專題摘要

一、 關鍵詞

classification; cnn; geoguessr; deep learning; 地景辨識

二、 專題研究動機與目的

起初我們發現了 Geoguessr[1] 這個遊戲。在 Geoguessr 中，電腦會將玩家放置於半隨機的 Google 街景地點，玩家要根據自己在地圖中移動位置和旋轉方位所得到的線索來猜測他們在世界上的位置。在遊玩的過程中，我們發現如果單純以一般玩家對世界各地景物的認知與了解去做辨識是十分困難的，統計下來的正確率甚至不及 10%。因此，我們希望可以藉由深度類神經網路在影像辨識上的應用，訓練出一個可以精準歸類圖像所屬國家的模型，以可以高正確率的自行遊玩 GeoGuessr 這個遊戲。

三、 現有相關研究概況及比較

在過往的研究中，學者提出之辨識方法的目的多半是在判斷景物類別<1>，而非判斷景物所屬國家。而在眾多研究中最接近直接判斷景物所屬國家的方法，也僅是先將照片中的景色作分類<2>，再根據含有所屬類別的地區中，挑選出最有可能的選項作為預測之結果。而我們發現這樣的方法，不僅沒辦法精確地指出景物是屬於哪一個國家，更因為是用景物類別下去做訓練的關係，這樣的方法沒有辦法適應一個國家內有多種景物的情況。

四、 專題重要貢獻

基於過往的研究和現有的模型，在經過我們的修改及調校過後，我們訓練出一個得以辨識照片中景物所屬國家的模型。除此之外，在遊玩 GeoGuessr 方面，玩家們可以搭配我們的模型遊玩遊戲，藉由模型的輔助，甚至在完全依賴模型的情況下，玩家判斷國家的準確率可以提升到 70%以上，有十分顯著的提升。這不只大幅提升了玩家的遊玩體驗，更可以加速玩家的學習。

五、 設計原理、研究方法與步驟

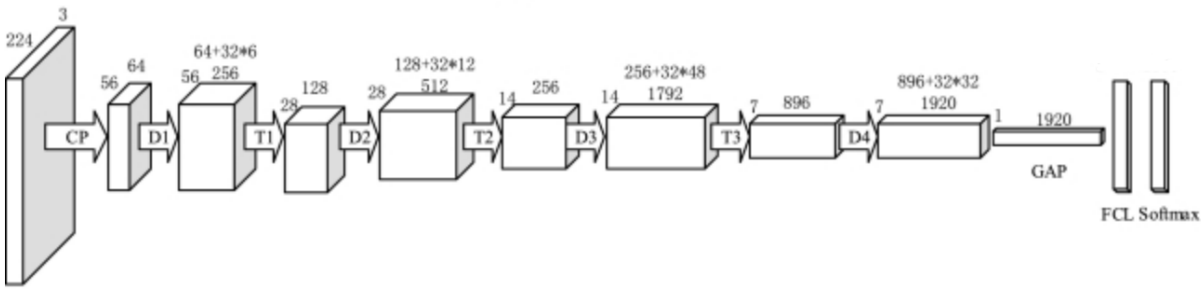
在最初資料收集的部分，我們使用的是 Google Cloud Platform 中的 Google Street View Static API[2] 來取得世界各地之街景圖。除此之外，由於我們要訓練的模型之主要目標是要精準判斷照片中景物的所屬國家，資料來源所屬地理位置的正確性就也變得格外重要。因此，在蒐集資料的同時，我們搭配使用 GeoNames[3]，一個涵蓋所有國家和地區的地理資料庫，以及 Google Cloud Platform 中的 Google Geocoding API[4] 來確認資料來源地點是否與其所屬標籤相符，以確保資料之可靠性。

最初在設計辨認景物所屬國家的模型時，我們提出了兩種做法：

- 分兩階段去做辨識，先判別圖片中景物是屬於哪一洲後，再分辨景物是屬於洲內哪　個國家，以降低單一模型的所需區分的國家數量
- 直接使用單一模型去判別圖中之景物屬於哪一個國家

在相互比較過後，我們選擇了第二種方法作為我們最終的辨識方法，我們將於下一個章節說明物這樣選擇的原因及兩種方法在判斷上之差別。

在模型挑選的部分，我們直接使用 Pytorch 預先建立好的模型去進行比較和調整，並且在最後加上一層 Fully Connected Layer 使最後的 Output 數量與我們的 Classes 數量為一致。除了嘗試各種不同的模型之外，我們也比較了各種不同的資料前處理方式，並嘗試進行各項參數之調整，最後選擇以 densenet201 作為我們的模型，由下圖一可見我們所使用之 Densenet 201 的架構圖，詳細比較結果將會在下一個章節詳述。



圖一、densenet201 架構

在訓練出一個有最佳表現的模型過後，我們便將此模型套用到我們設計的 GeoGuessr 輔助遊玩工具中，玩家可以自行擷取街景讓我們的模型進行預測，而我們的模型就會依據玩家擷取的數張圖片即時產生一組預測之結果，輔助玩家進行的判斷。最後，在我們得以達成基本的輔助功能後，我們也對我們輔助工具之介面進行優化，以增強使用者體驗。

六、 系統實現與實驗[5]

A. 資料蒐集

我們最初在訂定目標時，是希望可以訓練出一個得以自行遊玩 GeoGuessr 的電腦特務。因此，在挑選要進行辨識的國家時，我們是以在遊戲中出現頻率較高的國家[5] 來做挑選，讓我們在預測的結果上得以更加準確。而在利用 Google Street API 收集資料的同時，我們不只會利用先前提到的 GeoNames 服務及 Google Geocoding API 來確認資料來源位置是否正確，我們同時會以人工的方式挑選出不符合要求、或是會擾亂我們的模型之圖片。會需要做這樣的動作，主要是因為 Google Street API 的圖片來源除了街景之外，同時包含了建築物室內的照片、昏暗的照片、太聚焦在某一物體上的照片，如樹木、牆壁、車子等，以及使用者自行上傳的照片，而這些照片，如圖二中之範例圖片，都有讓模型失焦的可能性。

圖二、可能會誤導模型之圖片

B. 單一階段與兩階段式之分類方法

最初在設計辨認景物所屬國家的模型時，我們提出了兩種做法：

- 分兩階段去做辨識，先判別圖片中景物是屬於哪一洲後，再分辨景物是屬於洲內哪　個國家，以降低單一模型的所需區分的國家數量
- 直接使用單一模型去判別圖中之景物屬於哪一個國家

在評估及嘗試過後，我們最終選擇了第二種方法，主要是考量到單一模型的正確率本身有限，當兩個正確率有限的模型搭配起來，整體正確率將會低於直接以單一階段去分辨國家的方式。

以 33 個歐洲國家為例，進行單一階段與兩階段式的測試，結果發現雖然兩個階段的模型表現因為較低的 Classes 的數量而有較高的準確率，但並沒有非常顯著的提升，因此在兩個準確率相乘之後，甚至還不如直接訓練單一模型的結果出色，如表一中我們針對歐洲國

家用兩種方法去做辨識後之結果，這也是我們最後仍選擇使用單一模型來進行國家辨識的原因。

實驗名稱	將歐洲劃分為四個區域	將歐洲劃分為六個區域	整個歐洲(單一模型)
區分區域之模型	73%	68%	--
區分國家之模型	<56%	56%	--
整體 Accuracy	<40%	38%	47%

表一、單一階段與兩階段式模型比較

另外由於 Geoguessr 本身的遊戲機制，一個回合可以有三次作答機會，因此在以下的實驗中我們也會特別關注 top-3 accuracy 的結果，以更符合遊戲遊玩時的情境。

C. Training Data 數量對模型準確度的影響

在整個實驗的過程中，我們嘗試了各種不一樣的方法來增加模型的正確率，其中我們發現最顯著的莫過於增加資料量，在我們的實驗過程中，當我們每增加一次資料量，正確率就會對應提升，如表二中不同資料量對應之準確率。而最終我們總共蒐集了約 57,000 張圖片作為我們的 Dataset，詳細資料分布請見表三。

South America (7 Classes)			
Training data per class	200	300	400
Training Accuracy	74%	84%	84%
Validation Accuracy	62%	63%	65%

Europe (34 Classes)				
Training data per class	200	300	400	500
Training Accuracy	60%	60%	62%	61%
Validation Accuracy	40%	42%	44%	47%

World (66 Classes)				
Training data per class	400	500	600	700+
Training Accuracy	82%	83%	80%	83%
Validation Accuracy	55%	56%	58%	61%
Top-3 Accuracy	76%	76%	78%	80%

表二、訓練資料量

Training Data	Validation Data
48877 images 752 images/per country	8597 images 132 images/per country
85.04%	14.96%

表三、最終 Dataset 的數量（Training Data 佔比 85.04%， Validation Data 佔比 14.96%）

D. 不同模型及實驗方法之相互比較

我們也嘗試了各種不同的 Pytorch 內建模型，在嘗試的眾多模型中，我們發現相較於其他模型，如表四中紀錄之不同模型在辨識上的數據，在相同的設定下，VGG 和 DenseNet 有較好的表現。此外，我們也發現由越多層 Layer 及越多參數組成的模型之效果也相對越好，在相互比較過後，在後期我們調整參數時，都是以 DenseNet-201 為基礎去做訓練。

South America (7 Classes)				
Model	resnet18	alexnet	vgg11_bn	densenet12
Training Accuracy	72%	66%	82%	44%
Validation Accuracy	59%	57%	64%	50%

World (66 Classes)						
Model	resnet18	alexnet	vgg11_bn	densenet12	resnet101	densenet201
Training Accuracy	56%	46%	54%	65%	78%	82%
Validation Accuracy	47%	39%	51%	52%	52%	55%
Top-3 Accuracy	69%	61%	73%	74%	74%	76%

表四、不同 Pytorch 模型於辨識景物所屬國家上之表現比較

除了做參數上的調整，我們也有針對圖片做了一些處理，除了將圖片做基本的 Resize 之外，還會將圖片做 Random Crop，但這樣的作法讓我們懷疑，是否會將我們圖片中重要的部分給裁切掉，如房屋、路樹等等。因此我們便嘗試用 Random Crop 以及 Center Crop 來作替代，我們發現這樣的替換會造成 Training Accuracy 暴增到近乎 100%的狀況，但 Validation Loss 卻不減反增，有嚴重的 overfitting 狀況出現。

我們猜測會造成這樣結果的原因主要與三者的運作方式有關。由圖三可見，相較於

Random Resize Crop，Center Crop 和 Random Crop 只會以一個相同大小的區域左右移動做裁切。相較於 Random Resize Crop 在裁切大小和位置上會有較多的變化，Center Crop 和 Random Crop 較容易讓電腦誤以為發現規律的表徵且記住一些資訊細節，進而導致 Training Accuracy 可以到近乎 100% 的狀況，但由於我們的資料量並不是很巨大，因此在整體上的表現會不夠 general，使 validation 時的表現十分糟糕。

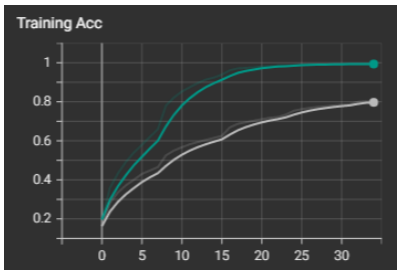
圖四則呈現在 Random Crop 的情況下，模型的 training accuracy 快速收斂至近乎 100%的狀況，如圖四之一，但其 validation 結果卻較 Random Resized Crop 差，無論是 accuracy 或是 loss 都是如此，如圖四之二及圖四之三。

Center Crop

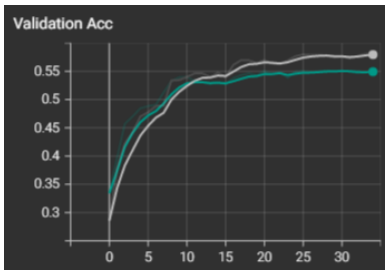
Random Crop

Random Resize Crop

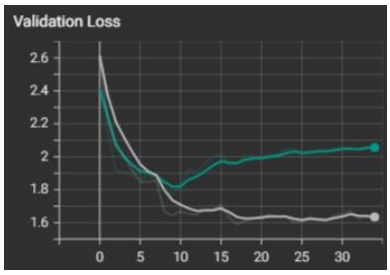
圖三、三種不同的圖片裁切方式之差異



圖四之一



圖四之二



圖四之三

圖四之一 Random Crop (綠) 與 Random Resized Crop (白) 之 training accuracy

圖四之二 Random Crop (綠) 與 Random Resized Crop (白) 之 validation accuracy

圖四之三 Random Crop (綠) 與 Random Resized Crop (白) 之 validation loss

此外，我們也研究了將不同的資料前處理方式應用在我們模型上的可能性，其中我們嘗試將圖片轉為灰階以及 Gaussian Blur 兩種不同的資料前處理方式。嘗試將圖片轉灰階去做訓練，主要是我們想要驗證圖片的顏色對於我們這個模型的訓練上，是不是真的有幫助，或是對模型是不是有干擾的作用。但在套上灰階過後，模型的表現明顯變差，由此可見，圖片中景物的色彩表現在模型的學習中仍扮演著十分重要的角色。

再者，在驗證色彩表現在模型的學習中舉足輕重過後，我們從眾多資料前處理方式中選擇 Gaussian Blur 來做嘗試，主要是希望藉由對圖片進行模糊處理來試圖解決 training 過程的 overfitting 問題，因此我們便嘗試利用 Gaussian Blur 將圖片做模糊處理。結果顯示，經過 Gaussian Blur 處理過後，模型的整體表現同樣有下降的趨勢，因此，我們進而推測模糊化圖片將會減少過多資訊量並導致判別效果下降，且對於解決 overfitting 沒有顯著幫助。

最後我們也有針對其他 hyperparameter 進行測試，例如 learning rate、batch size、learning rate decay 等等的參數進行調整，最終模型的設定請見表五。

Batch size	Learning rate	Step / Gamma	Pre-trained
16	0.001	8 / 0.5	Yes

表五、hyperparameters 設定

E. Confusion Matrix [6]

在將我們的模型應用到遊戲中來做驗證的同時，我們也會將各個國家出現的次數、模型判斷的結果、我們的遊玩狀況等重要數據做好完整的記錄。我們將遊玩的數據整理過後，製作成了一個 66 x 66 的混淆矩陣。透過這個矩陣，我們發現正確率高於 80%的國家往往都具有十分鮮明的特色，如：以高原地形聞名的賴索托（準確率 91%）、被稱為海港之都的新加坡（準確率 86%）等等。

另外，我們也發現正確率較低的國家，如法國（準確率 28%）、義大利（準確率 32%）都有一個共同特色，就是其國家內部的地形風景都有較劇烈的變化，使得模型較難捕捉到單一關鍵的特徵。另外在矩陣中我們也發現特別容易被電腦誤判為彼此的國家，如表六中所述，比利時和荷蘭、加拿大和美國、英國和愛爾蘭等等，彼此都是互相接壤，因此也會有較近的地理風貌。其中比較特殊的是愛沙尼雅和立陶宛，他們雖然並不相鄰，但因為同屬波羅的海三小國，他們的地形、植物種類仍十分相近，相信這是他們容易被誤判的原因。

Ground truth	Prediction	占 Ground truth 比例
Belgium	Netherlands	10%
Canada	United States	10%
Estonia	Lithuania	10%
France	Belgium	10%
Montenegro	Albania	14%
Russia	Ukraine	11%
Slovakia	Austria	11%
Ukraine	Russia	11%
United Kingdom	Ireland	11%

表六、容易被模型混淆之國家及其占比

七、 效能評估與成果[7]

最後我們對模型在真實遊戲中的表現進行了測試，並比較了在訓練模型時以及真實在遊玩 GeoGuessr 間的差異。我們發現今天在訓練模型時的 validation accuracy 約為 61%，但當我們今天把模型套用到我們的輔助工具中，讓玩家搭配著我們的模型去遊玩時，相互配合下可以達到超過七成的正確率。在分析過後，我們認為實際遊玩時有下列幾點差異：

第一，我們的模型在實際遊玩遊戲的時候不是以一張圖片定生死，它可以藉由玩家調整不同視角以及利用移動至不同的地方，得到不同位置及視角的街景圖，並且根據不同視角過後做出判斷，因此在做出一個國家的預測時，是經過多張玩家的攫取圖片所判定的，因此能大大提升答題正確率。

第二，城市的比例，我們所使用的資料收集方法是採用隨機的方式，如表七中所述，收集到大城市資料的機率，相較於大城市在遊戲中出現的機率（13%）更低。這樣會造成我們的模型缺乏辨識大城市的能力，進而導致在辨識大城市時，容易混淆判斷成那些原本就有相當比例都市圖片的國家的狀況，如新加坡、德國等。

	大城市	郊區或鄉村	整體測試
占總回合比例	13%	87%	100%
Accuracy	42%	75%	70%
Top-3 Accuracy	79%	88%	81%

表七、城市和郊區在整體資料中之占比及其正確率

第三，由於我們的模型並未囊括所有國家，因此在真實遊玩這個遊戲時，仍然會有低機率碰到一些不包含在模型 classes 中的國家(測試時有 6%出現非目標國家)，如表八中所述，進而降低整體的正確率。若將這些例外排除，準確率與訓練時的 validation accuracy 提升了不少。

	訓練中 validation 的狀況	排除非目標國家	整體測試
Accuracy	61%	74%	70%
Top-3 Accuracy	80%	87%	81%

表八、城市和郊區在整體資料中之占比及其正確率

八、 結論

在這個的專題研究中，我們驗證了運用深度類神經網路在影像辨識上的應用方法去做街景所屬國家辨識的可能性，並且讓我們的模型在辨識 66 個國家的街景時，得以達到 61%的準確度。如果要進一步提升模型的準確度，我們認為首先可以多搜集一些來自每個國家大城市的資料，因為 Geoguessr 中隨機跳出的考題有一定比例會在大都市之中，因此，我們認為透過大幅度增加各個國家中城市的資料量，勢必可以提升模型在遊玩時的準確度。

另外，在玩家靠自己的認知去玩 Geoguessr 時，相較於我們訓練的模型，人類可以看到一些更細微的東西，如文字、車輛是靠左或靠右行駛、太陽的方位、車型、國旗等等。未來若是我們要進一步提升模型的準確度，我們認為可以與一些特殊的模型，如文字語言辨識模型、國旗辨識模型、太陽方位辨識模型等做搭配，相信在這樣的搭配項，結果一定會更為準確、也更為亮眼，相信這個模型也將應用於更多領域，發揮更大、更廣的作用！

九、 參考文獻

<1> Recognizing Cities from Street View Images
http://cs231n.stanford.edu/reports/2016/pdfs/422_Report.pdf

<2> IM2GPS: estimating geographic information from a single image
http://graphics.cs.cmu.edu/projects/im2gps/im2gps.pdf

十、 附錄

[1] Geoguessr: https://www.geoguessr.com

[2] Google Street View Static API: https://developers.google.com/maps/documentation/streetview/overview

[3] GeoNames: http://www.geonames.org

[4] Google Geocoding API: https://developers.google.com/maps/documentation/geocoding/overview

[5] 系統實現與實驗

- Github Repository:
  https://github.com/jonathan-liu-0204/CS-Graduate-Project--GeoGuessr-Agent
- Comparison of Models and Experiments:
  https://docs.google.com/spreadsheets/d/1xkLweQziOTVoZh3IRLCucdoJTCRWPkm7/edit?usp=sharing&ouid=100497560184725586643&rtpof=true&sd=true

[6] Confusion Matrix: https://i.imgur.com/GedQKoc.png

[7] Testing Results:
https://docs.google.com/spreadsheets/d/1BSw9XqoiUzrIwHlmMrB24LggP-HaTZbH/edit?usp=sharing&ouid=100497560184725586643&rtpof=true&sd=true