Swordle——美國手語字母辨識遊戲

林政夆, 王筠婷, 張子捷, 鄭嘉齡, 蘇慶欣, 林妤蓁, 何信佑†
國立中興大學人工智慧程式設計與應用開發人才培訓班第 01 期
† 通訊作者，電子郵件：jb22621550@gmail.com

摘要—在台灣的 12.5 萬名聽障人士中，僅有 2 至 3 萬人使用手語，反應出手語學習成本高、效率低的問題。由於台灣手語資源有限，我們首先選擇技術成熟的美國手語（American Sign Language, ASL）作為切入點，開發 ASL 字母辨識工具，進行技術驗證與市場測試，為未來台灣手語系統的開發奠定基礎。我們利用 YOLOv8 與 Django 框架開發了一套手語辨識系統，專注於透過英文單字手語進行字母辨識，並透過遊戲化的學習方式提升聽障人士的學習興趣，解決手語翻譯不足及學習動力低落的問題。在成功開發手語辨識遊戲 Swordle 後，我們將進一步增強影像處理，優化辨識率，解決遞延與偵測物距離遠近對準確率的影響，並逐步擴展至中文手語辨識，以有效解決台灣手語學習資源不足的問題。

關鍵詞—YOLOv8, 美國手語, Django

I. 研究動機

根據世界聾人聯盟(World Federation of the Deaf, WFD）調查，全球有超過 7 千萬名聽障者，使用 300 種以上不同的手語 [1]，依照區域、文化的不同，衍生發展各自的手語系統，甚至也有同一地區內部所產生的變體，除此之外，聽障者群體也發展一套可跨國交流的國際手語，顯現全球手語系統的多樣性。

在全球手語系統多樣性的背景下，台灣手語（Taiwan Sign Language, TSL）展現了其獨特的歷史與文化演變。台灣手語的發展始於日治時期，受日本手語影響逐步形成，並在台灣北、中、南地區形成了不同的方言。隨著社會對手語和聾人文化的重視，台灣手語現已成為台灣的國家語言之一。然而，儘管台灣擁有約 12.5 萬名聽障人士，實際使用台灣手語的人數卻僅有 2 至 3 萬人 [2]。這一現象的原因在於學習成本過高以及使用效率低下。

II. 模型應用與技術參考

手語辨識目前是一個活躍的研究領域，已經有許多專案和開源資源可以參考。在 Kaggle、Hugging Face[1] 和 GitHub 等平台上，有許多研究者和開發者分享了他們的手語辨識模型和程式碼。此外，Google 也曾在 Kaggle 舉辦 ASL 拼字序列資料辨識的競賽[2]，吸引社群投入研究。這些資源涵蓋了各種手語語言的資料集和模型，為其他開發者提供了寶貴的參考和基礎。

在本研究中，我們主要參考了現有的手語辨識模型和相關技術，並基於這些技術進行了應用與改進。由於時間與資源限制，我們未進行廣泛的文獻回顧，而是著重於實踐中高效且被證明有效的技術方案。這些參考模型的選擇是基於它們在相關研究中的成功應用，並結合我們的需求進行了具體的調整與優化，以確保在有限的時間和資源內達到預期的辨識效果。

A. Sign Language on Hugging Face

這個專案使用名為 HuggingPics 的圖像收集工具 [3], [4] 並發佈在 Hugging Face。HuggingPics 使用 Google 的 Vision Transformer (ViT) ImageNet-21k 作為模型，並利用 PyTorch Lightning 進行微調。根據該專案自我報告，該模型的準確率達到 100%。專案還提供了 ASL 字母的範例圖像，以展示辨識效果。

B. ASL Fingerspelling Recognition w/ TensorFlow

此 Kaggle 專案由 Luiz Gustavo Martins 等人提出，取得 2023 年 Google 美國手語拼字辨識競賽的金牌獎。該專案展示如何使用 TensorFlow 框架和自行搭建的 ASLFR Transformer 處理經過 MediaPipe[3] 擷取的雙手關鍵點（keypoint）序列資料 [5]。其所使用的資料集非常全面，包含 67,208 筆序列資料。而且專案結構完整，包含資料預處理、模型訓練和評估，提供了 TensorFlow 在 ASL 辨識任務中的有效應用實例。

C. Sign Language Alphabets Detection and Recognition using YOLOv8

Muhammad Moin 的這個 GitHub 專案著重於使用 YOLOv8 模型來檢測和辨識手語字母 [6]。YOLOv8 以其在物件偵測任務中的速度和精確度而著稱，在此應用於識別和分類對應於字母的手勢。專案提供了詳細的說明文件，涵蓋了模型的設置、使用自定義資料集進行訓練，以及部署模型進行即時辨識的過程。使用 YOLOv8 在需要快速且精準檢測的場景中具有優勢，使其適合於即時應用如手語翻譯。

[1]https://huggingface.co/
[2]https://www.kaggle.com/competitions/asl-fingerspelling
[3]https://github.com/google-ai-edge/mediapipe

D. Action Detection for Sign Language

Nicholas Renotte 的 GitHub 專案使用 MediaPipe 擷取手部、臉部和上半身關鍵點座標，再自行錄製三種手語影像序列，以便訓練長短期記憶模型（LSTM）[7]。該專案不僅能辨識靜態手語，還能檢測構成手語交流基礎的動作。專案包括了設置環境、訓練模型和部署模型進行即時檢測的 YouTube 教學影片。

III. 模型技術說明

經過需求評估之後，我們採用 Muhammad Moin 的 *Sign Language Alphabets Detection and Recognition using YOLOv8* 作為藍本發展深度學習模型，詳細的說明如下。

A. 模型訓練環境 *Kaggle*

Kaggle[4] 提供了一個高度整合的資料科學平台，擁有資料集、Notebook 環境以及計算加速裝置，如 GPU 和 TPU，讓資料科學家可以輕鬆進行模型訓練和測試。本專案選擇在 Kaggle 平台上訓練模型，利用其雲端環境和豐富的 GPU 資源來管理大型資料集和模型訓練過程。為了加速訓練並提高計算效率，我們使用了兩個 NVIDIA Tesla 4 GPU 進行並行計算，這種雙 GPU 配置能夠有效處理 YOLO 模型在訓練過程中所需的大量計算和記憶體資源，特別適合於物件偵測任務。

B. *Roboflow* 平台與資料集

Roboflow[5] 提供了強大的電腦視覺工具，支持資料集管理、標註、增強和模型部署，使用者可以有效率地進行圖像標註和增強。因此，本專案採用 Roboflow 平台來準備資料集，再透過 Roboflow API 下載資料集至 Kaggle 平台，使用自定義的配置檔案來指定訓練和驗證資料的位置，以及模型應學習的類別標籤，為模型提供訓練的基礎。

本專案採用的資料係參考 David Lee[8] 在 Roboflow 所提供的資料集，主要圖像共 720 張，其中訓練集為 504 張（70%），驗證集 144 張（20%），以及測試集 72 張（10%）。除此資料集之外，又於該平台蒐集 ASL 字母圖像至 16,993 張，並將擴增之資料集再度投入訓練，以取得最佳成果。

C. *YOLO* 模型特色

YOLO (You Only Look Once) 是一種基於深度學習的高效物件偵測（object detection）技術，其核心特色在於只需一次前向傳播即可識別圖像中的所有物體及其位置。這種設計使得 YOLO 特別適合需要低延遲的應用場景，如自動駕駛和監控系統。

YOLO 的發展始於 2015 年，由 Joseph Redmon 首次提出 [9]。初代 YOLOv1 引入了即時物件偵測的概念，雖然速度快，但在小物件偵測上仍有局限。隨後的 YOLOv2 和 YOLOv3 逐步改善了小物件偵測能力和模型的準確性，並引入了多尺度預測架構，平衡了速度與準確性。2020 年，YOLOv4 由台灣學者王健堯及其團隊與俄羅斯學者 Alexey 共同開發，通過新增骨幹網絡和特徵提取技術，顯著提升了模型性能 [10]。

YOLOv8[6] 是 Ultralytics 公司在 2023 年推出的版本，在技術上進行了多項改進，特別針對邊緣裝置進行了優化。其主要特色包括無錨檢測技術（Anchor Free Detection），減少了模型的計算負擔，提高了推理速度。此外，YOLOv8 引入了新的卷積層設計，進一步增強了特徵提取和信息融合能力。這些改進使 YOLOv8 能夠在更多樣化的環境中運行，尤其是在邊緣裝置上表現出色，適用於影像分類、物件偵測、實例分割等多種 AI 視覺任務 [11]。

D. 訓練的參數設定

在 YOLO 模型的訓練中，選擇了 YOLOv8 large（`yolov8l.pt`）作為預訓練模型，這一版本在準確性和資源需求之間取得了平衡。訓練任務設為「物件偵測」，目的是識別影像中的拼字邊界框（bounding box）並進行類別標註。訓練過程共進行了 150 個迭代週期（epoch），以便模型能充分學習手語的特徵，並透過多次迭代來優化其參數。

批次大小（batch size）設為 32，這有助於在訓練中維持穩定的梯度估計，同時避免超過 GPU 的記憶體限制，從而在訓練速度和資源使用之間取得平衡。影像尺寸則設定為 800 像素，旨在保留足夠的圖像細節，同時控制計算負擔，以提高模型在偵測不同大小手語字母時的效能。

E. 物件偵測模型評估

在我們訓練的最終結果中，模型的損失函數（loss functions）包括邊界框回歸損失（bounding box loss, `box_loss`）、分類損失（classification loss, `cls_loss`）和分佈焦點損失（distribution focal loss, `dfl_loss`），這些損失分別為 0.27564、0.17242 和 0.93065。這些數值代表的是模型在預測物件邊界框、類別以及邊界框精確度方面的誤差，損失值越低，表示模型預測越接近真實值。關於損失函數隨著迭代週期的變換，可參見圖 1。

性能指標（evaluation metrics）方面，我們的模型在邊界框精確率（Precision, P）上達到了 0.95166，意味著邊界框中物件中有大約 95% 是真實存在的；召回率（Recall, R）為 0.84991，表示真實存在的手勢中有 85% 被正確檢測到。此外，平均精確度 `mAP50` 為 0.94697，`mAP50-95` 為

[4]https://www.kaggle.com/.

[5]https://roboflow.com/.

[6]https://github.com/ultralytics/ultralytics

0.803，這些數值反映了模型在不同 IoU（Intersection over Union）門檻下的檢測精確度。總體來說，這些數據表明我們的模型具有甚高的準確性和穩定性。關於性能指標隨著迭代週期的變換，可參見圖 2。

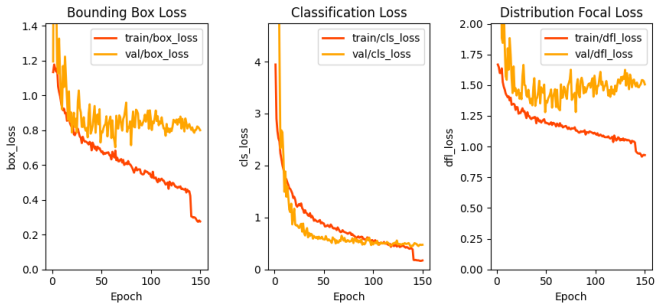

圖 1. 損失函數 vs. 迭代週期

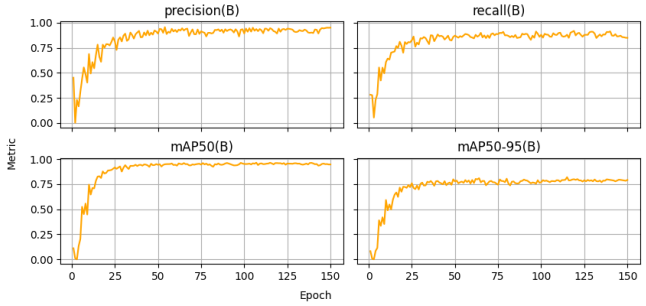

圖 2. 性能指標 vs. 迭代週期

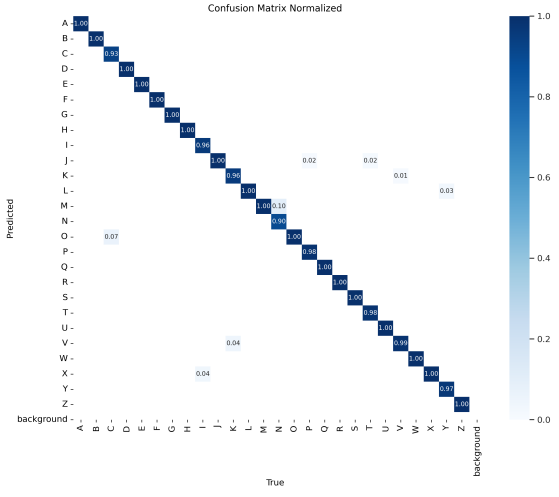

圖 3. 標準化混淆矩陣

IV. Web 框架與部署

本專案使用 Django 作為前後端框架，開發了具備 Wordle 遊戲[7]、字母手語教學及影像辨識等功能的互動網頁應用，並實現會員系統及排名功能。應用程序在 GCP 上部署，使用 Apache 伺服器、`mod_wsgi` 模組和 Let's Encrypt 的 HTTPS 憑證，確保應用的安全性和功能性，提供加密通訊以保障使用者隱私。

A. Django

本專案以 Django 作為前後端框架的基礎，開發了具有多項功能的網頁應用系統，涵蓋 Wordle 遊戲、字母手語教學、會員系統和遊戲排名功能，並結合影像辨識技術，提供了互動式的學習和遊戲體驗。後端部分利用 Django 的 ORM 進行資料庫操作和業務邏輯處理，並通過 Django REST Framework 實現了 RESTful API，支持 JSON 格式的數據傳輸，使前端能夠靈活地進行資料交換。以上即為本專案基於 Django 框架的架構說明。

1) 上傳圖片、辨識 *API:* 在這個手語辨識的過程中，前端客戶端透過網頁開啟攝影機來擷取手語的手型，並將圖像上傳到伺服器。伺服器接收到圖像後，會先進行色彩和翻轉的資料處理，然後將處理過的圖像發送至模型進行辨識。如果上傳成功，模型會返回辨識結果給伺服器，伺服器再將結果回傳給前端客戶端，透過 Wordle 模式或教學模式與使用者互動。如果上傳失敗，伺服器則會將錯誤訊息返回給客戶端。

2) 註冊登入 *API:* 當前端客戶端使用 POST 方法發送註冊請求時，伺服器首先查詢使用者資料。如果使用者已存在，伺服器會返回錯誤訊息。如果使用者不存在，伺服器會創建新帳號，生成新的 token，並記錄登入時間。token 用於身份驗證和會話管理，確保每次請求都能確認使用者身份並維持會話狀態。伺服器會將註冊成功的訊息、token 和登入時間返回給客戶端。

對於登入請求，伺服器會查詢使用者資料。如果使用者存在，伺服器會檢查密碼是否正確。若密碼正確，伺服器會更新登入時間，生成新的 token，並將登入成功的訊息、token 和登入時間返回給客戶端。客戶端會將登入狀態儲存到 `sessionStorage`。若密碼錯誤或使用者不存在，伺服器會返回相應的錯誤訊息。

3) 提交遊戲成績 *API:* 當客戶端發送遊戲結果時，首先從網址列獲取 `userid`，並從網頁的`sessionStorage` 取得 token。然後，客戶端使用 POST 方法將`userid`、遊戲分數和 token 發送至伺服器。伺服器接收到請求後，查詢使用者資料。如果使用者存在，伺服器會檢查 token 是否正確。如果 token 正確，伺服器會更新該使用者的總分

[7]Wordle 為 Josh Wardle 編寫的英文文字遊戲，六次機會內猜中某個有五字英文字母的詞彙。遊戲規則詳見後文敘述，或參考https://www.nytimes.com/games/wordle/index.html。

數，並返回成功訊息給客戶端。如果 token 錯誤，伺服器會返回錯誤訊息。如果使用者不存在，伺服器會返回使用者不存在的錯誤訊息。

4) 排名 *API:* 對於用戶排名查詢，客戶端使用 GET 方法發送包含 `userid` 的請求至伺服器。伺服器會查詢該使用者的資料。如果使用者存在，伺服器會進一步查詢所有使用者的總分數，按分數排序，並返回包括排名、分數、使用者數量以及前三名使用者的資料。如果使用者不存在，伺服器會返回無此使用者的錯誤訊息。

B. GCP 部署

我們在 Google Cloud Platform（GCP）上使用 Google Compute Engine（GCE）的虛擬機器來部署我們的 Web App，並且用 Apache HTTP Server 配置虛擬主機。為了確保 Web 應用程式的安全性和功能性，尤其是為了讓瀏覽器可以安全存取視訊攝影機，我們使用 Let's Encrypt[8] 申請了 SSL 憑證，並配置了 Apache 虛擬主機以支援 HTTPS 連接。而且所有的 HTTP 請求均被配置為永久重定向至 HTTPS，以提供加密通訊，保障使用者的隱私。關於 GCE 虛擬機器的規格與伺服器的配置，請參見表 I。

表 I
虛擬機器與伺服器規格表

項目	規格描述
虛擬機器名稱	swordle-vm0
虛擬機器類型	e2-small
CPU 平台	Intel Broadwell
架構	x86/64
處理器	1 vCPU
記憶體	2 GB
儲存空間	30 GB 已平衡永久磁碟
作業系統	Ubuntu 22.04 LTS (Jammy Jellyfish)
IP 位址	靜態 IPv4 (`35.221.146.182`)
HTTPS 支援	已啟用（使用 Let's Encrypt 申請 SSL 憑證）
Apache 模組	安裝 `mod_wsgi`，支援 Django 框架
防火牆設定	啟用 HTTP 和 HTTPS 流量，允許外部存取

V. 使用者體驗

本專案的網頁介面設計基於 HTML、CSS 和 JavaScript 三項核心技術，全面提升系統功能和使用者體驗。專案中包含兩種模式：教學模式和 Wordle 模式，以促進手語學習和娛樂的有效結合。

教學模式旨在協助使用者掌握英文字母的手語表達。使用者可選擇字母，系統將展示該字母的手語圖像及說明，並即時提供手語識別結果及準確度評估，以助使用者調整並改進其手語技巧，為 Wordle 模式奠定堅實的學習基礎。

[8]https://letsencrypt.org/zh-tw/

Wordle 模式融合手語識別與單字猜測遊戲，創造挑戰性和互動性的學習環境。玩家須根據系統反饋逐步推測單字，系統提供手語識別結果及準確度，以便玩家檢查手語表達。提示顏色包括綠色（字母正確且位置正確）、黃色（字母正確但位置錯誤）和灰色（字母不在單字中）。註冊並登入玩家的遊戲成績將根據猜測所需的嘗試次數進行計算，並顯示於排行榜上。Wordle 模式旨在通過手語識別挑戰提升英文字母手語掌握，增強學習的互動性和趣味性。

VI. 展望

目前的專案專注於 ASL 的應用，並已成功將模型辨識技術與遊戲開發結合。在這一階段，我們掌握了手語辨識的基礎技術。透過在 Kaggle 平台上使用 YOLOv8 訓練出的 ASL 字母辨識模型，本團隊將其整合至 Django REST Framework 的 API 中，建立了一個美 ASL 拼字遊戲網站。這一成果為未來發展 TSL 辨識系統應用提供了初步的基礎。

展望未來，基於 TSL 的獨特性，我們有幾個值得探索的發展方向。例如，逐步建立 TSL 語料庫，並結合深度學習技術，開發出能夠辨識和翻譯 TSL 詞彙的系統。我們期望本專案的延續能為未來台灣聽障人士提供一個更具實用性和無障礙的溝通平台。

參考文獻

[1] National Geographic. "Sign language." (2024), [Online]. Available: https://education.nationalgeographic.org/resource/sign-language/ (visited on 08/31/2024).

[2] Yahoo 新聞. "12 萬聽障者有知的權力: 手語扮橋梁." (Apr. 2020), [Online]. Available: https://tw.news.yahoo.com/12%E8%90%AC%E8%81%BD%E9%9A%9C%E8%80%85%E6%9C%89%E7%9F%A5%E7%9A%84%E6%AC%8A%E5%8A%9B-%E6%89%8B%E8%AA%9E%E6%89%AE%E6%A9%8B%E6%A2%81-201008663.html (visited on 09/04/2024).

[3] O. Akyıldız. "Sign language." (2023), [Online]. Available: https://huggingface.co/RavenOnur/Sign-Language (visited on 08/31/2024).

[4] N. Raw. "Huggingpics." (2021), [Online]. Available: https://github.com/nateraw/huggingpics (visited on 08/31/2024).

[5] L. G. Martins. "Asl fingerspelling recognition with tensorflow." (2023), [Online]. Available: https://www.kaggle.com/code/gusthema/asl-fingerspelling-recognition-w-tensorflow (visited on 08/31/2024).

[6] M. Moin. "Sign language alphabets detection and recognition using yolov8." (2023), [Online]. Available: https : / / github . com / MuhammadMoinFaisal/Sign-Language-Alphabets-Detection-and-Recongition-using-YOLOv8 (visited on 08/31/2024).

[7] N. Renotte. "Action detection for sign language." (2021), [Online]. Available: https : / / github . com / nicknochnack / ActionDetectionforSignLanguage (visited on 08/31/2024).

[8] D. Lee. "American sign language letters." (2022), [Online]. Available: https://github.com/nateraw/huggingpics (visited on 09/03/2024).

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1506.02640.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2004.10934.

[11] X. Zhai, Z. Huang, T. Li, H. Liu, and S. Wang, "Yolo-drone: An optimized yolov8 network for tiny uav object detection," *Electronics*, vol. 12, no. 17, p. 3664, Aug. 2023. [Online]. Available: https : / / www.mdpi.com/2079-9292/12/17/3664.