中央大學MIAT實驗室

鄧祺文

IMG & PDF OCR WEBUI

技術手冊

**目錄**

* 系統架構設計
* 辨識模型
* 專案代碼（後端）
* 專案代碼（前端）
* 網頁功能入口
* 網頁結果輸出

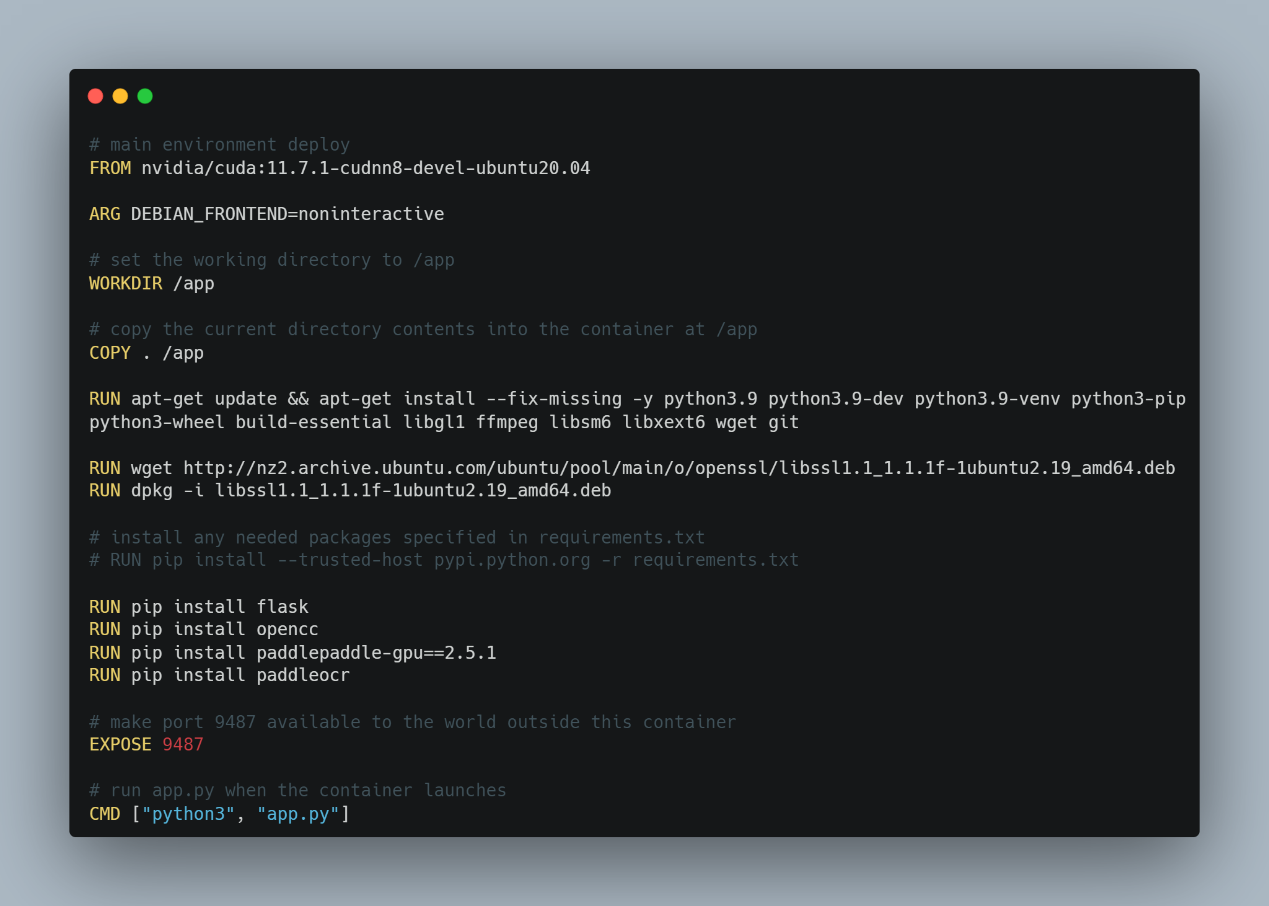
**系統設計架構**

* 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：

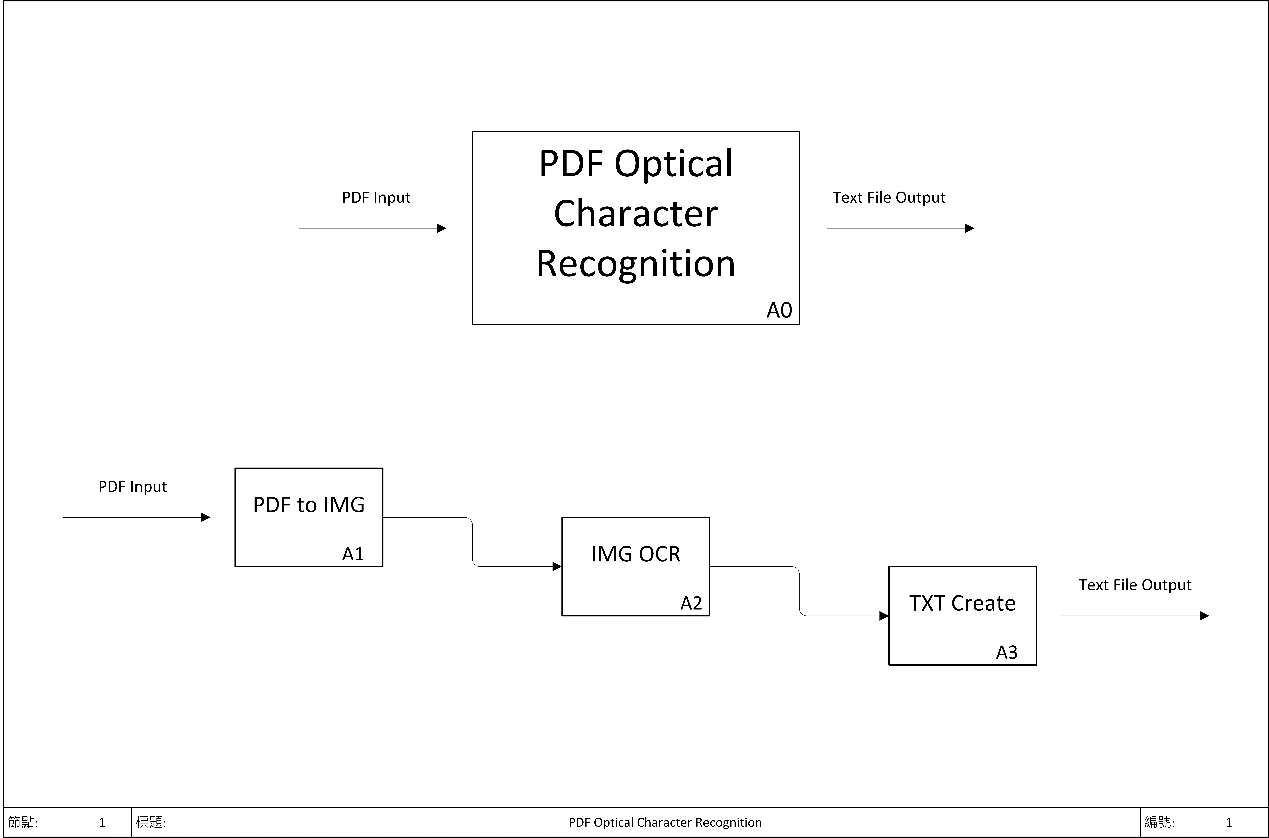
<https://github.com/toby0622/IMG-Optical-Character-Recognition-Tool>

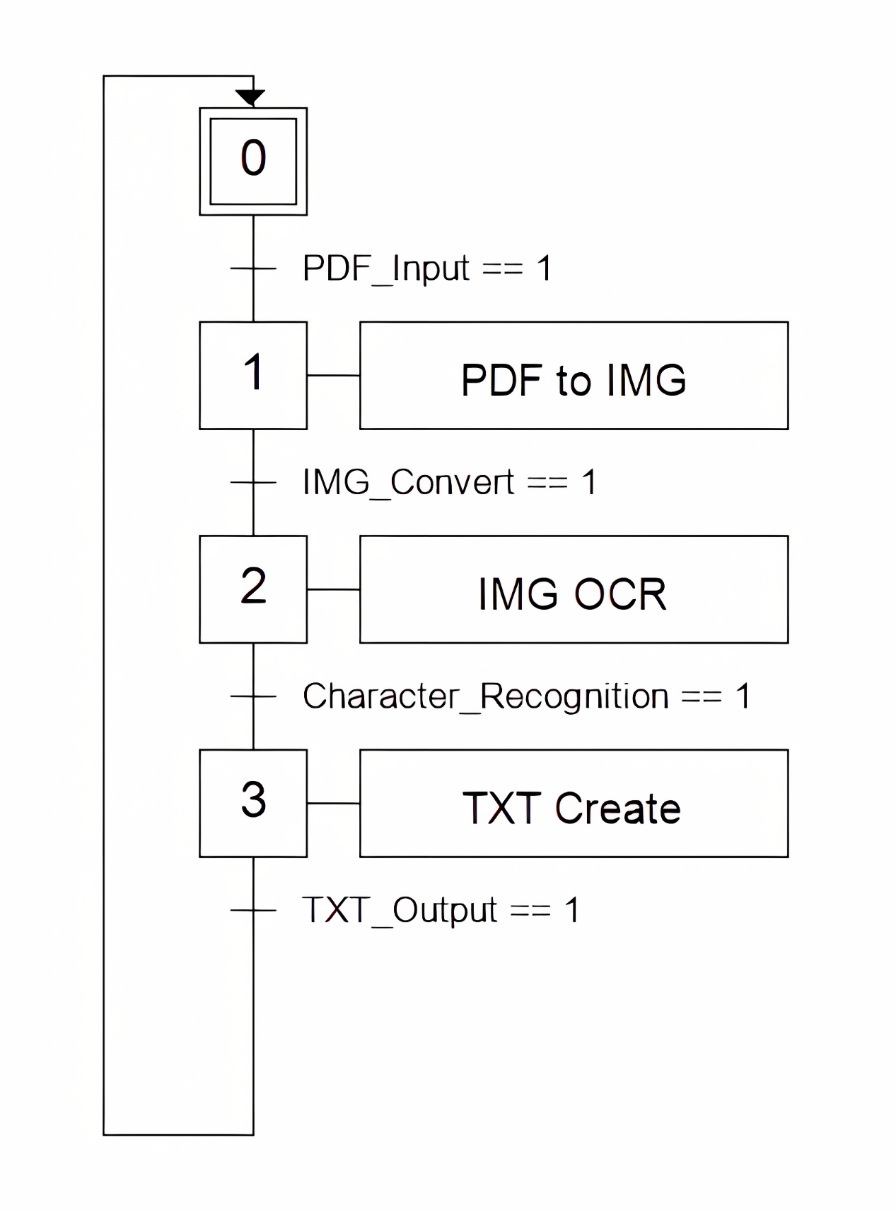
* Docker專案自動化部屬：

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



* IDEF0和Grafcet





**辨識模型**

* 論文主題：

SVTR: Scene Text Recognition with a Single Visual Model

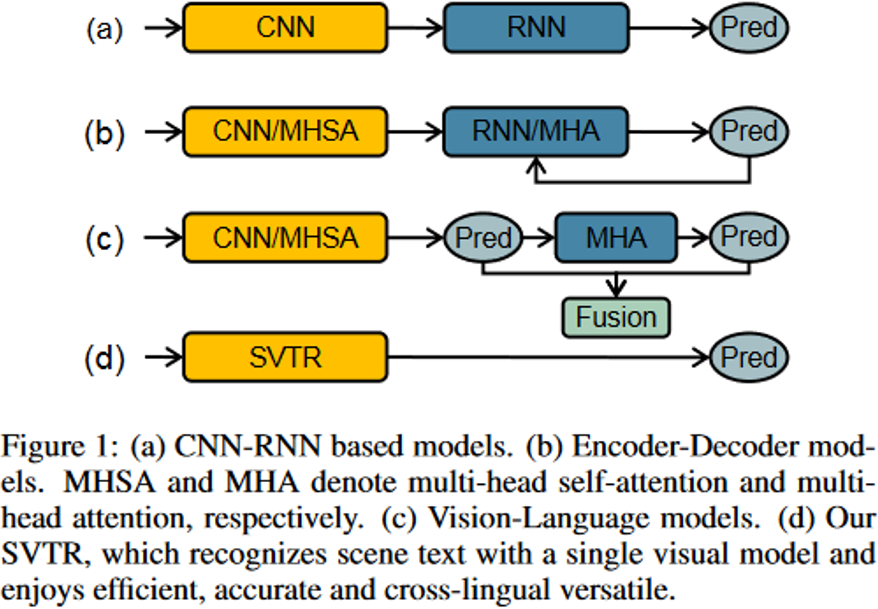
* 論文地址：

<https://arxiv.org/abs/2205.00159>

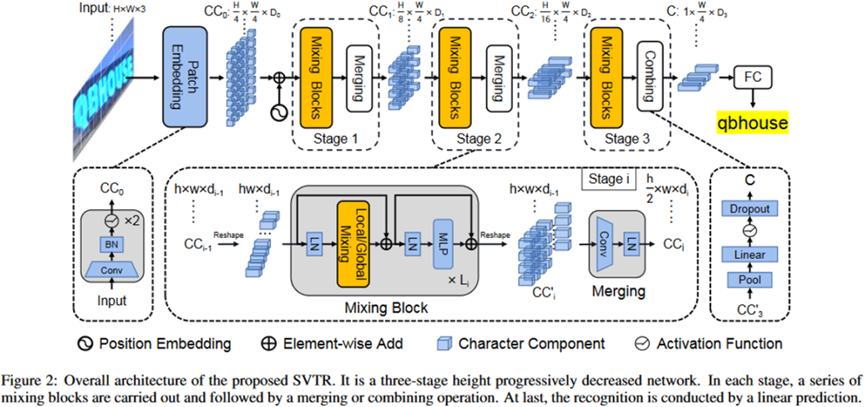
* 模型代碼開源地址：

<https://github.com/PaddlePaddle/PaddleOCR>

* 技術簡介：

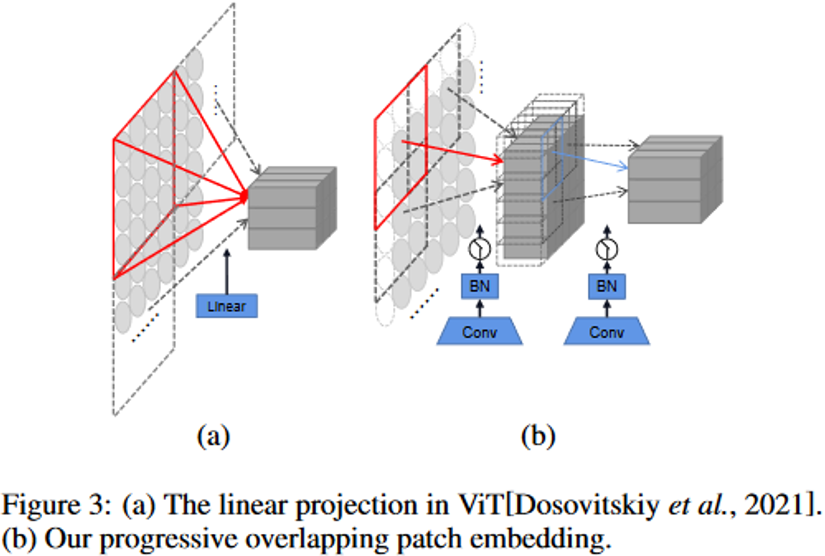


場景文字識別可以看作是一個從圖像映射到序列的任務。大多數的識別算法通常由兩個模塊構成，分別包含用於特征提取的視覺模塊，以及用於文本輸出的序列模塊。比如早期基於CNN-RNN的CRNN，和現在一些基於注意力機制，進行自回歸式解碼的算法，如上圖（圖一）所示。但是這樣設計出之雙階段算法的推理速度往往較慢，難以滿足工業應用的需求。因此該論文從推理速度和模型性能的雙重角度出發，提出了只由Transformer構成的純視覺模塊網絡SVTR，在消費級顯示卡上達到了毫秒級的推理速度，並且參數量僅有6 M。

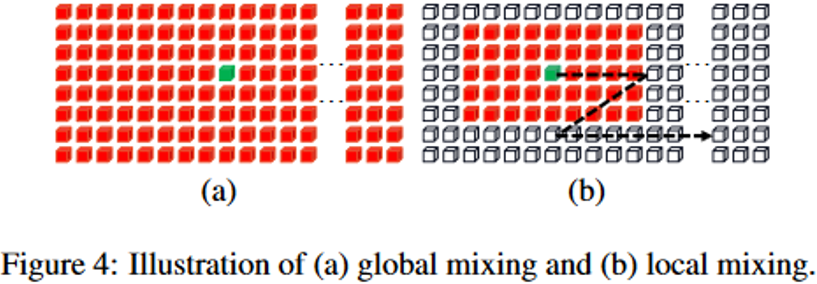


上圖（圖二）為該論文所提出模塊網路SVTR的整體結構，采用類似於SwinTransformer的視覺模型和一個全連接層以及CTC解碼器進行文本序列預測。

首先和ViT 類似，將輸入尺寸為 圖像按照Patch進行劃分，得到 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 ，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。



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 。

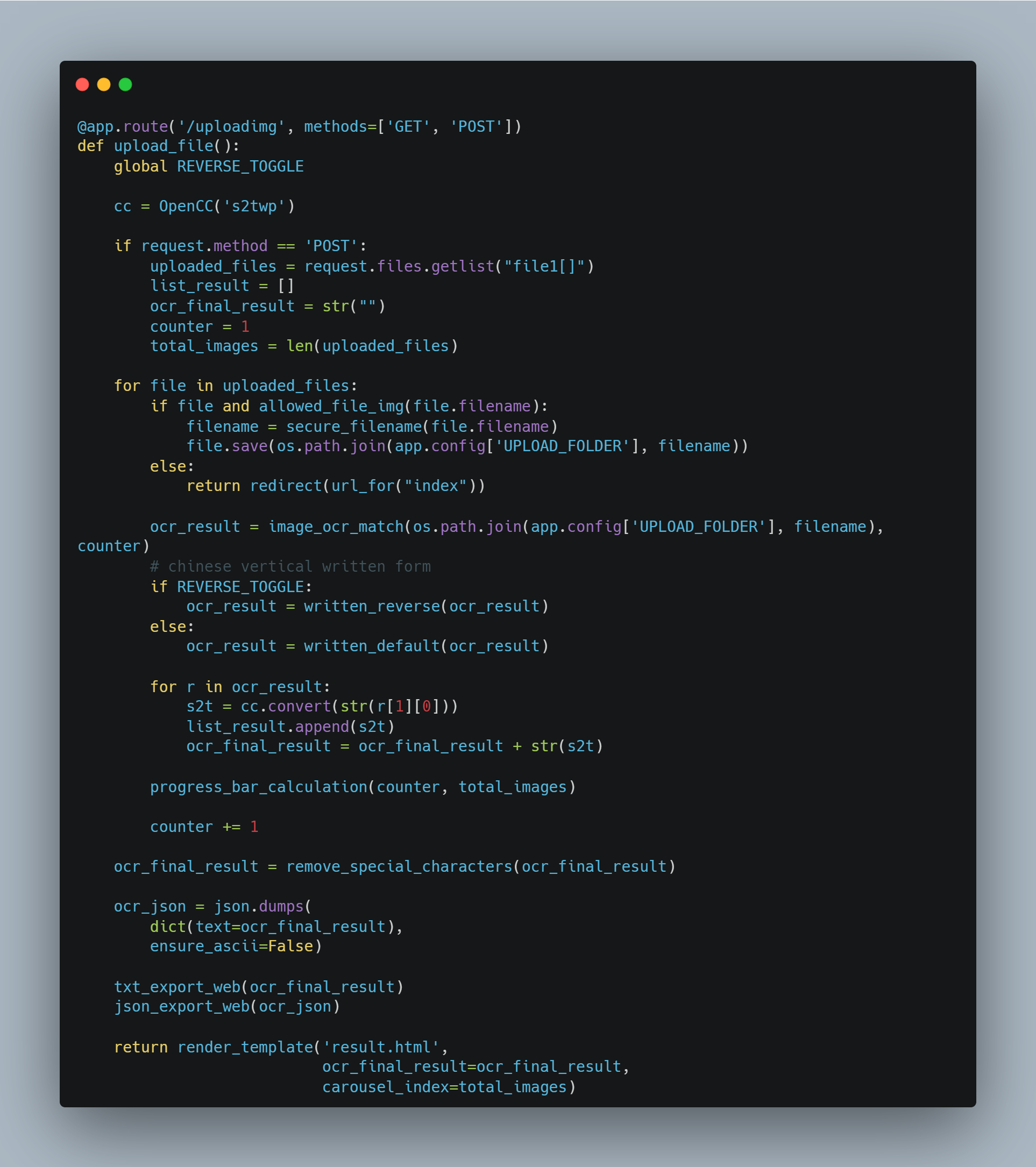


Merging層扮演著將輸入序列進行下采樣的角色。其由高度方向步距為2，寬度方向步距為1，卷積核大小為 的卷積層構成。將輸入序列的尺寸由 縮小為 。同時每經過一次Merging層，序列的Channel維度也會增大，從而彌補在高度上的信息損失。

**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）。

****

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。

****

* 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。



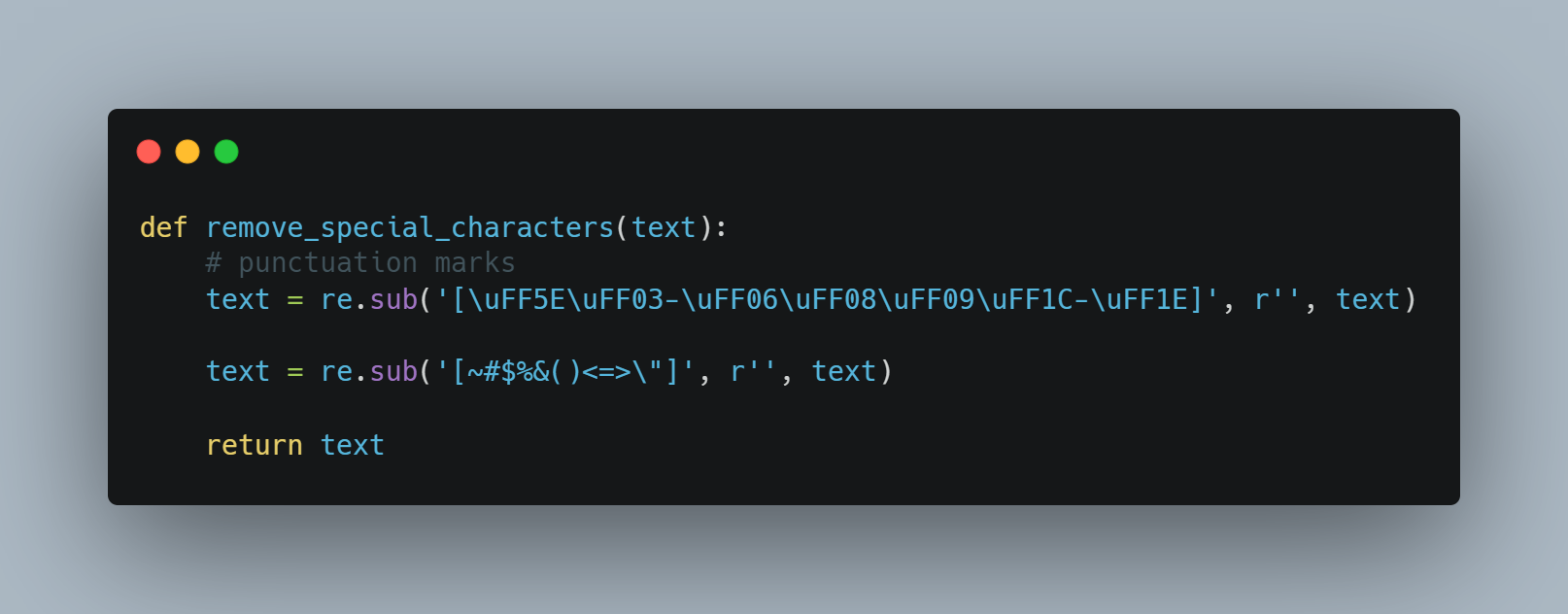
* export.py關鍵部分

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。

****

* 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。



**專案代碼（前端）**

* index.html關鍵部分

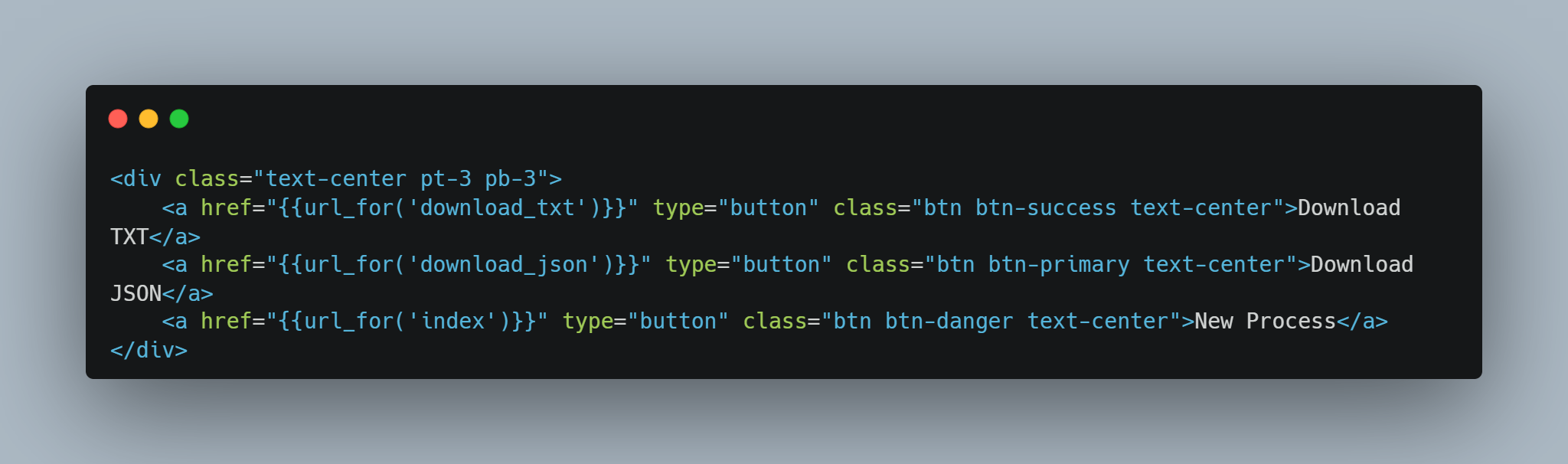


上傳功能介面

* result.html關鍵部分



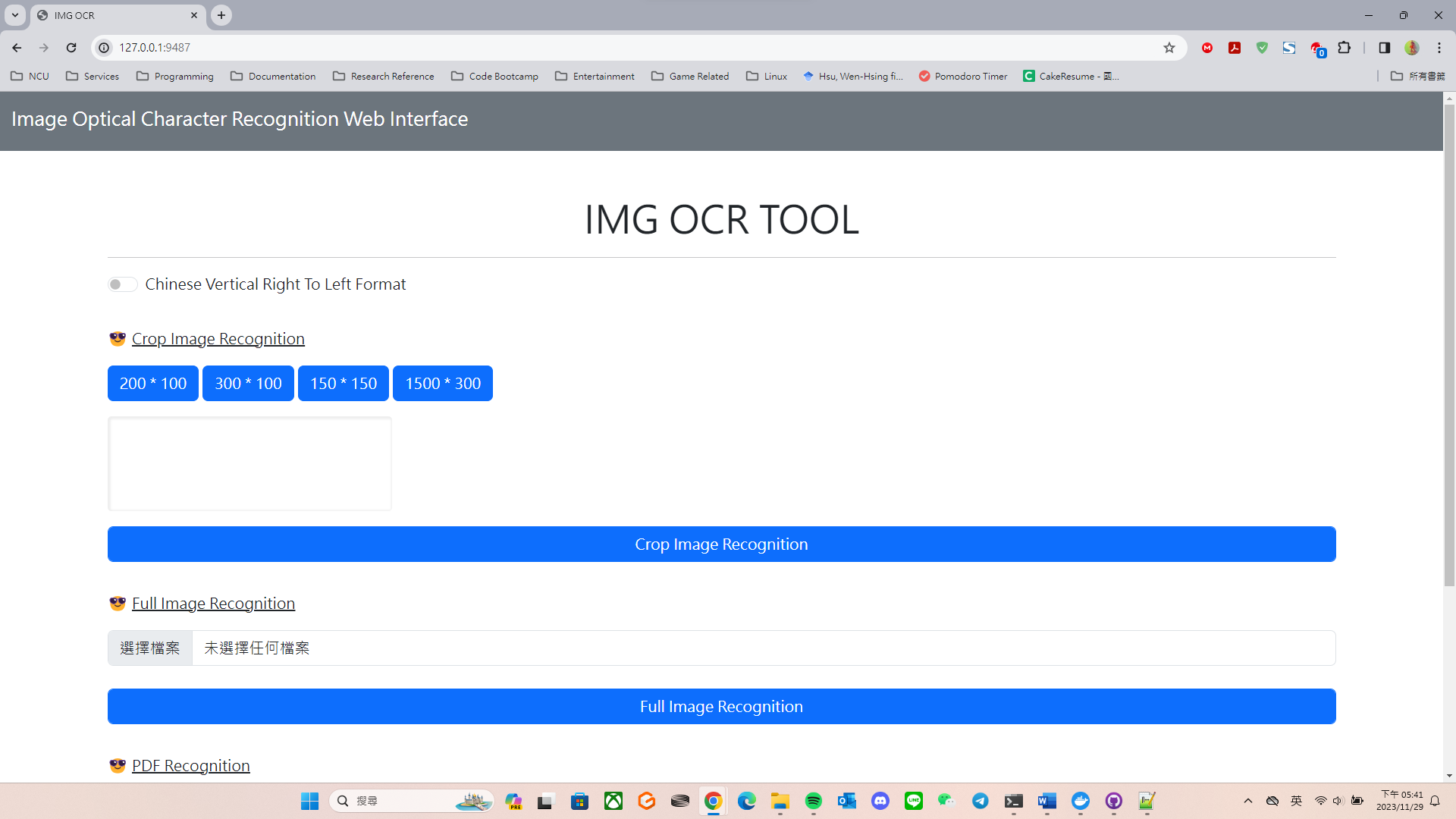
辨識結果視覺化燈箱

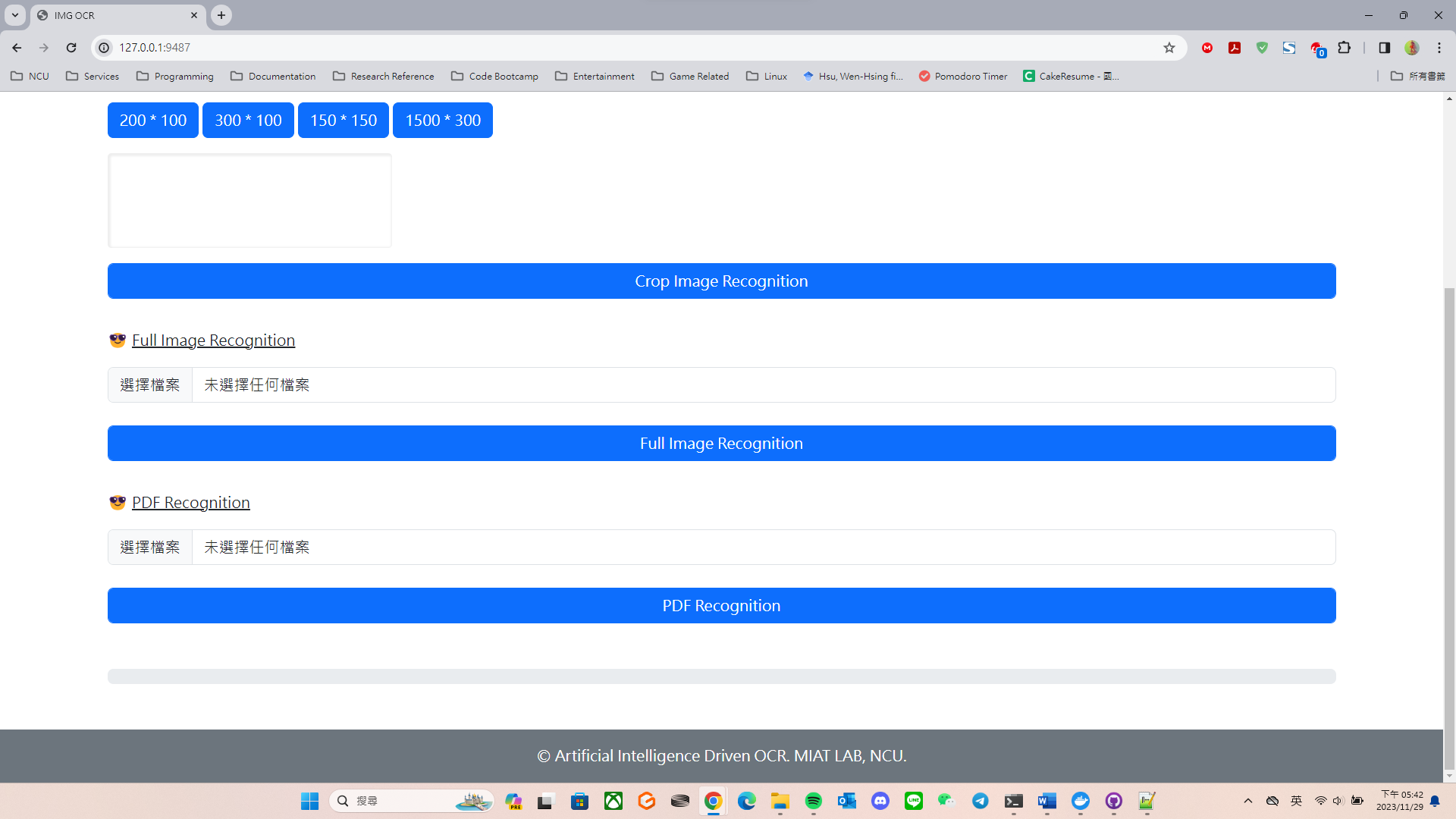


各項功能跳轉

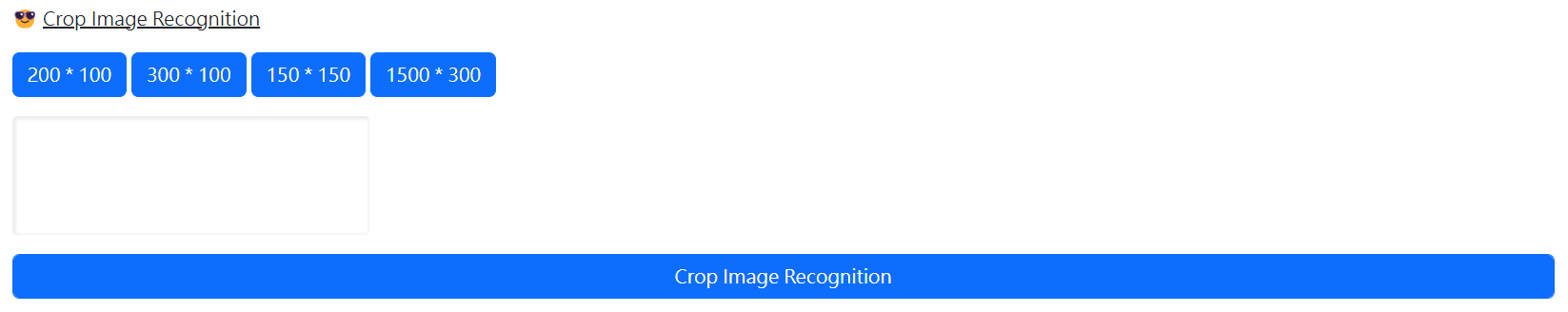
**網頁功能入口**

* 功能入口，支持JPG、JPEG、PNG圖片及PDF文檔

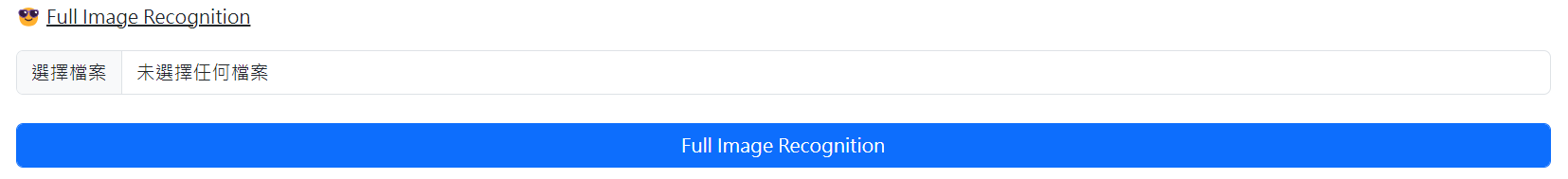




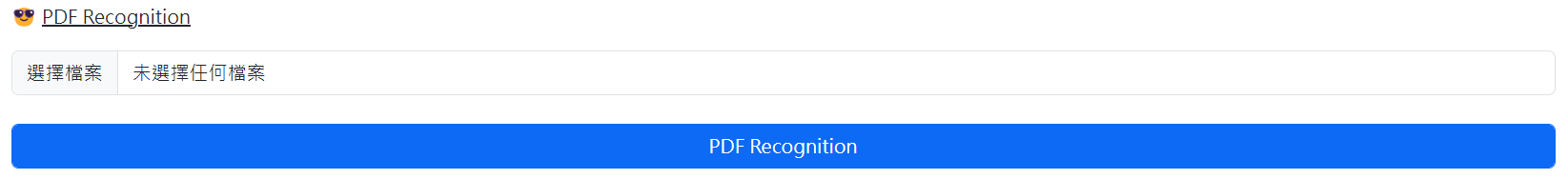
* 第一選擇框：裁切圖片辨識



* 第二選擇框：完整圖片辨識



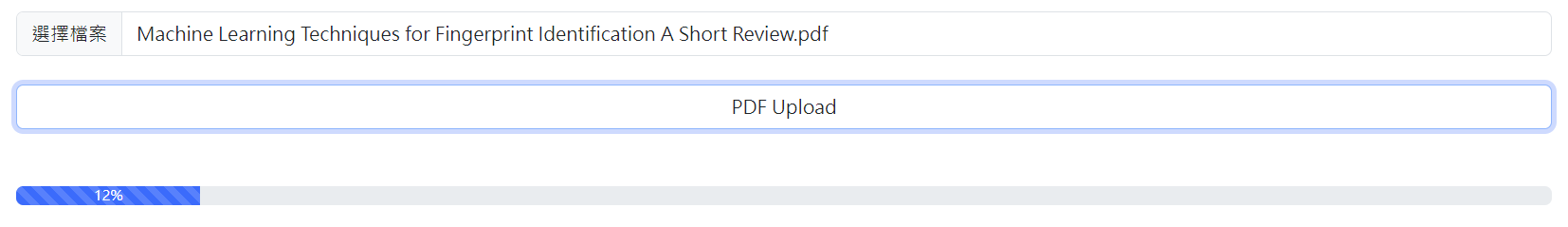
* 第三選擇框：PDF文檔辨識



* 圖像特殊排序：中文直式由右至左（預設標準為由上至下）

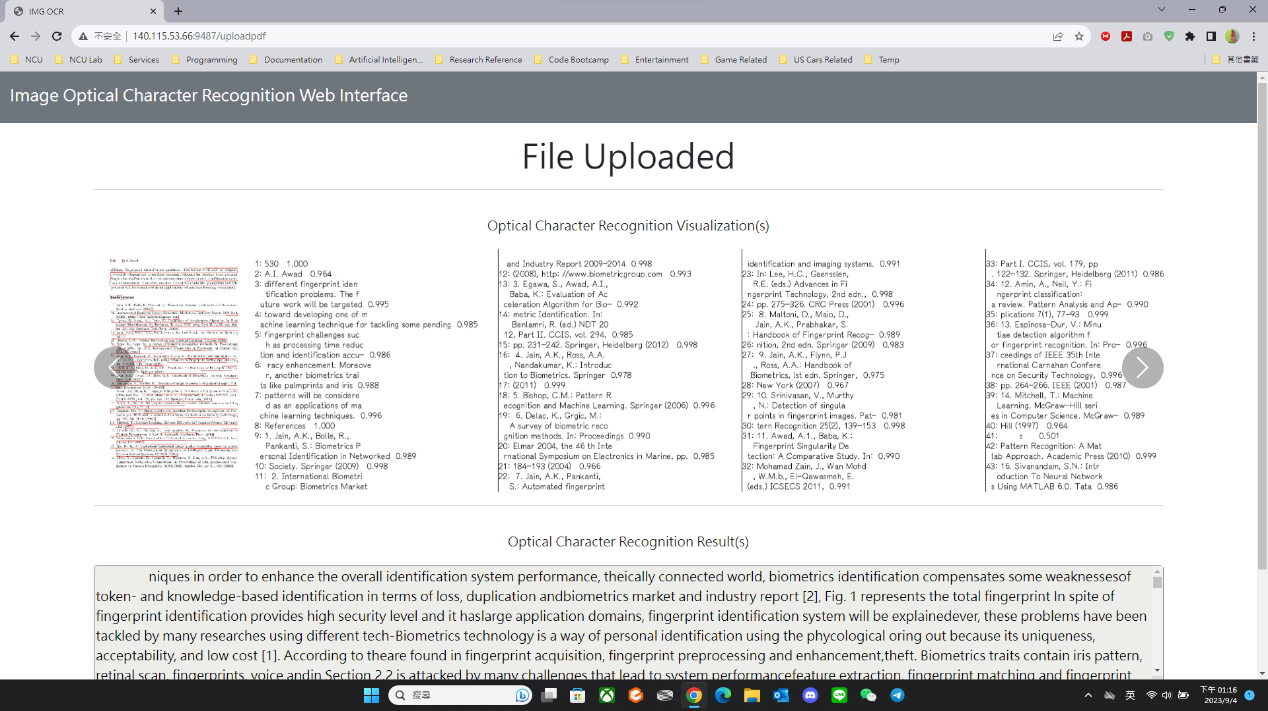


* 成功讀取資料後，可於下方看見處理進度條

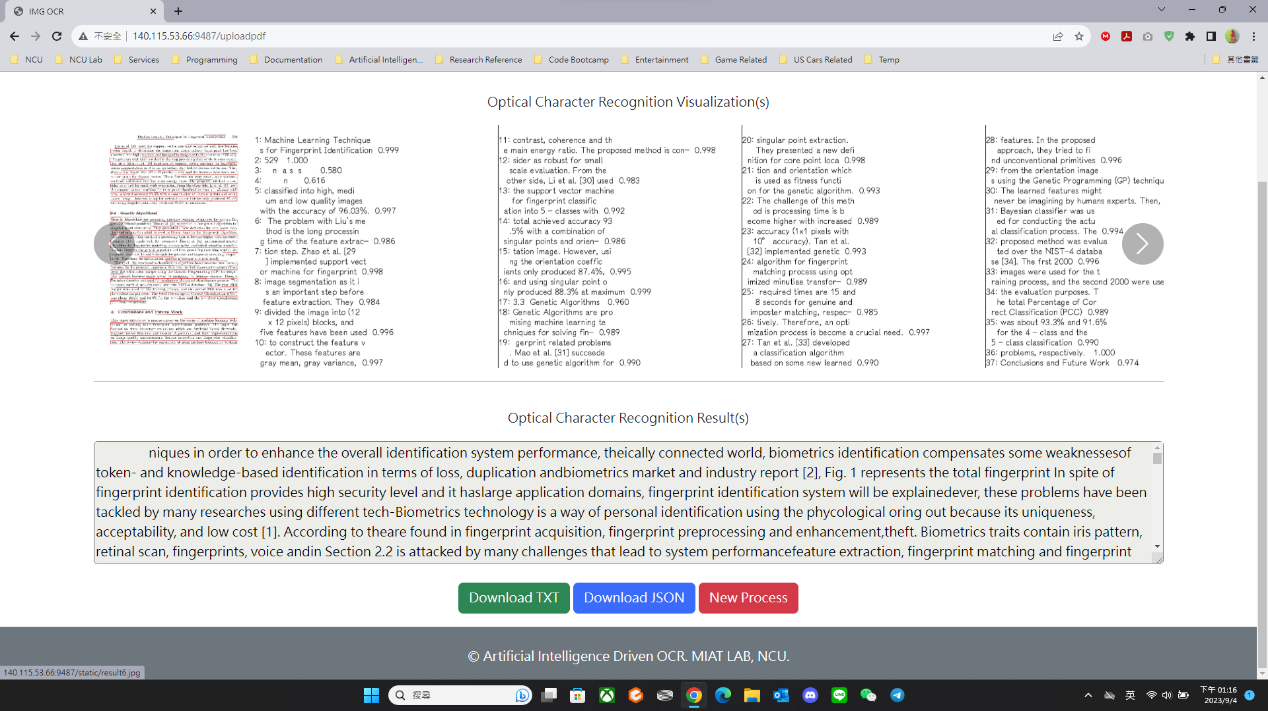


**網頁結果輸出**

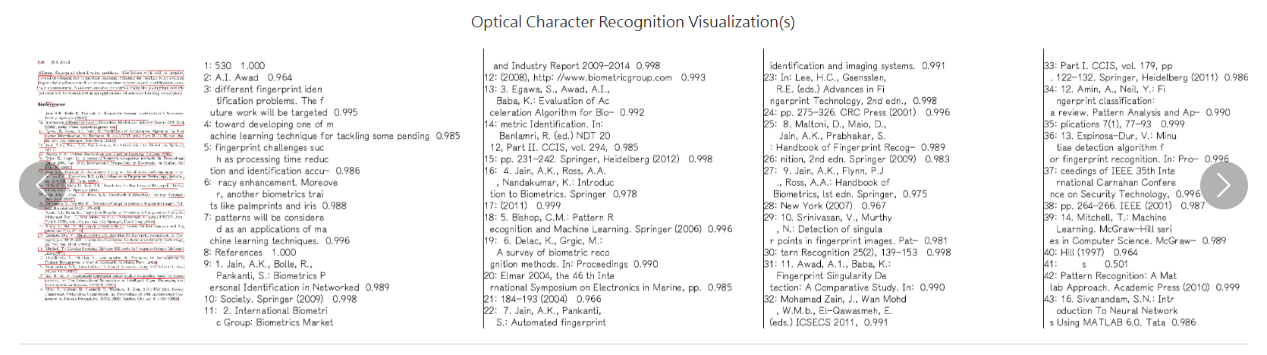
* 結果輸出頁面：上半部



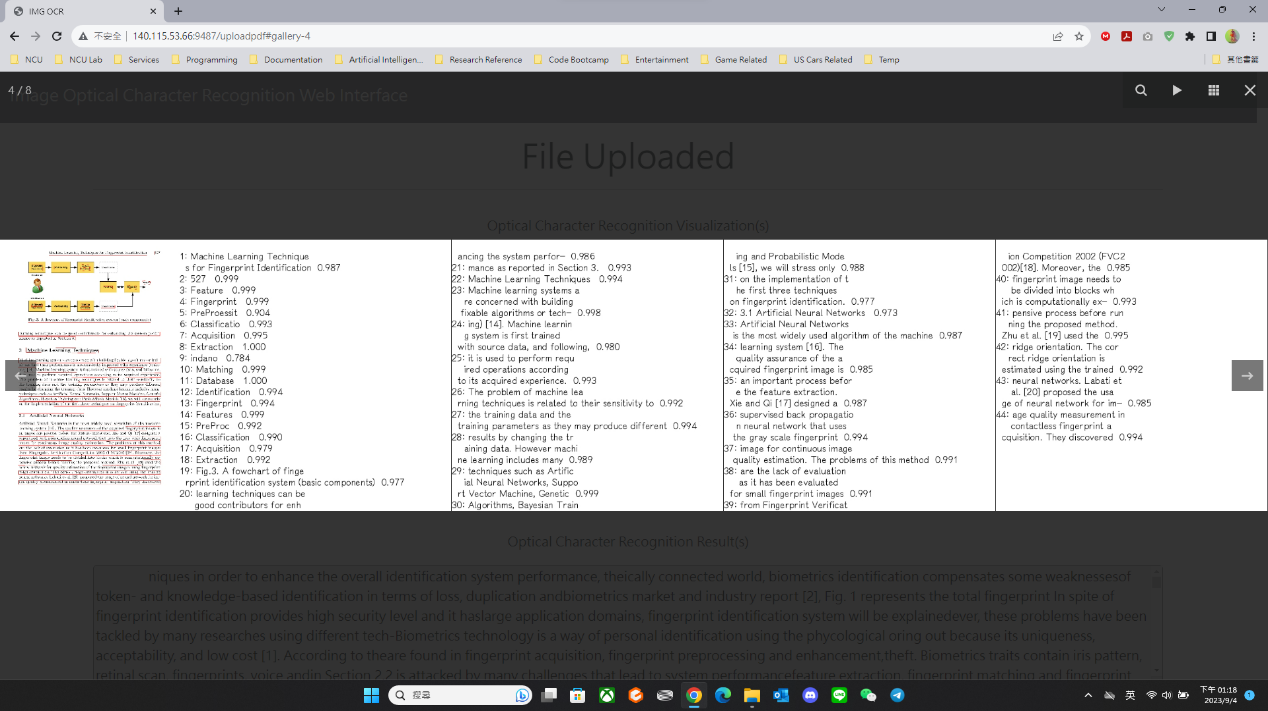
* 結果輸出頁面：下半部

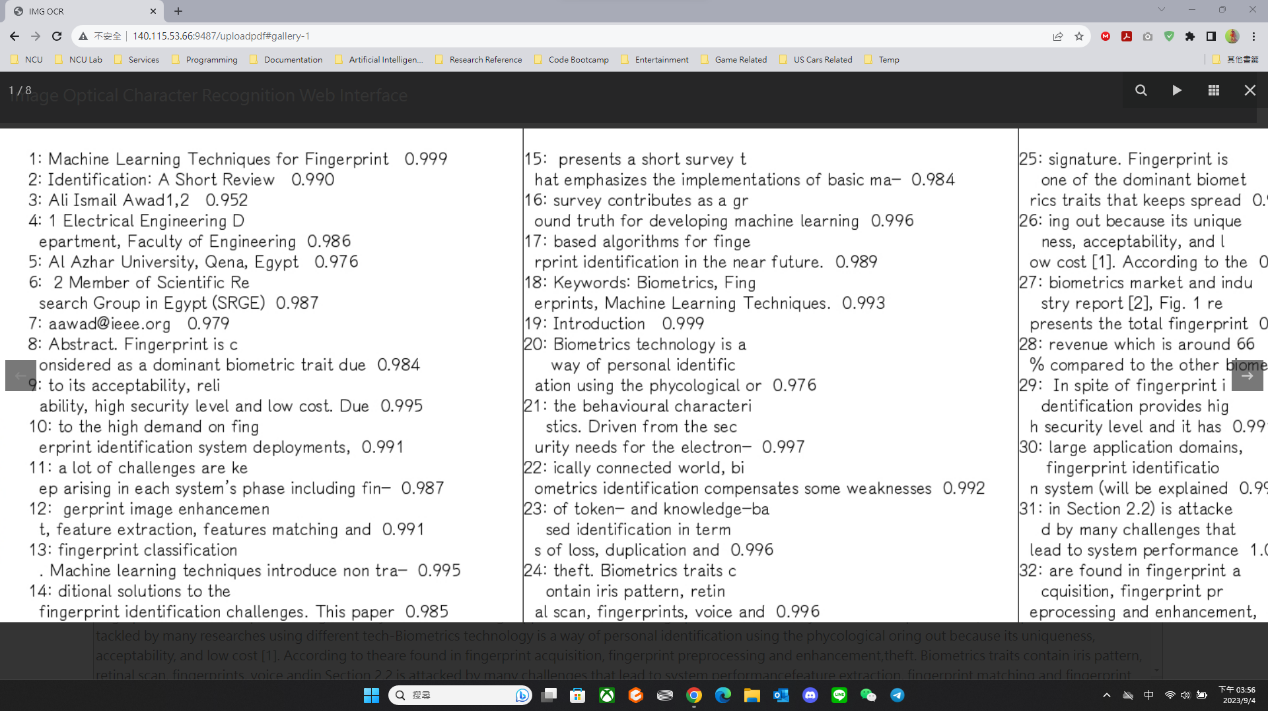


* 輸出視覺化：包含辨識框及輸出結果展示，左右按鍵可變換當前展示頁面

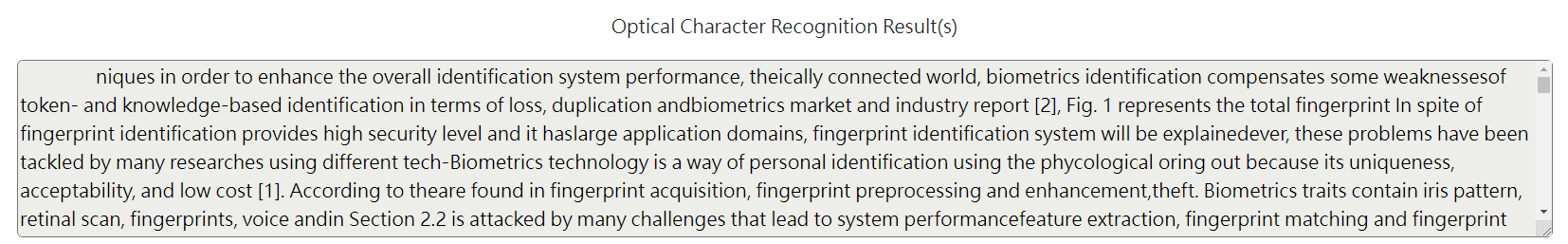


* 直接點擊圖片進入相簿模式，可以對圖片進行放大





* 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

