**主題：應用深度學習技術作黑白漫畫上色之研究**

指導教授：余執彰

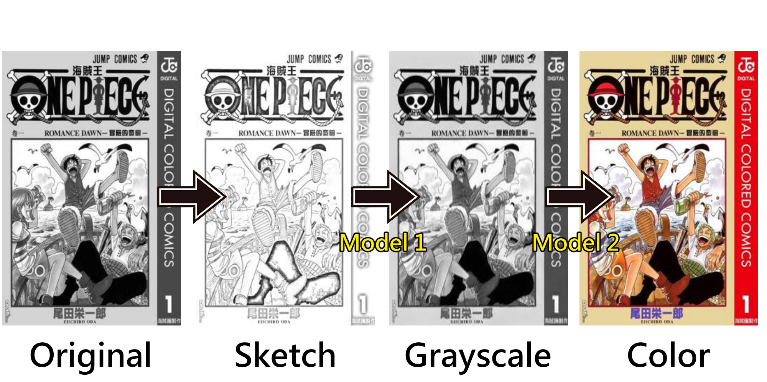
1. **摘要**

　　本專題將運用深度學習結合影像處理技術，基於GAN架構，歸納漫畫中的特徵，依此調整訓練參數，並在Lab色彩空間中，輸入黑白漫畫的L通道，透過模型預測上色結果，再使用圖像處理技巧優化輸出圖片，同時探討不同方法流程對上色結果造成的影響，尋求最佳解。

1. **研究動機**

　　至今已經有許多AI圖像上色工具可供使用，但依舊無法完全仰賴於機器上色，而這也是本專題所聚焦改善的方向，目標能夠對傳統的日式黑白漫畫進行全自動上色（Automatic Colorization）。其中本研究要處理的問題大致如下：

1. 訓練用的資料格式不一，如何對資料進行預處理。
2. 探討要如何運用有限的數據量來進行深度學習。
3. 自動上色方式的神經網路（Neural Net-work）之架構設計。
4. 如何對日式漫畫的特徵做個別處理，包含網點、對話框、擬聲詞、漸層，這些技法的上色方式較為特殊，需要額外做處理。
5. **研究方法與結果**

****

圖（一）：上色流程示意圖

　　本專題的上色流程設計如圖（一）所示，首先輸入原始漫畫，將其轉換成線稿後，接著輸入給線稿轉灰階的上色模型，將得到的灰階漫畫再輸入給灰階轉彩色的上色模型，最後會得到彩色漫畫。為了簡潔說明，以下會將線稿轉灰階的上色模型、灰階轉彩色的上色模型分別稱作Model1、Model2。

**第一部分：資料集處理**

　　本專題使用「海賊王漫畫彩色版本」作為彩色漫畫資料集（共計12477張圖片），進一步對原圖片套用灰階轉換，取得灰階漫畫資料集，再藉由高斯模糊找到圖片的輪廓，其運算結果作為線稿漫畫資料集。因為是採用監督式學習，我們在製作資料集時，針對不同資料夾下的圖片使用相同的命名，確保圖片之間正確對應。

**Step 1-1：**

　　由於原圖片尺寸較大，不利於模型訓練，因此我們將圖片縮放成256\*256像素，提高模型的收斂速度。

**Step 1-2：**

　　將圖像從RGB模式轉換成LAB模式， LAB模式下，L代表明亮度，取值範圍為0到100，A用來表示從綠色到紅色的範圍 B則代表藍色到黃色的範圍，A和B的取值範圍皆介於-128到127之間。如果採用LAB模式，對於Model1來說，圖片以L通道數值作為輸入，會得到輸出L'數值並成為Model2的輸入，最後會輸出預測的AB通道數值。因此我們可以藉由輸入圖片提供的資訊，預測上色，這麼做可減少複雜程度且加快模型收斂速度。

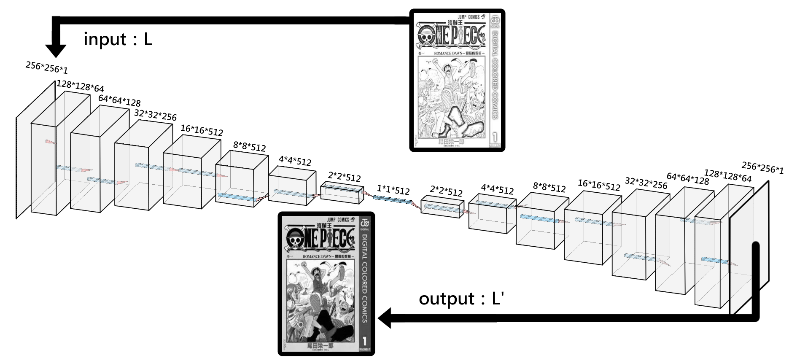
**Step 1-３：**

　　為了讓模型更有效學習漫畫物件特徵，在模型訓練過程中會對圖片進行shuffle，可以防止模型抖動，避免發生過擬合，並且使得模型學到更加正確的特徵。

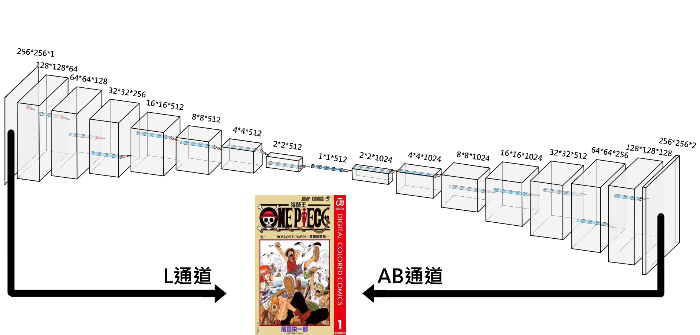
**第二部分：模型架構說明**

　　本專題中的Model1與Model2都是採用GAN（Generative Adversarial Network）來建立模型架構，包含生成器（Generator）、判別器（Discriminator）。

**生成器：**



圖（二）Model1生成器架構圖

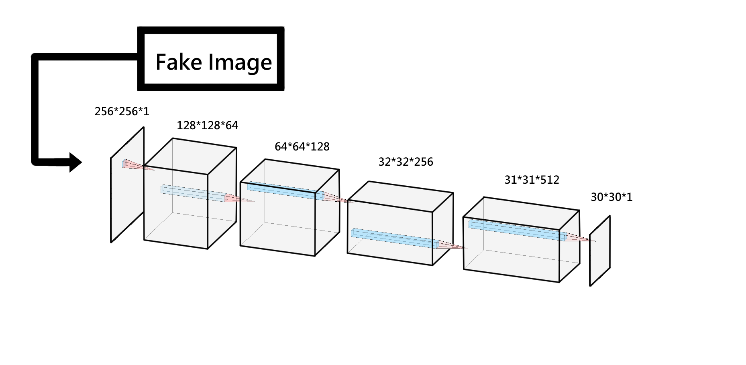


圖（三）Model2生成器架構圖

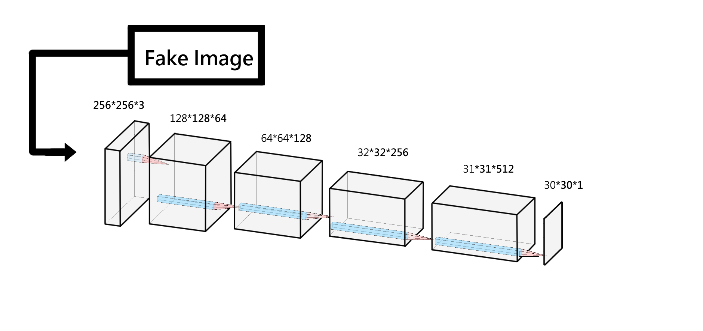
　　Model1和Model2的生成器如圖（二）、圖（三）所示，參考U-net架構設計而成。對Model1生成器的輸入一張256\*256的單通道（L）線稿圖片，經由卷積層（Convol-ution Layer）後，會獲得64張128\*128的特徵圖（Activation Maps），依此類推。獲取多張特徵圖，也就意味著擁有更豐富的資訊，模型對於學習的掌握度會提升，到達U-net最底層時，共有512張1\*1的特徵圖，由於U-net是對稱結構，因此還會再進行多次反卷積運算，輸出一張256\*256的單通道（L'）灰階圖片；Model2生成器也是依照相同的模式搭建出來的，不同之處在於Model2的輸入是Model1的輸出，而Model2的輸出則是雙通道（AB）彩色圖片，最後再將L'通道與AB通道合併，會得到上色成果圖。

　　生成器架構中，每一次卷積運算所採用的卷積核（Kernal Size）為4\*4、滑動步長（Strides）為2，運算前會運用補0（Zero Padding）的手法，一定程度上彌補邊界訊息缺失的問題，運算後則選用Leaky ReLU作為激勵函數（Activating function），可以節省運算量，並且避免梯度消失的問題。

**判別器：**



圖（四）Model1判別器架構圖



圖（五）Model2判別器架構圖

　　Model1和Model2的判別器如圖（四）、圖（五）所示，我們使用PatchGAN來鑑定圖像的生成品質，其輸出為一張N\*N的特徵圖，基於感受野（Receptive Field）在圖像上的預測結果來計算損失（Loss），能夠考慮到圖像不同部分的影響，讓模型能夠更關注圖像細節，對上色也較有幫助。



圖（六）Model2訓練流程圖

　　接著要來探討生成器與判別器之間是如何互動的，以下將以Model2的生成器和判別器來舉例說明。生成器會訓練出一個神經網路，用來產生彩色圖，而判別器會判定生成彩色圖與原始彩色圖的差別，分辨圖片是生成彩色圖或是原始彩色圖，並且為該圖片評分，如果是生成彩色圖，得到的分數會較低，若是原始彩色圖則得到較高的分數，生成器和判別器之間會不斷地進化，也就是說，判別器識別圖片的標準會不斷提高，因此生成器也必須增強，產生出更接近原始彩色圖的圖片，目標通過判別器的識別，示意如圖（六）。

　　要成功騙過判別器的話，生成器必須產出以假亂真的彩色圖片，我們使用以下步驟定義生成器的損失：

1. L1Loss = ，是目標值（原始彩圖的特徵圖）, 是估計值（生成彩圖的特徵圖）。
2. 對圖片套用Sigmoid函數，將數值正規化到0到1之間，BCELoss = ，是目標值（1矩陣）, 是估計值（生成彩圖的特徵圖）。
3. 最後得到損失函數Loss\_G = L1Loss + BCELoss。

　　另外，為了避免生成器去生成出一些偏離目標，但卻可以欺騙判別器的圖，我們使用以下步驟定義判別器的損失：

1. 對圖片套用Sigmoid函數，將數值正規化到0到1之間，BCELoss\_fake = ，是目標值（0矩陣）, 是估計值（生成彩圖的特徵圖），BCELoss\_real = ，是目標值（1矩陣）, 是估計值（原始彩圖的特徵圖）。
2. 最後得到損失函數Loss\_D = BCELoss\_fake + BCELoss\_real。

　　這些損失可以成為模型評估指標，以了解生成出的圖片跟真實圖片在結構（線條、顏色）上的差異，再來決定判別器的更新方向。實務上，我們將生成器和判別器的參數隨機初始化，每一次迴圈要完成兩個步驟：



圖（七）Model2流程說明圖

**步驟一**：

　　將生成器的參數固定，調整判別器的參數，參考圖（七）。將灰階圖片傳給生成器後會產生一張彩色圖，此時判別器從資料集中取出原始彩色圖並與生成彩色圖做比較，通常原始彩色圖會得到較高的分數，而生成彩色圖分數較低，接著判別器會更新參數，變得更加嚴格，當生成器生成彩色圖可以拿到接近原始彩色圖的高分，就代表該預測模型訓練成功。



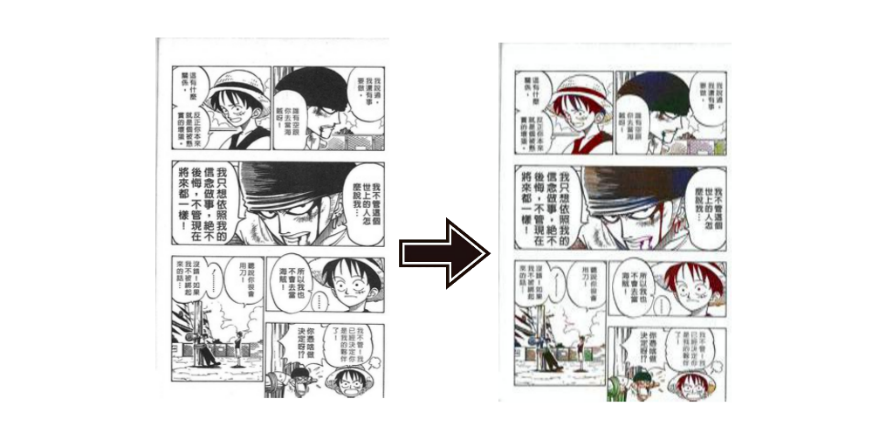
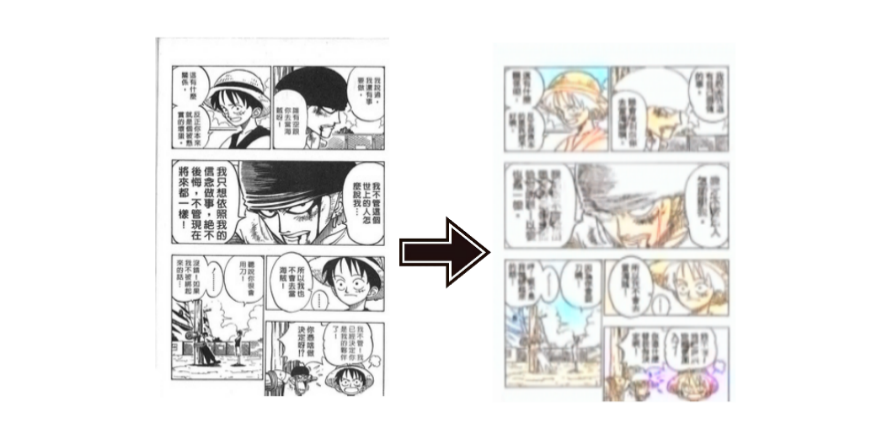
圖（八）Model2流程說明圖

**步驟二**：

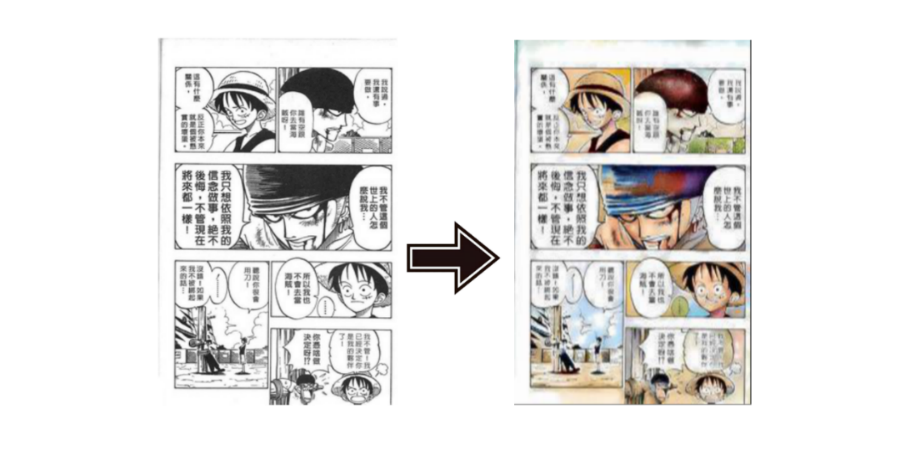
　　將判別器的參數固定，調整生成器的參數，參考圖（八），判別器會根據生成彩色圖的好壞給予評分，我們希望得到的分數越高越好

**第三部分：分析與優化**

　　至此，您可能會很好奇為何要使用二階段上色呢？事實上，我們已經嘗試過在同樣模型架構下設計單階段上色的方法，但在實驗過程中發現其在實際漫畫的表現上不甚理想。舉例來說，灰階圖片轉彩色圖片的上色模型無法對漫畫中的空白區域上色，因為空白區域的L通道數值為0，而且在訓練集中，灰階圖片的空白區域都是固定不被上色的，但是此模型對非空白區域的上色結果是傑出的，如圖（九）所示；線稿圖片轉彩色圖片的上色模型能夠有效學習到漫畫物件的資訊，給予合適的顏色，但仍有些美中不足之處，線稿圖片較缺乏L通道的資訊，導致上色彩度較低，其中黑色的L=0, A=0, B=0，白色的L=100, A=0, B=0，由於線稿圖片的漫畫物件只保留線條外框，大面積的塗色會變為白色，而黑色與白色的AB通道是相同的，因此可能會導致頭髮、衣服、眼睛等黑色區塊被上成白色，如圖（十）所示；而使用二階段上色模型可以獲得圖（十一）的結果。

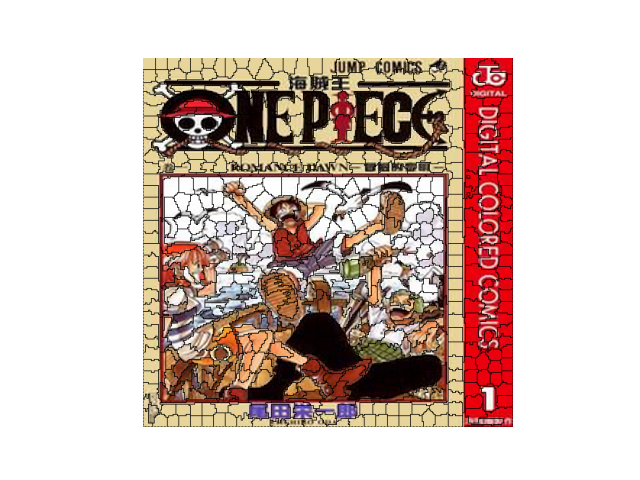
圖（九）單階段模型-灰階圖片轉彩色圖片 圖（十）單階段模型-線稿圖片轉彩色圖片



圖（十一）二階段模型上色結果

　　為了使模型上色更豐富多樣，我們基於現有的模型架構，嘗試將Model1和Model2的生成器架構改以Resnet34建構成的U-net取代，並分別訓練1500個Epoch（5000張圖片跑1000個Epoch+10000張圖片跑500個Epoch）及700個Epoch（10000張圖片跑700個Epoch），而後引入結構相似性指標（structural similarity index measure, SSIM），將原始彩色圖片與生成彩色圖片作為輸入，輸出SSIM數值（介於0到1之間），從而判斷其相似程度。我們從資料集中隨機取樣100張灰階圖片進行上色後，再與對應的原圖比較，平均得到的SSIM數值高達0.82，較原架構高出0.7，足以證明新的生成器架構具備更好的學習力。

　　在實驗過程中，察覺到某些生成彩色圖片的大面積色塊較不平均，視覺觀感上較為突兀，因此導入了SLIC（Simple Linear Iterative Cluster）演算法，用來生成超像素（Super-pixel）圖片來做處理。SLIC演算法和K-means具類演算法十分類似，會先生成K個種子點，搜尋每個與種子最近的像素，與該種子點劃分成一類，並對群體取平均值計算新的種子點，將舊的種子點移到新的種子點，反覆上述動作直到收斂，得到的像素分割如圖（十二）所示，接著會計算同一類別的所有像素的色彩平均值，最後用色彩平均值取代該類別的所有像素點，如圖（十三）所示。我們將生成彩色圖片轉換成超像素圖片，並讓超像素圖片的部分像素取代生成彩色圖片，使區域色塊較為平均。

圖（十二）超像素分割示意圖，K=1000 圖（十三）色塊平均示意圖，K=1000

1. **結論**

　　我們成功透過二階段上色模型解決特定漫畫區域不被上色的問題，搭配Resnet34的U-net生成器架構，增添了更多上色可能性，之後結合SLIC演算法生成的超像素圖片，進行後處理，使得整體畫面較為協調，最後設計圖形使用者介面作為呈現，實現黑白漫畫的全自動上色。

　　相較於其他自動上色工具，本專題現階段已經能夠提供航海王漫畫一個良好的上色效果，上色模型準確學習到漫畫物件的特徵，像是漫畫對話框、頁邊與間隔通常不上色、天空與海洋會是藍色、人臉是皮膚色等。但由於不同漫畫的物件線條以及上色風格差異頗大，上色模型需要面對前所未見的資訊，導致上色成果不甚理想，因此，若要將模型運用在其他漫畫的上色工作上，仍然是一大挑戰，同時也作為我們未來持續努力的方向。

1. **未來展望**

　　由於輸出圖片經過縮放，導致畫面解析度低及處理後的對話框文字模糊，下一步會嘗試加入文字區塊辨識，偵測最初輸入圖片的文字區域，獲取對應位置的像素以取代輸出圖片的文字區域，保持文字的原始樣式，至於圖片解析度低的問題可以運用超解析度還原的模型，使得圖片放大後可以保持較高的解析度。此外，也期望能夠進一步優化圖形使用者介面，提供濾鏡後處理功能及簡易的圖片編輯儲存功能，帶來更棒的應用體驗。

1. **參考文獻**

[1] Paulina Hensman, Kiyoharu Aizawa. cGAN-based Manga Colorization Using a Single Training Image. arXiv:1706.06918v1 [cs.GR] 21 Jun 2017.

[2] Gustav Larsson1, Michael Maire2, and Gregory Shakhnarovich2. Learning Representations for Automatic Colorization. arXiv:1603.06668v3 [cs.CV] 13 Aug 2017.

[3] CHIE FURUSAWA\*, KAZUYUKI HIROSHIBA\*, KEISUKE OGAKI, YURI ODAGIRI, YURI ODAGIRI, DWANGO Co., Ltd. Comicolorization: Semi-Automatic Manga Colorization. arXiv:1706.06759v4 [cs.CV] 28 Sep 2017.

[4] Yiyang Yan. ZING! POW! – COMIC COLORIZATION WITH CGAN. Fall 2018.

[5] Yifan Liu 1 , 2 , Zengchang Qin 1 , Zhenbo Luo 2 , and Hua Wang 2. Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks. arXiv:1705.01908v2 [cs.CV] 7 May 2017.

[6]梁培俊, 刘怡俊. 基于条件生成对抗网络的漫画手绘图上色方法《计算机应用研究》2019年第36卷第2期.

[7] Ian J. Goodfellow, Jean Pouget-Abadie∗ , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. arXiv:1406.2661v1 [stat.ML] 10 Jun 2014.

110學年度科技部大專生研究計畫補助

計畫編號：110-2813-C-033-021-E

學生 黃于九