IMG & PDF OCR WEBUI

技術手冊

中央大學 MIAT 實驗室

鄧祺文

目錄

系統設計架構

✧ 功能需求：讀取 IMG 和 PDF 文件，將其中的內容透過光學字元辨識（OCR, Optical Character Recognition）進行擷取，最後輸出成可運用的文字檔案格式。

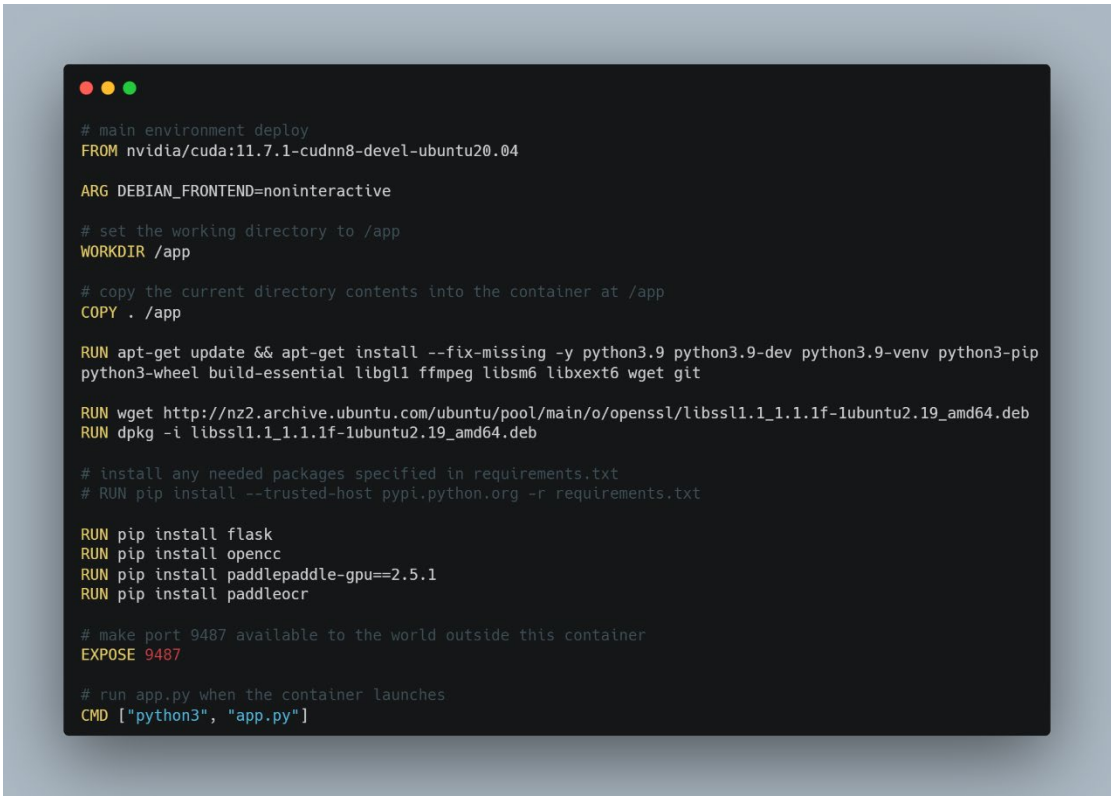
✧ 開發環境：函式 Python（IMG & PDF 之前處理、OCR 模型相關功能調用），後端 Flask 和 JavaScript（功能於網頁上的串接、跳轉，以及動態部件控制），前端 HTML（網頁實際展示 UI）

✧ 專案代碼開源地址：

https://github.com/toby0622/IMG-Optical-Character-Recognition-Tool

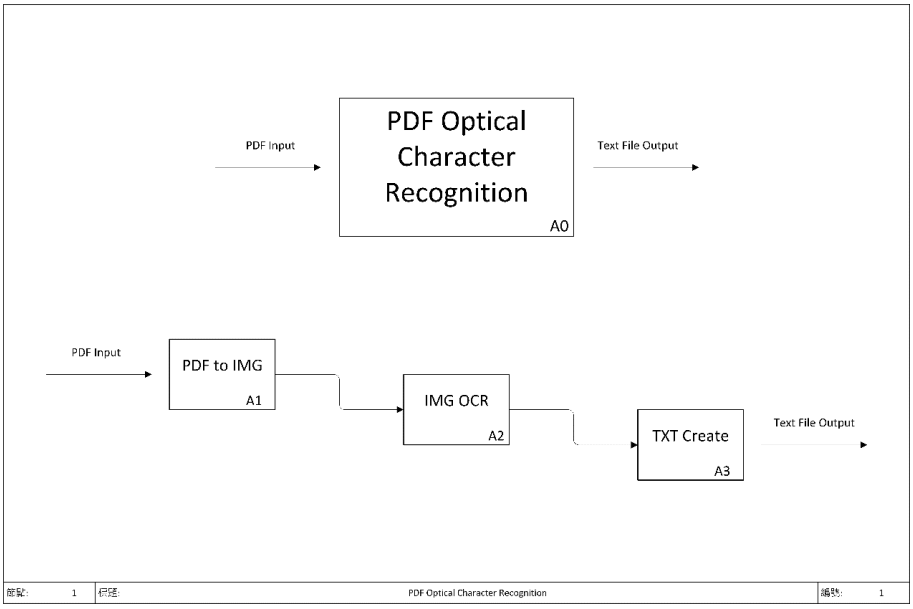
✧ Docker 專案自動化部屬：

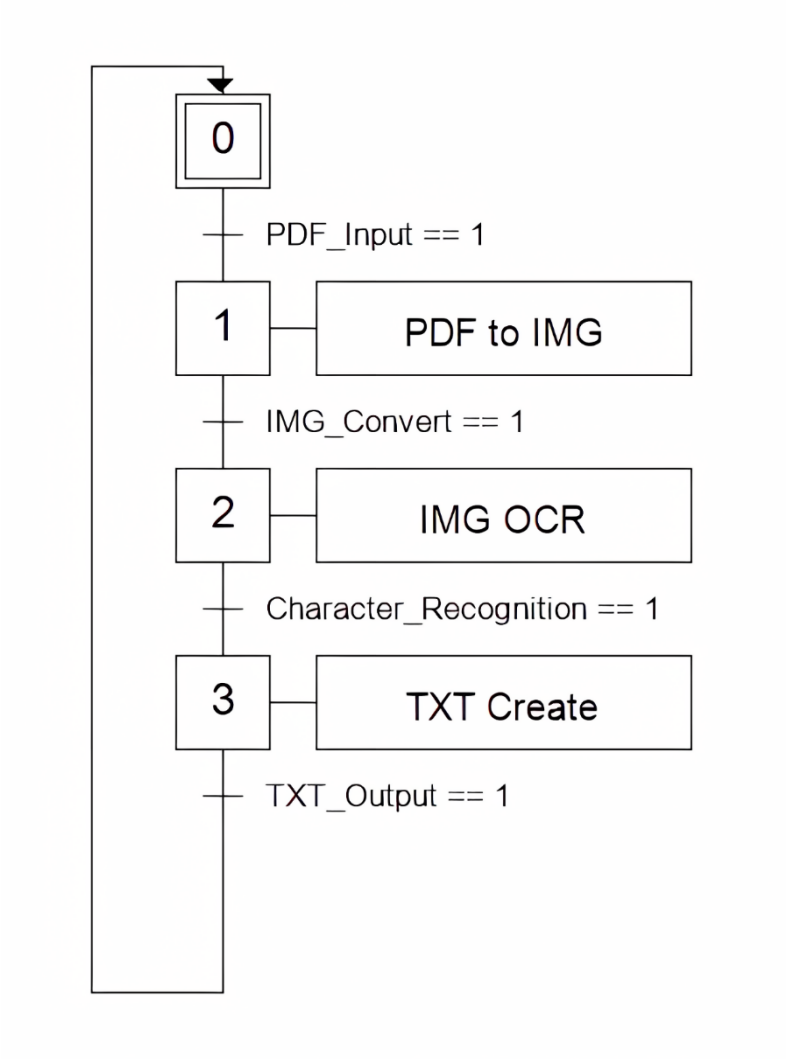
https://github.com/toby0622/IMG-Optical-Character-Recognition-Tool-Docker



```
# main environment deploy
FROM nvidia/cuda:11.7.1-cudnn8-devel-ubuntu20.04

ARG DEBIAN_FRONTEND=noninteractive

# set the working directory to /app
WORKDIR /app

# copy the current directory contents into the container at /app
COPY . /app

RUN apt-get update && apt-get install --fix-missing -y python3.9 python3.9-dev python3.9-venv python3-pip
python3-wheel build-essential libgl1 ffmpeg libsm6 libxext6 wget git

RUN wget http://nz2.archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.1f-1ubuntu2.19_amd64.deb
RUN dpkg -i libssl1.1_1.1.1f-1ubuntu2.19_amd64.deb

# install any needed packages specified in requirements.txt
# RUN pip install --trusted-host pypi.python.org -r requirements.txt

RUN pip install flask
RUN pip install opencc
RUN pip install paddlepaddle-gpu==2.5.1
RUN pip install paddleocr

# make port 9487 available to the world outside this container
EXPOSE 9487

# run app.py when the container launches
CMD ["python3", "app.py"]
```

✧ IDEF0 和 Grafcet





辨識模型

✧ 論文主題：

SVTR: Scene Text Recognition with a Single Visual Model

✧ 論文地址：

https://arxiv.org/abs/2205.00159

✧ 模型代碼開源地址：

https://github.com/PaddlePaddle/PaddleOCR

✧ 技術簡介：

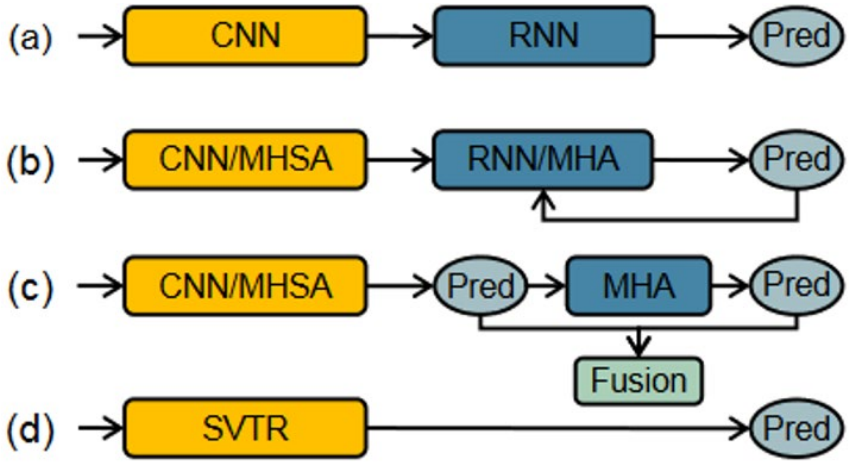


Figure 1: (a) CNN-RNN based models. (b) Encoder-Decoder models. MHSA and MHA denote multi-head self-attention and multi-head attention, respectively. (c) Vision-Language models. (d) Our SVTR, which recognizes scene text with a single visual model and enjoys efficient, accurate and cross-lingual versatile.

場景文字識別可以看作是一個從圖像映射到序列的任務。大多數的識別算法通常由兩個模塊構成，分別包含用於特征提取的視覺模塊，以及用於文本輸出的序列模塊。比如早期基於 CNN-RNN 的 CRNN，和現在一些基於注意力機制，進行自回歸式解碼

的算法，如上圖（圖一）所示。但是這樣設計出之雙階段算法的推理速度往往較慢，難以滿足工業應用的需求。因此該論文從推理速度和模型性能的雙重角度出發，提出了只由 Transformer 構成的純視覺模塊網絡 SVTR，在消費級顯示卡上達到了毫秒級的推理速度，並且參數量僅有 6 M。

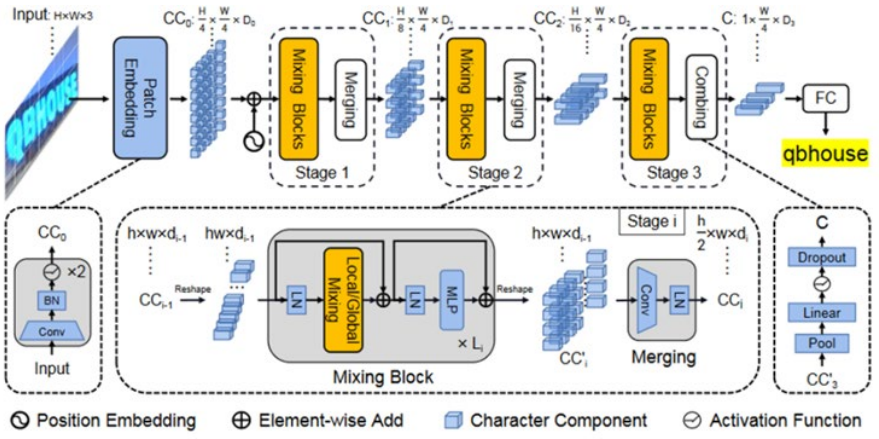


Figure 2: Overall architecture of the proposed SVTR. It is a three-stage height progressively decreased network. In each stage, a series of mixing blocks are carried out and followed by a merging or combining operation. At last, the recognition is conducted by a linear prediction.

上圖（圖二）為該論文所提出模塊網路 SVTR 的整體結構，采用類似於 SwinTransformer 的視覺模型和一個全連接層以及 CTC 解碼器進行文本序列預測。

首先和 ViT 類似，將輸入尺寸為 $H \times W \times 3$ 圖像按照 Patch 進行劃分，得到 $\frac{H}{4} \times \frac{W}{4} \times D_0$ Embeddings。本文采用的 Patch Embedding 操作和 ViT 中的有些許差異，其由兩層步距為 2，卷積核大小為卷積層 3×3，以及 BN 層構成。這樣不同的 Patch 之

間是存在著重疊的，如下圖（圖三）所示。經過 Patch Embedding 後的序列將經過一系列的 Stage，每一個 Stage 都由一系列的 Mixing Block 和 Merging Layer 構成。

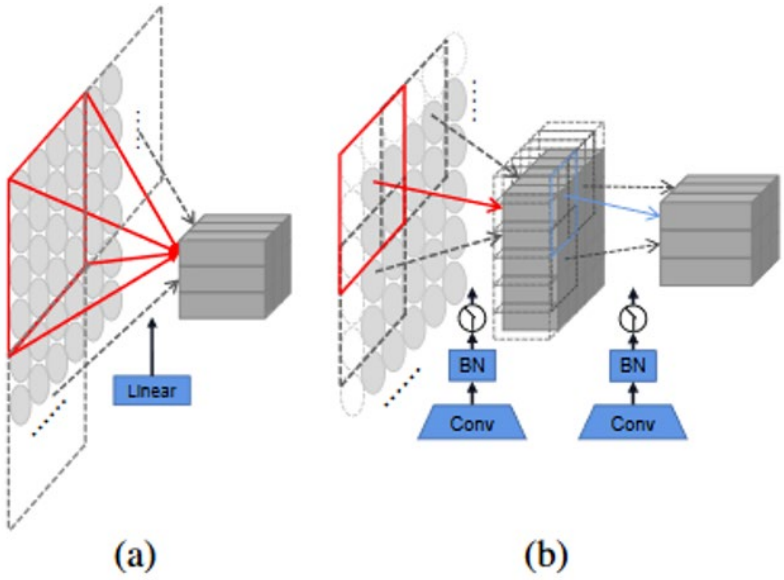


Figure 3: (a) The linear projection in ViT[Dosovitskiy *et al.*, 2021]. (b) Our progressive overlapping patch embedding.

作者認為文本識別需要兩種特征。第一種是局部特征，如筆畫特征。它編碼了字符的不同部分之間的形態特征和相關性。第二種是字符間的依賴性，如不同字符之間或文字與非文字成分之間的相關性。因此，作者設計了兩個混合模塊，即 Global Mixing 和 Local Mixing，通過使用不同大小感受的自注意層來實現。如下圖所示。Global Mixing 層本質上就是一個 Transformer Block，由一個多頭自注意層，一個 Layer Norm 層，以及一個 MLP 層構

成。通過自注意力機制的全局建模特性來進行全局字符建模。Local Mixing 則是采用了帶窗的自注意層，窗大小設置為了 7×11。

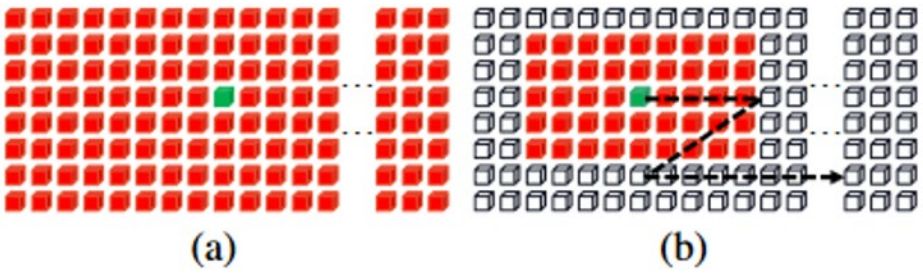


Figure 4: Illustration of (a) global mixing and (b) local mixing.

Merging 層扮演著將輸入序列進行下采樣的角色。其由高度方向步距為 2，寬度方向步距為 1，卷積核大小為 3×3 的卷積層構成。將輸入序列的尺寸由 $h \times w \times d_{i-1}$ 縮小為 $\frac{h}{2} \times w \times d_{i-1}$。同時每經過一次 Merging 層，序列的 Channel 維度也會增大，從而彌補在高度上的信息損失。

專案代碼（後端）

✧ app.py 關鍵部分

下圖一為 IMG 上傳之 Flask Route，一開始是各項變數的宣告和用以進行檔案上傳的 Web Request。得到檔案之後先對副檔名進行檢查，確認無誤後即可進行 OCR 之操作並對輸出進行調整，最後打包成所需之檔案格式（PDF 檔案同理，只是多做轉檔相關前處理）。

```python
@app.route('/uploadimg', methods=['GET', 'POST'])
def upload_file():
    global REVERSE_TOGGLE

    cc = OpenCC('s2twp')

    if request.method == 'POST':
        uploaded_files = request.files.getlist("file1[]")
        list_result = []
        ocr_final_result = str("")
        counter = 1
        total_images = len(uploaded_files)

    for file in uploaded_files:
        if file and allowed_file_img(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        else:
            return redirect(url_for("index"))

        ocr_result = image_ocr_match(os.path.join(app.config['UPLOAD_FOLDER'], filename),
counter)
        # chinese vertical written form
        if REVERSE_TOGGLE:
            ocr_result = written_reverse(ocr_result)
        else:
            ocr_result = written_default(ocr_result)

        for r in ocr_result:
            s2t = cc.convert(str(r[1][0]))
            list_result.append(s2t)
            ocr_final_result = ocr_final_result + str(s2t)

        progress_bar_calculation(counter, total_images)

        counter += 1

    ocr_final_result = remove_special_characters(ocr_final_result)

    ocr_json = json.dumps(
        dict(text=ocr_final_result),
        ensure_ascii=False)

    txt_export_web(ocr_final_result)
    json_export_web(ocr_json)

    return render_template('result.html',
                           ocr_final_result=ocr_final_result,
                           carousel_index=total_images)
```

下圖二則是下載功能的實現，通過使用 Flask 所提供之 send_file 功能，即可於網頁中將前述所提及之打包檔案進行下載。

```python
@app.route('/downloadtxt')
def download_txt():
    return send_file(
        'download/output.txt',
        mimetype='text/plain',
        download_name='result.txt',
        as_attachment=True)


@app.route('/downloadjson')
def download_json():
    return send_file(
        'download/output.json',
        mimetype='application/json',
        download_name='result.json',
        as_attachment=True)
```

✧ ocr.py 關鍵部分

下圖三為 OCR 實際進行辨識的過程，一開始先載入 OCR 辨識所需的 SVTR 模型（於 GitHub 上開源為 PaddleOCR）。接下來針對上傳或經過轉換的 PDF 圖檔使用 OpenCV 進行影像前處理來提升識別率。最後再將前處理完成之圖片送至辨識模型，即可得到文字輸出結果。

為了方便進行確認辨識情況，還額外進行了辨識視覺化的展示，實際產出的圖像可於本技術手冊之「網頁結果輸出」章節進行查看，其中將會包含輸入的文件圖片、模型判定出之辨識框，以及實際文字的輸出結果和該段文字辨識的置信率。

```python
def image_ocr_match(image_path, counter_number):
    process_start = datetime.datetime.now()  # process starting time

    cc = OpenCC('s2twp')

    ocr_model = PaddleOCR(use_angle_cls=True, lang="ch",
                          use_gpu=True, enable_mkldnn=True,
                          ocr_version="PP-OCRv4",
                          det_model_dir="models/det",
                          cls_model_dir="models/cls",
                          rec_model_dir="models/rec")

    image = cv2.imread(image_path, cv2.IMREAD_COLOR)
    gray_image = cv2.cvtColor(image, 7)
    inverted_image = cv2.threshold(gray_image, 95, 255, cv2.THRESH_BINARY_INV)[1]

    kernel = np.ones((2, 2), np.uint8)

    dilation_image = cv2.dilate(inverted_image, kernel, iterations=1)

    recognition_result = ocr_model.ocr(dilation_image)

    data = recognition_result[0]

        # result static
    visual = Image.open(image_path).convert('RGB')
    rec_boxes = [line[0] for line in data]
    rec_texts = [cc.convert(str(line[1][0])) for line in data]
    probability = [line[1][1] for line in data]
    im_show = draw_ocr(visual, rec_boxes, rec_texts, probability, font_path='font/Yozai-Regular.ttf')
    im_show = Image.fromarray(im_show)
    im_show.save('static/result' + str(counter_number) + '.jpg', quality=100)

    process_finish = datetime.datetime.now()  # process finishing time

    print('Page ' + str(counter_number) + ' OCR Process Time =', (process_finish -
process_start).seconds)

    return data
```

✧ export.py 關鍵部分

下圖四包含兩個函式，用以實際產出 app.py 中敘述之打包文件。本系統選擇兩個較為常見的檔案格式，一為 TXT，二為 JSON。作為一個目標為輸出文字的工具，TXT 可以十分直觀的對輸出文字進行複製或擷取；而 JSON 作為 Web API 之間傳遞最為通用的格式，預留

作未來的功能串接（如將 OCR 辨識出的文字通過其他 AI 進行潤色等），保留系統功能擴充之彈性。

```python
def txt_export_web(datafile):
    # process_start = datetime.datetime.now()  # process starting time

    storage_path = "download/output.txt"

    text_file = open(storage_path, 'w', encoding='UTF-8')

    # with open(output_folder + "SunHan" + ".txt", 'w', encoding='UTF-8') as textfile:
    #     textfile.write(str(datafile))

    text_file.write(datafile)
    text_file.close()

    # process_finish = datetime.datetime.now()  # process finishing time


def json_export_web(datafile):
    # process_start = datetime.datetime.now()  # process starting time

    storage_path = "download/output.json"

    text_file = open(storage_path, 'w', encoding='UTF-8')

    # with open(output_folder + "SunHan" + ".txt", 'w', encoding='UTF-8') as textfile:
    #     textfile.write(str(datafile))

    text_file.write(datafile)
    text_file.close()

    # process_finish = datetime.datetime.now()  # process finishing time
```

✧ function.py 關鍵部分

下圖五乃針對特殊符號進行去除，用以減少檔案格式相關錯誤。

```python
def remove_special_characters(text):
    # punctuation marks
    text = re.sub('[\uFF5E\uFF03-\uFF06\uFF08\uFF09\uFF1C-\uFF1E]', r'', text)

    text = re.sub('[~#$%&()<=>\"]', r'', text)

    return text
```

專案代碼（前端）

✧ index.html 關鍵部分

```
<div>
    <form class="form-inline pt-3 text-center" action="uploadimg" method="post" enctype="multipart/form-
data">
        <input class="form-control" type="file" multiple="" name="file1[]">
        <br/>
        <input class="form-control btn btn-primary" type="submit" value="IMG Upload">
    </form>
</div>

<div class="pt-3"></div>

<div>
    <form class="form-inline pt-3 text-center" action="uploadpdf" method="post" enctype="multipart/form-
data">
        <input class="form-control" type="file" name="file2[]">
        <br/>
        <input class="form-control btn btn-primary" type="submit" value="PDF Upload">
    </form>
</div>
```

上傳功能介面

✧ result.html 關鍵部分

```
<div id="resultcarousel" class="carousel slide" data-bs-ride="carousel">
    <p class="text-center pt-3">Optical Character Recognition Visualization(s)</p>

    <div class="carousel-inner" style="text-align: center">
        {% for i in range(1, carousel_index + 1) %}
            <div class="carousel-item {% if i == 1 %} active {% endif %}">
                <a data-fancybox="gallery" href="{{url_for('static', filename='result' + i|string +
'.jpg')}}">
                    <img src="{{url_for('static', filename='result' + i|string + '.jpg')}}"
                         style="object-fit: contain; object-position: center; height: 40vh">
                </a>
            </div>
        {% endfor %}
    </div>

    <button class="carousel-control-prev" type="button" data-bs-target="#resultcarousel" data-bs-
slide="prev"
            style="background: black; border-radius: 50%; opacity: 0.3; height: 50px; width: 50px; top:
50%">
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Previous</span>
    </button>

    <button class="carousel-control-next" type="button" data-bs-target="#resultcarousel" data-bs-
slide="next"
            style="background: black; border-radius: 50%; opacity: 0.3; height: 50px; width: 50px; top:
50%">
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Next</span>
    </button>
</div>
```

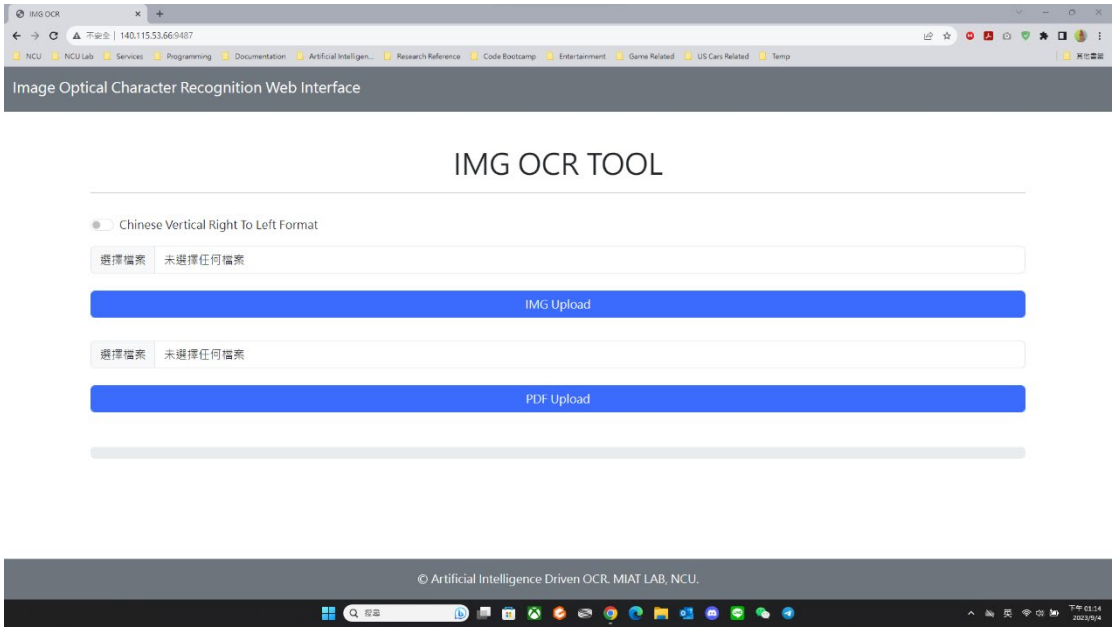
辨識結果視覺化燈箱

```
<div class="text-center pt-3 pb-3">
    <a href="{{url_for('download_txt')}}" type="button" class="btn btn-success text-center">Download
TXT</a>
    <a href="{{url_for('download_json')}}" type="button" class="btn btn-primary text-center">Download
JSON</a>
    <a href="{{url_for('index')}}" type="button" class="btn btn-danger text-center">New Process</a>
</div>
```

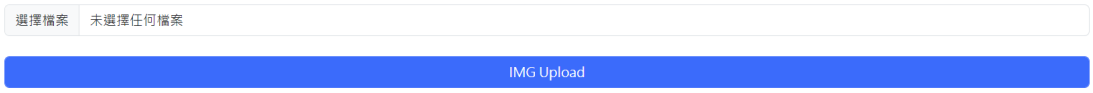
各項功能跳轉

網頁功能入口

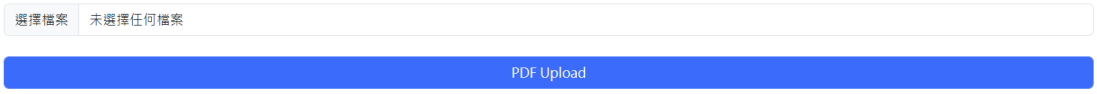
✧ 功能入口



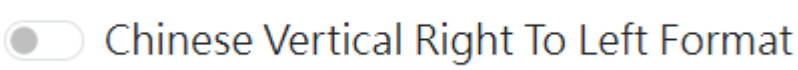
✧ 上半部選擇框：支持 JPG、JPEG、PNG



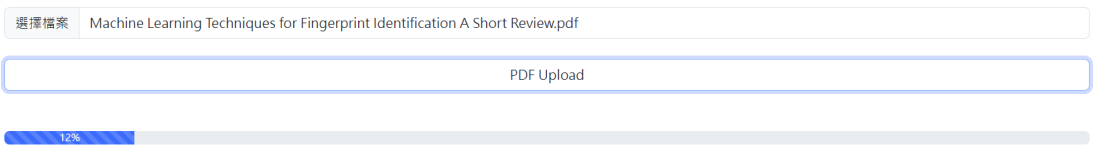
✧ 下半部選擇框：支持 PDF



✧ 圖像特殊排序：中文直式由右至左（預設標準為由上至下）

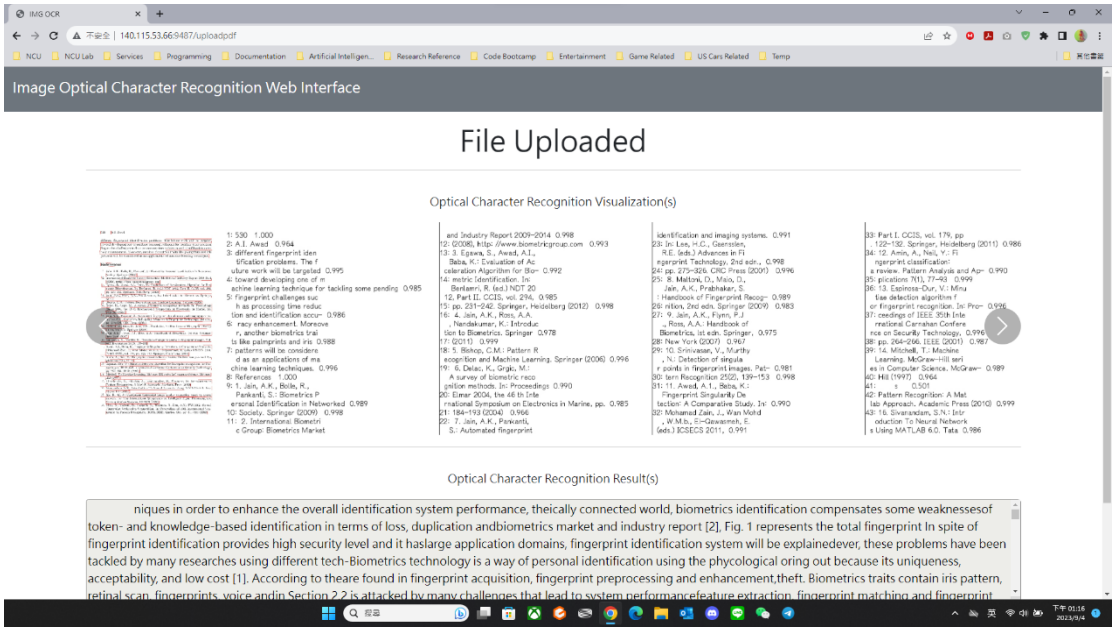


✧ 成功讀取資料後，可於下方看見處理進度條

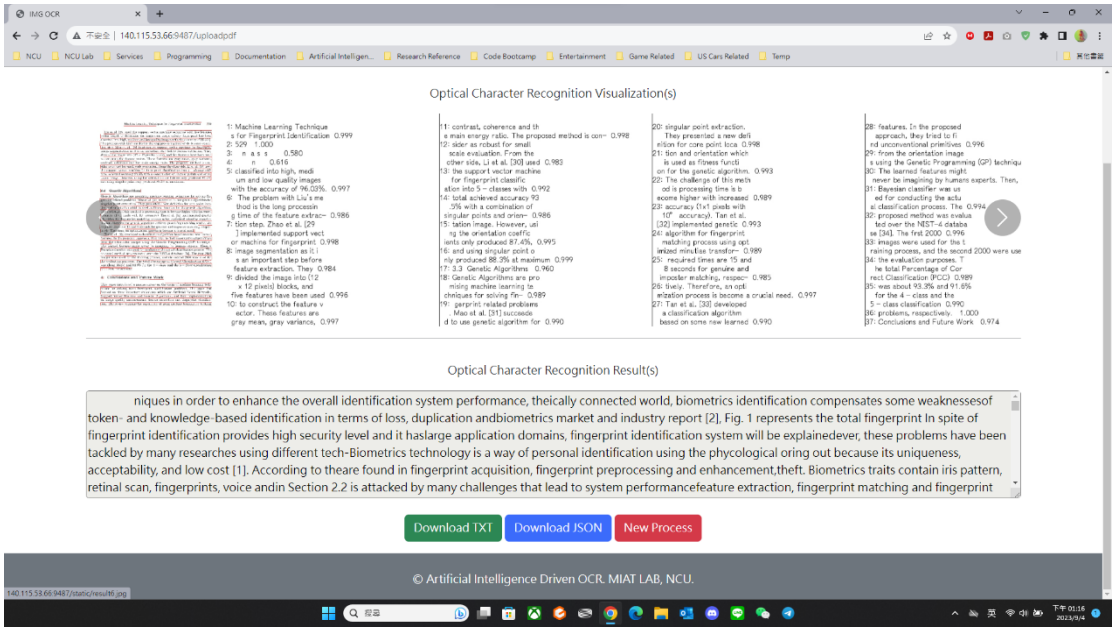


網頁結果輸出

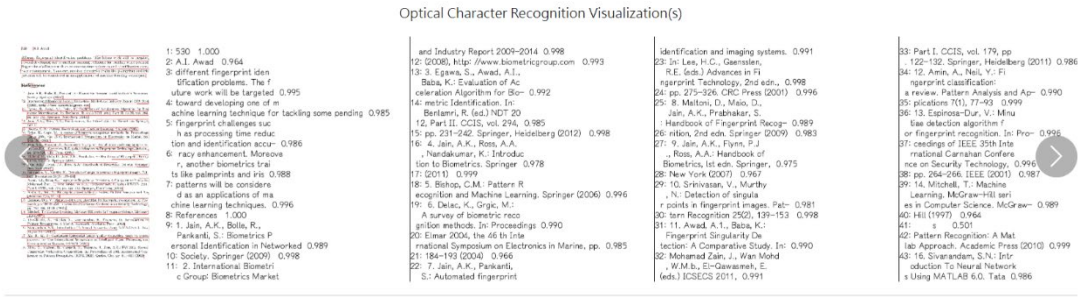
✧ 結果輸出頁面：上半部



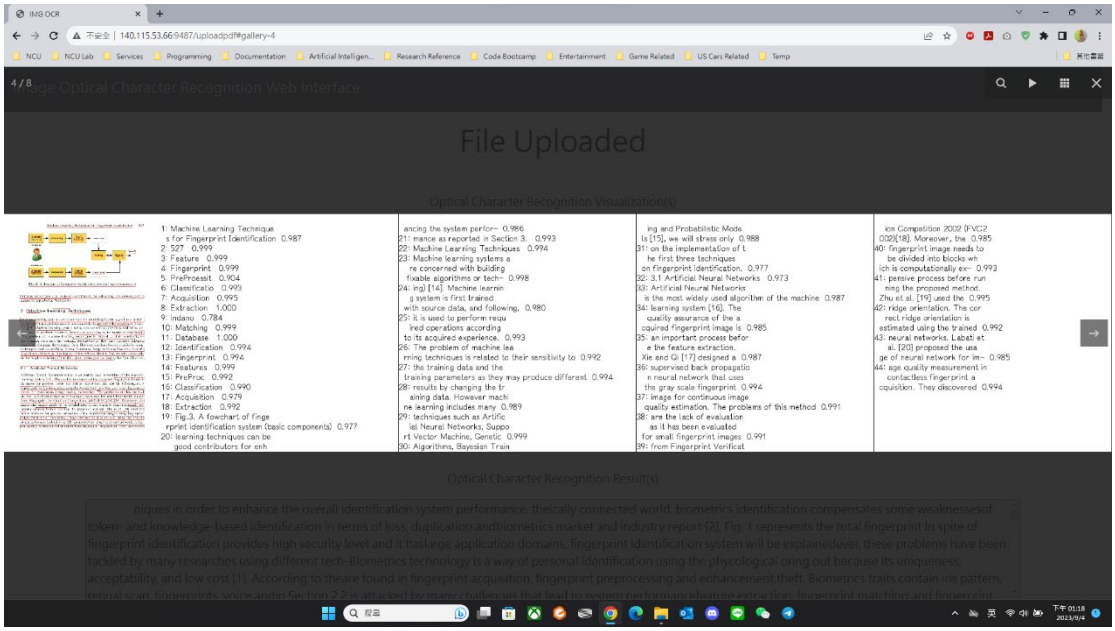
✧ 結果輸出頁面：下半部

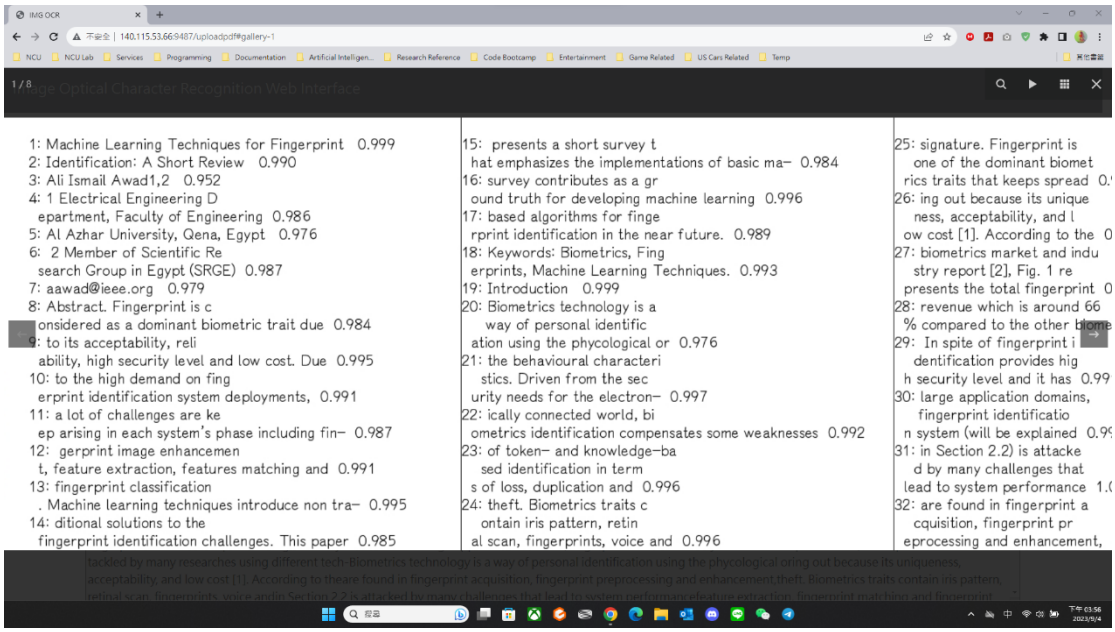


✧ 輸出視覺化：包含辨識框及輸出結果展示，左右按鍵可變換當前展示頁面

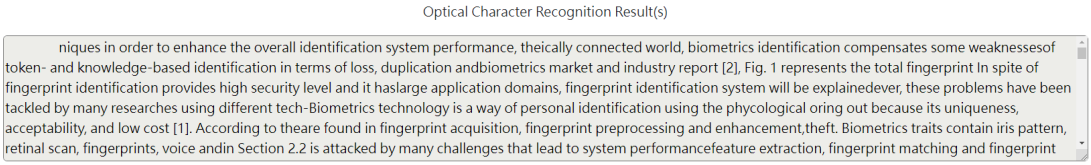


✧ 直接點擊圖片進入相簿模式，可以對圖片進行放大





✧ 完整文字輸出：包含所有頁面辨識出之完整文章



✧ 文章打包功能：包含 TXT 文字檔下載、API 串接用 JSON 檔案下載，以及重新開始新一輪識別