application.py

導入模組

```
import argparse
# import imp
# import logging
from msilib.schema import File
import os
import torch
from PIL import Image
from arch import deep_wb_model
import utilities.utils as utls
from utilities.deepWB import deep_wb
import arch.splitNetworks as splitter
from arch import deep_wb_single_task
import os
# import urllib.request
from flask import Flask, request, redirect, jsonify, send_file
from werkzeug.utils import secure_filename
from predict_result import dataInput_creatMask_rgb1, dataInput_creatMask_rgb2
# import io
import crop as crop
```

1. 建立Flask
2. 定義全域變數
3. 定義上傳圖像類型

```
app = Flask(__name__)
app.secret_key = "secret key"
is_anemia = False
analyze_data = 0
WB_modelname = 'net_awb.pth'
crop_modelname = 'model_0501.h5'
sclera_modelname = 'model_1102.h5'

# app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
```

```python
def get_args():
    parser = argparse.ArgumentParser(
        description='Changing WB of an input image.')
    parser.add_argument('--model_dir', '-m', default='./models',
                        help="Specify the directory of the trained model.", dest='model_dir')
    parser.add_argument('--input', '-i', help='Input image filename', dest='input',
                        default='../example_images/1.JPG')
    parser.add_argument('--task', '-t', default='AWB',
                        help="Specify the required task: 'AWB', 'editing', or 'all'.", dest='task')
    parser.add_argument('--target_color_temp', '-tct', default=None, type=int,
                        help="Target color temperature [2850 - 7500]. If specified, the --task should be 'editing'",
                        dest='target_color_temp')
    parser.add_argument('--mxsize', '-S', default=656, type=int,
                        help="Max dim of input image to the network, the output will be saved in its original res.",
                        dest='S')
    parser.add_argument('--save', '-s', action='store_true',
                        help="Save the output images",
                        default=True, dest='save')
    parser.add_argument('--device', '-d', default='cuda',
                        help="Device: cuda or cpu.", dest='device')
    return parser.parse_args()
```

--model_dir：指定模型之路徑，預設為"./models"
--input：指定輸入圖像的檔案名稱，預設為"../example_images/1.JPG"
--task：指定所需的任務，可以是 'AWB'、'editing' 或 'all'，預設為"AWB"
--target_color_temp：指定目標色溫
--mxsize：指定輸入圖像傳遞給網絡的最大維度，預設為"656"
--save：保存圖像，預設為"True"
--device：指定使用的設備，預設為"cuda"

1. 確定使用設備
2. 設置輸出圖像的大小和是否保存輸出
3. 載入模型並進行 AWB 處理
4. 處理輸入圖像

```python
def image_awb(image):
    args = get_args()
    if args.device.lower() == 'cuda':
        device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    else:
        device = torch.device('cpu')
    # Output_Image_Size
    S = args.S
    # (saving)true or false
    tosave = args.save
    # import AWB_model
    if args.task.lower() == 'awb':
        print(os.path.join('.\models', 'net_awb.pth'))
        if os.path.exists(os.path.join('.\models', 'net_awb.pth')):
            # load awb net
            net_awb = deep_wb_single_task.deepWBnet()
            net_awb.to(device=device)
            net_awb.load_state_dict(torch.load(os.path.join('./models', 'net_awb.pth'), map_location=device))
            net_awb.eval()
        elif os.path.exists(os.path.join('.\models', 'net.pth')):
            net = deep_wb_model.deepWBNet()
            net.load_state_dict(torch.load(
                os.path.join('./models', 'net.pth')))
            net_awb, _, _ = splitter.splitNetworks(net)
            net_awb.to(device=device)
            net_awb.eval()
        else:
            raise Exception('Model not '+os.path.join('.\models', 'net_awb.pth'))
    else:
        raise Exception(
            "Wrong task! Task should be: 'AWB', 'editing', or 'all'")
    # fn = path of image
    img = Image.open(image).convert('RGB')
    if args.task.lower() == 'awb':  # awb task
        out_awb = deep_wb(img, task=args.task.lower(),net_awb=net_awb, device=device, s=S)
        if tosave:
            global result_awb
            result_awb = utls.to_image(out_awb)
```

檢查圖像格類型

```
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

對應到上述之類型

```
    ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
```

定義POST路由

1. 指定裁切模型路徑(1貧血2黃疸)
2. 檢查有無接收到檔案
3. 獲取圖像檔案
4. 處理圖像-進行RGB分析
5. 處理分析結果並設置全域變數

dataInput_creatMask_rgb1&2
請參考PPT Page18

```python
@app.route('/', methods=['POST'])
def upload_image():
    model_where1 = './models/model_0501.h5'
    model_where2 = './models/model_1102.h5'

    if 'file' not in request.files:
        resp = jsonify({'message': 'No file part in the request'})
        resp.status_code = 400
        return resp
    file = request.files['file']
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        image_awb(file)
        result_awb.save("AWB_1.png", "png")
        analyze_data1 = dataInput_creatMask_rgb1("./AWB_1.png", model_where1)
        analyze_data2 = dataInput_creatMask_rgb2("./AWB_1.png", model_where2)

        print(analyze_data1)
        x = analyze_data1["r_minus_gb_avg_withoutmax"][1]["data"]
        print(analyze_data2)
        y = analyze_data2["r_minus_gb_avg_withoutmax"][1]["data"]

        global pedict_result_data
        pedict_result_data = 485 - 9195*x+64223*(x ** 2)-192665*(x ** 3)+211014*(x ** 4)

        global is_anemia
        if pedict_result_data < 11:
            is_anemia = True
            pedict_result_data = pedict_result_data
        else:
            is_anemia = False
            pedict_result_data = pedict_result_data
        # crop.crop_eyelids("./AWB_1.png")
        resp = jsonify({'message': 'File successfully uploaded'})
        resp.status_code = 201
        return resp
    else:
        resp = jsonify({'message': 'Allowed file types are  png, jpg, jpeg'})
        resp.status_code = 400
        return resp
```

定義GET路由(/crop)

- 回傳剪裁後"眼瞼"圖像

定義GET路由(/sclera)

- 回傳剪裁後"眼白"圖像

定義GET路由(/awb)

- 回傳白平衡圖像

```python
@app.route('/crop', methods=['GET'])
def download_crop_image1():
    resp = jsonify({'message': 'crop_image successfully downloaded'})
    resp.status_code = 201
    return send_file("AWB_1_crop_eyelids.png")

@app.route('/sclera', methods=['GET'])
def download_crop_image2():
    resp = jsonify({'message': 'crop_image successfully downloaded'})
    resp.status_code = 201
    return send_file("AWB_1_crop_sclera.png")

@app.route('/awb', methods=['GET'])
def download_AWB_image():
    resp = jsonify({'message': 'AWb_image successfully downloaded'})
    resp.status_code = 201
    return send_file("AWB_1.png")
```

定義GET路由(/data)

- 回傳所需資料

data:

1. 白平衡模型名稱
2. 貧血預測結果
3. 眼瞼剪裁模型名稱
4. 眼白剪裁模型名稱
5. 貧血預測數值

```python
@app.route('/data', methods=['GET'])
def get_data():

    data = {
        'WB_modelname': WB_modelname,
        'pred': is_anemia,
        'crop_modelname': crop_modelname,
        'sclera_modelname':sclera_modelname,
        'preddata': str(pedict_result_data),
    }
    resp = jsonify(data)
    resp.status_code = 201

    return resp
```

```python
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80,debug=True)
```

啟動Flask，設置端口80

crop.py

導入模組

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image as im
from keras.models import load_model
import cv2
```

定義二值化遮罩

```
def create_mask(pred_mask):
    pred_mask = tf.math.argmax(pred_mask, axis=-1)
    pred_mask = pred_mask[..., tf.newaxis]
    return pred_mask[0]
```

1. 讀取圖像&預處理圖像
2. 創建tensorflow dataset
3. 導入模型 model_5.h5並進行預測
4. 使用 create_mask 函數創建二值化的眼瞼遮罩
5. 根據遮罩和原始圖像創建新的圖像（只保留眼瞼部分）
6. 保存結果為 "AWB_1_crop_eyelids.png"

```python
def crop_eyelids(image_path):
    hi = tf.keras.utils.load_img(image_path, target_size=(128, 128), color_mode="rgb")
    hi = tf.keras.utils.img_to_array(hi)/255.0
    # ???
    dataset_2 = tf.data.Dataset.from_tensor_slices([hi])
    batched = dataset_2.batch(1)
    # ----------------------------------------------------------
    model = load_model('./models/model_5.h5')
    pred = model.predict(batched)  # Use model to predict
    pred = create_mask(pred)
    if pred.shape[-1] == 1:
        pred = np.squeeze(pred, axis=-1)
    # Scale the image to 0-255
    pred = (pred * 255).astype(np.uint8)
    # Save the image using OpenCV
    # print(pred.shape)
    px = np.zeros((128, 128, 3))
    for i in range(128):
        for j in range(128):
            if pred[i][j] != 0:
                px[i][j] = hi[i][j]
    out_px = tf.keras.utils.array_to_img(px)
    out_px.save("AWB_1_crop_eyelids.png")
```

1. 讀取圖像&預處理圖像
2. 創建tensorflow dataset
3. 導入模型 model_1102.h5並進行預測
4. 使用 create_mask 函數創建二值化的眼瞼遮罩
5. 根據遮罩和原始圖像創建新的圖像(只保留眼白部分)
6. 保存結果為 "AWB_1_crop_sclera.png"

```python
def crop_Sclera(image_path):
    hi = tf.keras.utils.load_img(image_path, target_size=(128, 128), color_mode="rgb")
    hi = tf.keras.utils.img_to_array(hi)/255.0
    dataset_2 = tf.data.Dataset.from_tensor_slices([hi])
    batched = dataset_2.batch(1)
    model = load_model('./models/model_1102.h5')
    pred = model.predict(batched)
    pred_mask = create_mask(pred)
    if pred_mask.shape[-1] == 1:
        pred_mask = np.squeeze(pred_mask, axis=-1)
    pred_mask = (pred_mask * 255).astype(np.uint8)
    px = np.zeros((128, 128, 3))
    for i in range(128):
        for j in range(128):
            if pred_mask[i][j] != 0:
                px[i][j] = hi[i][j]
    out_px = tf.keras.utils.array_to_img(px)
    out_px.save("AWB_1_crop_sclera.png")

if __name__ == "__main__":
    img_path = "C:/Users/DELL/Desktop/NTUT_project/AWB_Api/test7.png"
    crop_Sclera(img_path)
    #crop_eyelids(img_path)
```

predict_result.py

定義二值化遮罩

```
def create_mask(pred_mask):
    pred_mask = tf.math.argmax(pred_mask, axis=-1)
    pred_mask = pred_mask[..., tf.newaxis]
    return pred_mask[0]
```

1. 建構analyze_data
2. 依據file_name導入圖像並對圖像預處理
3. 依據model_where路徑導入模型
4. 使用 create_mask 函數建立遮罩
5. 轉為2維陣列

```
def dataInput_creatMask_rgb1(file_name , model_where):
    analyze_data = {"R_avg":[{'file_name': " " , 'data': 0 }] ,
                    "G_B_avg":[{'file_name': " " , 'data': 0 }],
                    "r_minus_gb_avg_withoutmax":[{'file_name': " " , 'data': 0 }]}
    try:
        input_data = tf.keras.utils.load_img( file_name, target_size=(128, 128), color_mode="rgb"
        input_data = tf.keras.utils.img_to_array(input_data)/255.0
        hsv_image = cv2.cvtColor((input_data * 255).astype(np.uint8), cv2.COLOR_BGR2HSV)
        dataset = tf.data.Dataset.from_tensor_slices([input_data])
        dataset_batched = dataset.batch(1)
        try:
            model = load_model(model_where, compile=False)
        except:
            return "Not find model"
        try:
            dataset_pred = model.predict(dataset_batched)  # Use model to predict
            dataset_pred = create_mask(dataset_pred)
        except:
            return "Predict error"
        if dataset_pred.shape[-1] == 1:
            dataset_pred = np.squeeze(dataset_pred, axis=-1)
```

```
    # Scale the image to 0-255
    dataset_pred = (dataset_pred * 255).astype(np.uint8)
    input_data_color = np.zeros((3, ))
    out_px = np.zeros((128, 128, 3))
    len = 0
    input_data_color_r_minus_gb = []
    for i in range(128):
        for j in range(128):
            if dataset_pred[i][j] != 0 :
                out_px[i][j] = input_data[i][j]
                h, s, v = hsv_image[i, j]
                if (s > 35 and v < 215):
                    # 平均
                    input_data_color +=  input_data[i][j]
                    len +=1
                    input_data_color_r_minus_gb.append(input_data[i][j][0] - (input_data[i][j][1]+input_data[i][j][2])/2)
    output_image = tf.keras.utils.array_to_img(out_px)
    output_image.save("AWB_1_crop_eyelids.png")
    q1 = np.percentile(input_data_color_r_minus_gb,25)
    q3 = np.percentile(input_data_color_r_minus_gb,75)
    iqr = q3 - q1
    lower_fence = q1 - 1 * iqr
    upper_fence = q3 + 1 * iqr
    output_color_r_minus_gb_value = 0
    output_color_r_minus_gb_len = 0
    for x in input_data_color_r_minus_gb:
        if x > lower_fence and x < upper_fence:
            output_color_r_minus_gb_value += x
            output_color_r_minus_gb_len += 1
    output_color_r_minus_gb_avg = output_color_r_minus_gb_value/output_color_r_minus_gb_len
    analyze_data["R_avg"].append({'file_name' : file_name,'data': input_data_color[0]/len})
    analyze_data["G_B_avg"].append( {'file_name' : file_name ,'data' : (input_data_color[1]/len + input_data_color[2]/len)/2})
    analyze_data["r_minus_gb_avg_withoutmax"].append( {'file_name' : file_name,'data' : output_color_r_minus_gb_avg})
    return  analyze_data
except:
    return "None file in this folder"
```

1. 初始化一些變數 input_data_color、out_px、len 、input_data_color_r_minus_gb
2. 將 input_data中不為0的像素值複製到out_px
3. 檢查 HSV 值，如果滿足(s > 35 & v < 215)，則計算平均值
4. 將圖像 out_px 保存到 AWB_1_crop_eyelids.png
5. 計算 R_avg、G_B_avg 和 r_minus_gb_avg_withoutmax
6. 更新並返還analyze_data

同理建置dataInput_creatMask_rgb2進行眼白切割與預測

- 注意圖像保存路徑要改