exploités dans le processus d'entraînement de notre modèle.

```
images = df.Image
output_folder = 'D:/output_folder'

os.makedirs(output_folder, exist_ok=True)

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)]

    for j in [0, 2]:
        x_coords = [angle[j][0], angle[j + 1][0]]
        y_coords = [angle[j][2], angle[j + 1][2]]
        cv2.line(mask, (x_coords[0], y_coords[0]), (x_coords[1], y_coords[1]), (255, 255, 255), 2)

    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
    mask = np.expand_dims(mask, axis=-1)

    output_path = os.path.join(output_folder, f'mask_{images[i][:-4]}.jpg')
    cv2.imwrite(output_path, mask)
```

Les masques seront utilisés par la suite dans le modèle.

4. Visualisation des images dentaires avec les points d'angles :

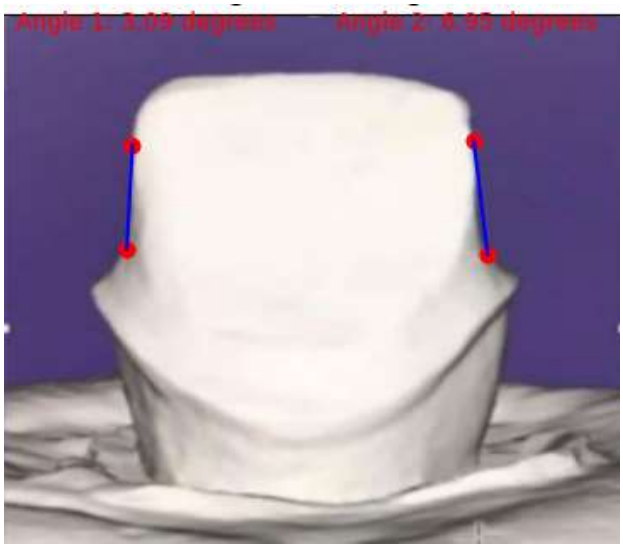
Le processus de prétraitement des données implique une étape cruciale de visualisation, permettant une inspection détaillée des images dentaires annotées avec leurs points d'angle.

On a utilisé la bibliothèque Matplotlib conjointement avec OpenCV pour créer des représentations visuelles informatives. Chaque image est chargée et affichée avec des points d'angle marqués en rouge, mettant en évidence les caractéristiques géométriques essentielles.

Des lignes bleues sont tracées entre les points d'angle, offrant une perspective claire sur les relations spatiales entre ces points. De plus, les angles formés par ces lignes sont calculés et affichés.

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

    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')

    angs=[]
    for j in [0,2]:
        x_coords = [angle[j][0], angle[j + 1][0]]
        y_coords = [angle[j][2], angle[j + 1][2]]
        plt.plot(x_coords, y_coords, color='blue')
        line1_start = np.array([angle[j][0], angle[j][2]])
        line2_start = np.array([angle[j+1][0], 448])

        line1_end = np.array([angle[j+1][0], angle[j+1][2]])
        line2_end = np.array([angle[j+1][0], 0])

        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)
```

Les résultats s’affiche comme suit :



5. Augmentation des données :

Dans le cadre de l'optimisation des performances de notre modèle de prédiction des angles de convergence dentaire, une étape cruciale réside dans l'augmentation

des données d'entraînement. Pour ce faire, nous avons appliqué une augmentation d'image utilisant la bibliothèque TensorFlow et son module ImageDataGenerator. Cette approche innovante permet de générer des variations significatives des images originales en appliquant des transformations telles que la rotation, le décalage horizontal et vertical, ainsi que des retournements horizontaux et verticaux. Ces variations enrichissent la diversité des données d'entraînement, permettant au modèle d'apprendre des motifs plus robustes et de généraliser efficacement. Les images augmentées, ainsi que leurs masques correspondants, sont désormais prêts à être intégrés dans le processus d'entraînement, contribuant ainsi à renforcer la capacité prédictive du modèle face à une gamme plus étendue de scénarios cliniques.

Le code utilisé est ci-dessous :

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

    augmented_image_batch = data_generator.flow(image_batch, batch_size=1, shuffle=False)

    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)
augmented_images = np.array(augmented_images)
augmented_masks = np.array(augmented_masks)

print(f'Augmented Images Size : {augmented_images.size}, Shape : {augmented_images.shape}')
print(f'Augmented Masks Size : {augmented_masks.size}, Shape : {augmented_masks.shape}')
```

L'objet ImageDataGenerator est créé et configuré avec différents paramètres d'augmentation, notamment :

- **rotation_range=20:** Rotation aléatoire dans la plage de ±20 degrés.
- **width_shift_range=0.1:** Décalage horizontal aléatoire dans la plage de ±10% de la largeur de l'image.
- **height_shift_range=0.1:** Décalage vertical aléatoire dans la plage de ±10% de la hauteur de l'image.
- **horizontal_flip=True:** Retournement horizontal aléatoire.
- **vertical_flip=True**: Retournement vertical aléatoire.

- **fill_mode='constant':** Remplissage des pixels nouvellement créés avec une valeur constante.
- **cval=0:** Valeur constante utilisée pour le remplissage.

IV. Le modele de prediction des angles :

Le modèle est implémenté avec TensorFlow et Keras et est spécialement adapté pour les tâches de segmentation d'images. La première moitié du modèle, appelée l'encodeur, est responsable de la capture des caractéristiques pertinentes des images dentaires en utilisant des couches de convolution et des opérations de max pooling. Dans la deuxième moitié, le décodeur, les caractéristiques extraites sont utilisées pour générer des masques de segmentation précis. Des opérations d'upsampling et de concaténation sont employées pour conserver les informations spatiales cruciales. Enfin, la couche de sortie utilise une fonction d'activation sigmoïde pour produire un masque binaire indiquant les zones d'intérêt pour la prédiction des angles de convergence. Ce modèle U-Net est ensuite compilé avec l'optimiseur Adam et la perte binaire_crossentropy, prêt à être entraîné sur les données augmentées pour améliorer sa précision et sa généralisation.

Le code utilisé est ci-dessous :

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


    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)

    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

input_shape = images[0].shape
model = build_unet(input_shape)

augmented_masks_single_channel = augmented_masks[..., 0:1]

model.compile(optimizer='adam', loss='binary_crossentropy')
```

La fonction build_unet crée l'architecture du modèle U-Net. Dans la phase d'encodage, deux couches de convolution suivies d'une opération de max pooling sont utilisées. Ensuite, dans la phase de décodage, une opération d'upsampling, une convolution, et une concaténation avec la sortie de l'encodage sont effectuées avant deux couches de convolution finale.

Les masques augmentés sont adaptés pour avoir un seul canal, assurant la compatibilité avec la configuration de sortie du modèle.

Le modèle est compilé avec l'optimiseur Adam et la fonction de perte binaire_crossentropy, prêt à être entraîné sur des données d'entraînement.

L'entrainement du modèle :

```
Epoch 1/10
5/5 [==============================] - 434s 83s/step - loss: 8.1781 - val_loss: 4.4443
Epoch 2/10
5/5 [==============================] - 381s 76s/step - loss: 2.4653 - val_loss: 3.3567
Epoch 3/10
5/5 [==============================] - 379s 77s/step - loss: 1.5102 - val_loss: 1.1518
Epoch 4/10
5/5 [==============================] - 390s 79s/step - loss: 0.7155 - val_loss: 0.9183
Epoch 5/10
5/5 [==============================] - 384s 75s/step - loss: 0.6190 - val_loss: 0.7720
Epoch 6/10
5/5 [==============================] - 372s 74s/step - loss: 0.4876 - val_loss: 1.1563
Epoch 7/10
5/5 [==============================] - 379s 76s/step - loss: 0.2955 - val_loss: 1.4839
Epoch 8/10
5/5 [==============================] - 376s 75s/step - loss: -0.0531 - val_loss: 2.0707
Epoch 9/10
5/5 [==============================] - 378s 75s/step - loss: -1.3355 - val_loss: 4.5477
Epoch 10/10
5/5 [==============================] - 367s 74s/step - loss: -4.3945 - val_loss: 8.2508
Out[101]: <keras.src.callbacks.History at 0x1afd5811210>
```

L'évaluation du modèle :

On a utilisé la méthode evaluate :

```
eval_loss = model.evaluate(val_images, val_masks)
print("Evaluation Loss:", eval_loss)
```

Les résultats sont comme suit :

```
Evaluation Loss: 0.6981333494186401
```

Conclusion

Ce TP nous a permis de créer un modèle pour la prédiction des angles en suivant plusieurs étapes. Le prétraitement des données a constitué une étape cruciale, permettant une extraction efficace des informations pertinentes à partir d'annotations XML, facilitant ainsi la création d'un ensemble de données structuré.

L'augmentation des données a été une stratégie essentielle pour améliorer la capacité de généralisation du modèle. En appliquant des transformations aléatoires aux images et aux masques, nous avons considérablement élargi la diversité des exemples d'entraînement, renforçant ainsi la robustesse du modèle face à différentes variations dans les images dentaires.

La création du modèle U-Net personnalisé a été guidée par une conception soigneuse des couches de convolution et des opérations d'up-sampling. Les choix architecturaux ont permis au modèle de capturer efficacement les caractéristiques complexes des images dentaires, conduisant à des prédictions précises des angles de convergence.

Ces trois étapes, le prétraitement des données, l'augmentation des données, et la création du modèle, ont été interconnectées pour former un processus holistique menant à des résultats significatifs.