جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

Université Hassan 1 er

Ecole national des sciences applique Berrechid

ENSA Ecole Nationale des Sciences Appliquées BERRECHID

Compte rendu du TP : Analyse d'annotations d'images de dents et mise en place d'un modèle pour la prédiction des angles de dépouille

Module : Fouille de données

Filière & Semestre : CI-ISIBD/S9

Année Universitaire : 2023-2024

Encadrant Pédagogique :

Pr. Hrimech Hamid

Réalisé par :

ESSAADI NIHAD

1. Analyse des données du DataFrame

Chargement des fichiers XML contenant des annotations d'images de dents. Ces annotations incluent des informations sur les dimensions de l'image, ainsi que les coordonnées des coins de chaque dent. Ces données sont ensuite stockées dans un DataFrame Pandas pour une manipulation plus aisée.

Le DataFrame est analysé à l'aide de la méthode `describe`, fournissant des statistiques descriptives telles que la moyenne, l'écart-type, le minimum, le 25e percentile, la médiane, le 75e percentile et le maximum pour chaque colonne. Ces statistiques offrent un aperçu général des données.

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

_dir = 'est'
os.chdir(_dir)

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_names = []
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(_dir, 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' column value
    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.describe)
```

2. Vérification des erreurs dans les coordonnées des coins

Le code vérifie s'il y a des erreurs dans les coordonnées des coins en comparant les valeurs y_min et y_max. Aucune erreur n'est détectée, comme indiqué par les DataFrames vides renvoyés pour chaque coin.

```
print([df[df[f'coin_{i}'].apply(lambda x: x[2]) != df[f'coin_{i}'].apply(lambda x: x[3])] for i in range(1, 5)])
print(df[df.height < 448])
```

3. Visualisation des images avec les points d'angle

Les images sont visualisées avec les points d'angle représentant les coins de chaque dent. Des lignes sont tracées entre les points pour visualiser les angles de dépouille. Les points d'angle sont affichés en rouge, les lignes entre les angles en bleu et une ligne verticale verte est tracée à partir du premier point d'angle vers le bas de l'image.

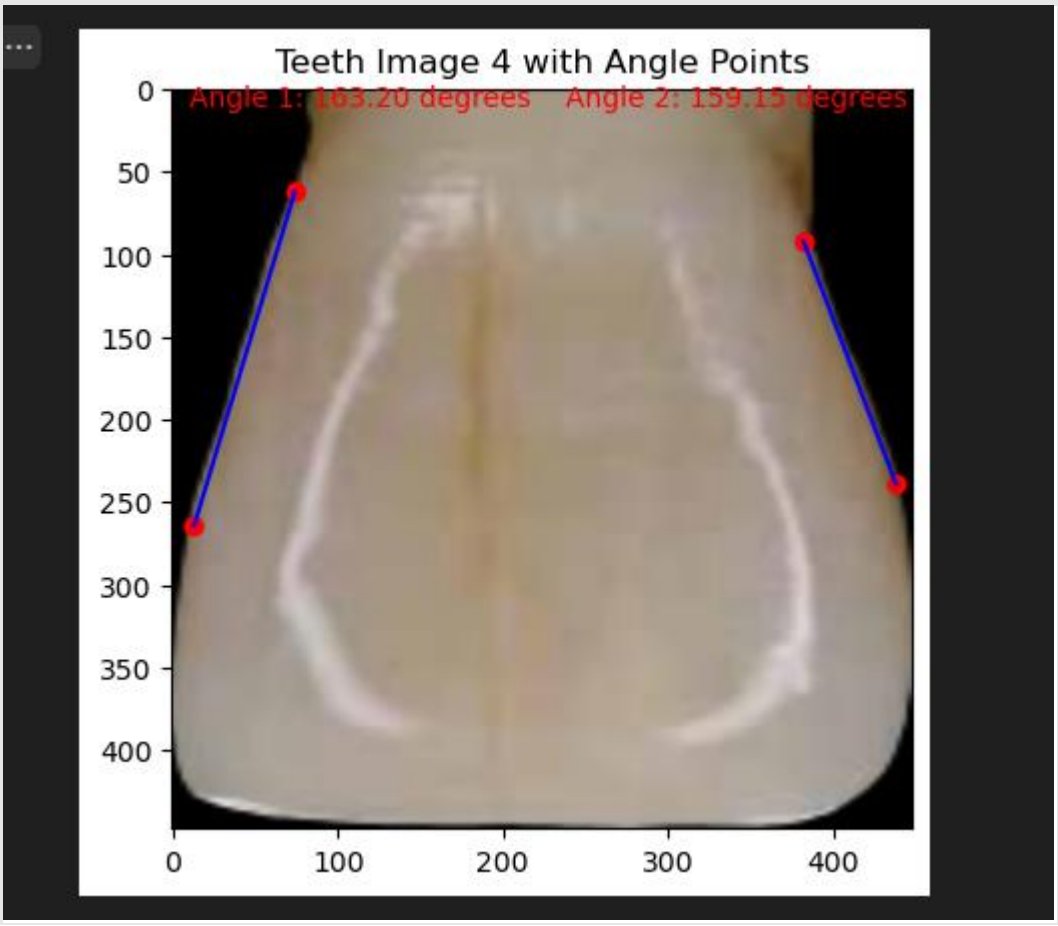
```
import matplotlib.pyplot as plt
import cv2

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 a line between the two points
```

```
    plt.plot()
# 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
    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()
```



4. Augmentation des données

Le code utilise TensorFlow pour créer un générateur d'images avec des augmentations telles que la rotation, le décalage horizontal et vertical, et la bascule horizontale et verticale. Ces augmentations sont appliquées aux images et aux masques correspondants.

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

```
# Create an image data generator with desired augmentations
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
)

# Generate augmented images and masks
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)

    # Apply data augmentation to images
    augmented_image_batch = data_generator.flow(image_batch, batch_size=1, shuffle=False)

    # Apply data augmentation to masks (use the same seed for consistency)
    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)


# Check the shapes of augmented images and 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}')
```

5. Construction du modèle :

Un modèle U-Net est construit pour prédire les angles de dépouille des images de dents. Le modèle comprend une architecture d'encodeur-décodeur, et il est compilé avec la fonction de perte binaire 'binary_crossentropy'. Le modèle est ensuite entraîné sur les images augmentées.

```
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

# Build the U-Net model
input_shape = images[0].shape
model = build_unet(input_shape)

# Modify the masks to have a single channel
augmented_masks_single_channel = augmented_masks[..., 0:1]

# Compile the model
model.compile(optimizer='adam', loss=')
```

6. Résultats de l'entraînement du modèle

Les résultats d'entraînement du modèle U-Net sont affichés, montrant les valeurs de perte pour chaque époque. La perte de validation est également surveillée pour évaluer la performance du modèle sur des données non vues.

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

```
1/1 [==============================]  - 40s 40s/step - loss: -31.2829
Evaluation Loss: -31.282934188842773
```