****

|  |  |  |
| --- | --- | --- |
| **Image Processing  Team project\_2 Final Report** | | |
| Project Name | 충치소년단 | |
| Team Number | 12 | |
| Team Name | 영처리와 상처리 | |
| Team Member | 20215697 권현민 | 20213883 박유나 |
| 20213710 채승훈 | 20214236 한예준 |

1. **Outline of program**

Program, through a graphical user interface (GUI), enables users to input oral images.

Upon loading the oral image, the program performs image preprocessing using the functions of enhancing (intensifying the color of the gums), blur (noise reduction for precise discrimination), and deleting gum (removing the gums).

Subsequently, it utilizes an implemented machine learning model to determine whether the corresponding tooth has cavities or not.

Through this process, users can easily assess oral health by loading oral images via the GUI and checking the results for the respective teeth.

1. **Design of program**

We wanted to separate the gum part and teeth part of the original oral image, because when we try to detect a cavity, the gum part of the image is not very useful. So we tried to delete the gum part of the image to get a higher accuracy for detection of cavities. We performed the following processes to reach our goal.

**<Image processing>**

|  |
| --- |
| def load\_image():  global image\_processing  file\_path = filedialog.askopenfilename() # 파일 선택 다이얼로그 열기  if file\_path:  print("이미지를 불러왔습니다:", file\_path)  # Open the image and resize if necessary  img = Image.open(file\_path)  # Check if resizing is needed  if img.width > max\_image\_size[0] or img.height > max\_image\_size[1]:  img.thumbnail(max\_image\_size)  img = ImageTk.PhotoImage(img)  print("파일 위치 : " + file\_path)  image\_processing = cv2.imread(file\_path)  label.config(image=img)  label.image = img  label.pack()  # 처리 버튼 및 판별 버튼 활성화  process\_button.config(state=tk.NORMAL)    # Calculate the window size with a margin  margin = 150  window\_width = min(img.width(), max\_image\_size[0]) + margin  window\_height = min(img.height(), max\_image\_size[1]) + margin  root.geometry(f"{window\_width}x{window\_height}") |

The “load\_image” function serves as the entry point for a program that loads oral images for classification. Through this function, images are loaded and displayed on a Tkinter window.

|  |
| --- |
| def process\_image():  global image\_processing  height, width, channels = image\_processing.shape  if image\_processing is None:  print("이미지가 없습니다.")  print("이미지 전처리 버튼이 눌렸습니다.")  print("이미지 처리하는 중...")  progress\_msg = "이미지 처리 중입니다..."  hsv\_image = convert\_to\_hsv(image\_processing)  print("enhancing 처리 중...")  enhance\_image(image\_processing)  enhanced = cv2.imread("enhanced\_image.jpg")  print("enhancing 완료...")  print("blur 처리 중...")  blur\_image(enhanced)  blurred = cv2.imread("blurred.jpg")  print("blur 완료...")  print("deleting gum 처리 중...")  delete\_gum(enhanced)  gum\_deleted = cv2.imread("image\_without\_gum.jpg")  print("deleting gum 완료...")  file\_path = 'image\_without\_gum.jpg' # 파일 선택 다이얼로그 열기  if file\_path:  print("이미지 전처리가 완료되었습니다.", file\_path)  # 이미지를 화면에 표시  image = Image.open(file\_path)  image = image.resize((width, height))  image = ImageTk.PhotoImage(image)  label.config(image=image)  label.image = image  label.pack()  detect\_button.config(state=tk.NORMAL)  root.lift() # Bring the main window to the front  detect\_button.config(state=tk.NORMAL) |

Using images loaded by “load\_image()” functions, to discern cavities, a sequence of preprocessing steps is performed using theconvert\_to\_hsv(), enhance\_image(), blur\_image(), delete\_gum() functions.

|  |
| --- |
| def convert\_to\_hsv(image):  # HSV로 변환전 BGR채널을 나눌 공간.  blue = np.zeros(image.shape[:2], np.uint8)  green = np.zeros(image.shape[:2], np.uint8)  red = np.zeros(image.shape[:2], np.uint8)  # HSV 저장 공간.  hsv\_img = np.zeros(image.shape, np.uint8)  # blue, green, red = cv2.split(image)  for row in range(image.shape[0]):  for col in range(image.shape[1]):  blue[row, col] = image[row, col, 0]  green[row, col] = image[row, col, 1]  red[row, col] = image[row, col, 2]  # cv2.cvtColor(cv2.COLOR\_BGR2HSV)  for row in range(image.shape[0]):  for col in range(image.shape[1]):  b = blue[row, col]  g = green[row, col]  r = red[row, col]  v = max(r, g, b)  s = ((v-min(r,g,b)) / v) \* 255  b\_ = b/255.0; g\_ = g/255.0; r\_ = r/255.0  num = ((r\_-g\_) + (r\_-b\_)) \* 0.5  den = np.sqrt((r\_-g\_)\*\*2 + (r\_-b\_) \* (g\_-b\_))  if den: theta = math.acos(num/den) \* (180/np.pi)  else: theta = 0  if b <= g: h = theta  else: h = 360-theta  hsv\_img[row, col, 0] = round(h/2)  hsv\_img[row, col, 1] = int(round(s))  hsv\_img[row, col, 2] = int(v)    return hsv\_img |

“convert\_to\_hsv” function converts an image received in RGB format to HSV. HSV separates color and brightness independently compared to RGB, making it simpler to select specific color ranges or apply filtering. This conversion is performed to facilitate subsequent operations.

|  |
| --- |
| def enhance\_image(image):  # 이미지를 HSV로 변환  hsv\_image = convert\_to\_hsv(image)  # 잇몸 부분(빨간색) 감지  lower\_red = np.array([0, 50, 50])  upper\_red = np.array([10, 255, 255])  red\_mask = inRange(hsv\_image, lower\_red, upper\_red)  lower\_pink = np.array([150, 50, 50])  upper\_pink = np.array([179, 255, 255])  pink\_mask = inRange(hsv\_image, lower\_pink, upper\_pink)  mask\_gum = red\_mask + pink\_mask  gum\_part = cv2.bitwise\_and(image, image, mask=mask\_gum)  gum\_part = np.clip(gum\_part + [20, 0, 30], 0, 255) # 빨간색 강도 증가  gum\_part = cv2.bitwise\_and(gum\_part, gum\_part, mask=mask\_gum)  tooth\_part = cv2.bitwise\_and(image, image, mask=~mask\_gum)  tooth\_part = cv2.bitwise\_and(tooth\_part, tooth\_part, mask=~mask\_gum)  gum\_part = gum\_part.astype(np.uint8)  tooth\_part = tooth\_part.astype(np.uint8)  enhanced\_image = cv2.bitwise\_xor(gum\_part, tooth\_part)  cv2.imwrite("enhanced\_image.jpg", enhanced\_image) |

convert\_to\_hsv() function is used to detect and emphasize the gum area in the oral image, which has been converted to HSV format. To detect the gum area, a mask is created for the regions corresponding to red and pink. The image is then separated into gum and tooth regions based on this mask. The intensity of the red color in the gum area is increased, and the enhanced gum and tooth regions are merged back together. The final result is saved as 'enhanced\_image.jpg'.

|  |
| --- |
| def blur\_image(image):  blurMask = np.array([[ 1, 2, 1],  [ 2, 4, 2],  [ 1, 2, 1]])  blurWeight = 16  height, width = image.shape[:2]  buffer = np.zeros(image.shape, np.uint8)  for i in range(1, height - 1):  for j in range(1, width - 1):  sumRed = sumGreen = sumBlue = 0  for k in range(3):  for l in range(3):  sumRed += blurMask[k][l] \* image[i + k - 1][j + l - 1][0]  sumGreen += blurMask[k][l] \* image[i + k - 1][j + l - 1][1]  sumBlue += blurMask[k][l] \* image[i + k - 1][j + l - 1][2]  sumRed = min(max(sumRed / blurWeight, 0), 255)  sumGreen = min(max(sumGreen / blurWeight, 0), 255)  sumBlue = min(max(sumBlue / blurWeight, 0), 255)  buffer[i][j] = [sumRed, sumGreen, sumBlue]  cv2.imwrite("blurred.jpg", buffer) |

Image is blurred using a 3x3 blur mask to reduce fine details and create a smoother appearance by averaging the surrounding colors and intensities. The result of applying this blur operation is saved as 'blurred.jpg'.

|  |
| --- |
| def delete\_gum(image):  hsv\_image = convert\_to\_hsv(image)  lower\_red = np.array([0, 50, 50])  upper\_red = np.array([10, 255, 255])  red\_mask = inRange(hsv\_image, lower\_red, upper\_red)  lower\_pink = np.array([150, 50, 50])  upper\_pink = np.array([179, 255, 255])  pink\_mask = inRange(hsv\_image, lower\_pink, upper\_pink)  mask\_gum = red\_mask + pink\_mask  # 잇몸 부분을 검은색으로 칠하기  image\_without\_gum = cv2.bitwise\_and(image, image, mask=~mask\_gum)  # 결과 저장  cv2.imwrite("image\_without\_gum.jpg", image\_without\_gum) |

The regions in the image corresponding to red and pink colors are filled with black. After removing the gum from this modified image, the result is saved as 'image\_without\_gum.jpg'.

|  |
| --- |
| def detect\_image():  global image\_processing  img\_path = 'enhanced\_image.jpg'    if img\_path:  img = image.load\_img(img\_path, target\_size=(64, 64))  img\_array = image.img\_to\_array(img) #배열화  img\_array = np.expand\_dims(img\_array, axis=0) #(1,64,64,3) 64\*64 크기의 RGB  img\_array /= 255.0 #정규화  #어레이로 변환한거 이제 모델 넘기기  # 모델을 사용하여 예측  predictions = loaded\_model.predict(img\_array)  # 충치 가능성 출력  print(predictions)  if predictions[0, 0] < 0.5:  # Cavity인 경우  print("Cavity!")  else:  # No Cavity인 경우  print("No Cavity!") |

“process\_image()” function provides the results regarding whether cavities are present in the processed image. It utilizes the trained cavity detection model, 'model.h5', which has been trained using the “Image\_load\_img” method. The function proceeds with the prediction using the model and prints the prediction result. If the predicted value is less than 0.5, it outputs "Cavity!" otherwise, it prints "No cavity!"

**<Model>**

Wrote the model code by referencing the open-source project at “<https://github.com/teeth-check>”

|  |
| --- |
| # 각 이미지에 대한 예측 및 분류  for img\_path in image\_paths:  # 이미지 불러오기 및 전처리  img = image.load\_img(img\_path, target\_size=(128, 128))  img\_array = image.img\_to\_array(img)  img\_array = np.expand\_dims(img\_array, axis=0)  img\_array /= 255.0 # 모델 학습 시 사용한 rescaling을 적용  # 모델에 이미지 주입 및 예측 수행  predictions = model.predict(img\_array)  # 예측 결과에 따라 분류  if predictions[0, 0] < 0.5:  # Cavity인 경우  shutil.copy(img\_path, os.path.join(output\_folder\_cavity, os.path.basename(img\_path)))  else:  # No Cavity인 경우  shutil.copy(img\_path, os.path.join(output\_folder\_no\_cavity, os.path.basename(img\_path))) |

After processing the aforementioned preprocessing-related code, it takes dental images and classifies them into decayed and normal teeth. This comprehensive preprocessing ensures that the dental images are optimized for the decay detection model, enhancing the accuracy of classification between decayed and normal teeth while also excluding unnecessary background information such as gums.

|  |
| --- |
| def build\_model(dropout\_rate=0.0):  model = Sequential()  model.add(Conv2D(filters=32,kernel\_size=(3,3),strides=(1,1),padding='same',input\_shape=(128,128,3),activation='relu'))  model.add(MaxPool2D(pool\_size=(2,2)))  model.add(Conv2D(filters=64, kernel\_size=(3,3),strides=(1,1),padding='same',activation='relu'))  model.add(MaxPool2D(pool\_size=(2,2)))  model.add(Conv2D(filters=64, kernel\_size=(3,3),strides=(1,1),padding='same',activation='relu'))  model.add(MaxPool2D(pool\_size=(2,2)))  model.add(Flatten())  model.add(Dense(512,activation='relu'))  model.add(Dropout(dropout\_rate))  model.add(Dense(2,activation='softmax'))  model.compile(  optimizer='adam',  loss='categorical\_crossentropy',  metrics=['accuracy'],  )  return model |

Convolutional Neural Network (CNN) model using the Keras Sequential API for image classification tasks. The model consists of three convolutional layers followed by max-pooling layers to extract features, a flattening layer to convert the data into a 1D array, a dense layer with 512 neurons and ReLU activation, a dropout layer for regularization, and a final dense layer with two neurons for binary classification using softmax activation. The model is compiled with the Adam optimizer, categorical crossentropy loss function, and accuracy as the evaluation metric.

1. **The project time line**

|  |  |
| --- | --- |
| ~11/13 | Selecting topic |
| 11/14~11/21 | Searching related open source |
| 11/22~11/25 | Developing CNN Model code |
| 11/26~11/30 | Code for Image Processing |
| 11/30 | Submit source code |

1. **Design process of group discussion**
2. **Team’s thoughts on this project**

권현민 : The process of using image processing techniques such as color adjustment and edge detection learned during the class, and applying them to preprocess dental images for training a cavity detection model, was meaningful. This comprehensive experience demonstrates how the fusion of image processing and machine learning can provide insights into solving realistic and meaningful problems.

박유나 : The process of applying image processing techniques, such as color adjustment and edge detection, learned during the class to preprocess dental images for training a cavity detection model, was immensely insightful. This comprehensive experience highlights how the fusion of image processing and machine learning can provide profound insights into solving practical and meaningful problems. In particular, the practical application of these methods in detecting cavities emphasizes the real-world applicability of the learned image processing techniques.

채승훈 : The process of modifying and training the model to fit the preprocessed images for improved accuracy in the project seems to have heightened my understanding of machine learning. Additionally, incorporating image processing to capture key features has enhanced my comprehension of the video processing pipeline, contributing to a more thorough understanding overall.

한예준 : During the image processing, I gained a sense of what aspects to focus on and which features of the images to select, especially in the context of discerning dental cavities. The various image processing techniques learned in class appeared applicable to real-life situations, and the practical use of these methods in detecting cavities became evident. Going through this process firsthand seemed to enhance my understanding, emphasizing the practicality of applying learned image processing methods to real-world issues.

1. **Result Images**

 