淡江大學資訊工程學系資訊網路與多媒體碩士班

碩士論文

指導教授： 洪文斌 博士

基於生成對抗網路 GAN 模型之書法字體生成系統

Calligraphy Character Generation System Based on the Generative Adversarial Networks Model

研究生： 連禹睿 撰

中華民國 110 年 6 月

致謝

首先要感謝指導教授洪文斌老師在這兩年的教導以及寶貴的意見，這兩年老師給了我很多機會去學習，讓我在這些經驗中獲益良多，也在其中學到了與人合作以及待人處事的道理。另外要特別感謝淡江大學出借高規格的電腦，讓我能如期的完成實驗，減少了大量訓練模型所需的時間。

感謝簡孝羽學長，在我剛入學時給予了很多的幫助以及指導，以及在緊要關頭時，協助我在學術上的問題，最後祝福翁宥杰以及陳鴻鈞同學在未來的路上平步青雲，能有豐碩的研究成果。

再來我想感謝我的父母親對我的栽培和支持，完成碩士學位一直都是我父母親的心願，也非常支持我讀資工系，感謝我的父母親對我付出無私的愛和照顧，您們都辛苦了！

論文名稱：基於生成對抗網路 GAN 模型之書法字體生成系統　頁數：58

校系(所)組別：淡江大學資訊工程學系資訊網路與多媒體碩士班

畢業時間及提要別：109 學年度第 2 學期碩士學位論文提要

研究生：連禹睿　　指導教授：洪文斌 博士

論文提要內容：

書法是我國文字書寫的藝術表現，有豐富的形狀和變化。本論文嘗試使用生成對抗網路來模擬生成歷代名家的書法字體。我們以書法名家的書法字體影像為學習的訓練資料集。本研究中，我們以 Zi2Zi 的方法為基礎，實作書法字體風格轉換的生成對抗網路模型，並探討其他研究的一些方法對模型產生的影響，嘗試得出最佳化的深度學習模型。我們探討加入 U-Net、類別內嵌(Category Embedding)、和 HAN 等方法，以及訓練資料集大小對模型造成的影響。在最後實驗中，我們比較了 pix2pix 模型、Zi2Zi 模型、HAN 模型、和 HAN-Zi2Zi 模型。經過實驗後發現，加入 U-Net 和 Category Embedding 都對模型的成果有所幫助，而使用越多字體進行訓練會有越好的成效。另外，HAN-Zi2Zi 效果最好。

關鍵字：GAN、HAN、ZI2ZI、Calligraphic

Title of Thesis: Calligraphy Character Generation System Based on the Generative Adversarial Networks Model

Total pages: 58

Key word: Deep Learning, GAN, ZI2ZI, HAN, Calligraphic

Name of Institute: MASTER'S PROGRAM IN NETWORKING AND MULTIMEDIA, DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION ENGINEERING, TAMKANG UNIVERSITY

Graduate date: 06/2021

Degree conferred: Mater degree

Name of student: LIEN,YU-JUI 連禹睿

Advisor: Dr. WEN-BING,HORNG 洪文斌 博士

Abstract:

Calligraphy is the artistic expression of Chinese character writing, with rich shapes and variations. This research attempts to use the Generative Adversarial Network to generate the calligraphy characters of famous calligraphers. We use calligraphy font images of famous calligraphy masters as training materials for learning. In this research, based on the Zi2Zi method, we implemented a generational confrontation network model for calligraphic font style conversion, and explored the impact of some other research methods on the model, and tried to arrive at an optimized deep learning model. We discuss methods such as adding U-Net, Category Embedding, and HAN, as well as the impact of the size of the training data set on the model. In the final experiment, we compared the pix2pix model, Zi2Zi model, HAN model, and HAN-Zi2Zi model. After experimentation, it is found that adding U-Net and Category Embedding are helpful to the results of the model, and the more fonts are used for training, the better the results will be. In addition, HAN-Zi2Zi works best.

目錄

圖目錄

表目錄

第一章 緒論

1.1 前言

由於科技的進步，人工智慧(Artificial Intelligence, AI)技術又再度的興起，近年來科學家們紛紛投入人工智慧的研究領域。過去因電腦硬體設備無法有效的提供給研究者們實現理論，所以人工智慧在過去的研究中，數次興起但最後又跌落了谷底。最近這幾年因硬體技術方面得到巨幅的提升，所以人們憑藉著硬體強大的效能，人工智慧的研究又再度進入了高峰期。

近年來,生成對抗網路(Generative Adversarial Networks, GAN)逐漸受到大家的注目,有許多研究者陸陸續續地開始加入研究，並開發許多各式各樣的模型，至今已經是個全面性的技術，而且有許多耳熟能詳的應用，最為知名的應用就是各種圖像風格轉換的 App 與前陣子造成許多名人困擾的換臉。以技術面來說，GAN 確實是近年來，深度學習(Deep Learning)發展中相當重要一環。

1.2 研究動機與目的

書法字體是古人在文書處理上都會用到的工具，是一些具有藝術性的字體，如行書體、草書、楷書等等。書法字體是非常久遠以前的字跡，因此有許多常用字或是稀有字上的缺失，若能使用深度學習的方式作轉換，如此一來就能大幅減少創造字體所需的時間。因此，本研究的目的就是在建立一個

字體轉換的深度學習模型，利用生成對抗網路(GAN)的方式實作，並對一些方法進行實驗比較及深入探討，期望能使字體風格轉換模型達到較好的效果。

1.3 論文架構

本論文架構主要分為五個章節，如下:

第一章 緒論

分為前言以及研究動機與目的。

第二章 文獻探討:

此章節分別介紹 GAN 生成對抗網路歷史的運作流程以及相關架構。

第三章 系統架構設計:

主要分為流程架構、資料來源以及模型訓練。

第四章 系統實作與結果討論:

評估系統架構之成效效與實驗結果呈現。

第五章 結論與未來展望:

總結實驗結果以及檢討未來可改進與拓展之方向。

第二章　文獻探討

在過去的文獻中，有許多的漢字生成方法，本篇論文所提出的方法為圖像轉換的問題，討論相關技術如下。

2.1 圖像轉換

圖像轉換是一種視覺上的問題，是在學習輸入與輸出之間的映射函數，使用這類方法模型有許多應用[15,8,11,9,22,2,21]，眾多的方法大部份都是基於 GAN[7]的條件上。

2.1.1 生成對抗網路

2014 由 Goodfellow 等人[7]提出生成對抗網路(GAN)的概念, 大量減少了深度學習所需要的資料量。GAN 主要透過生成網路(Generator,G)和判別網路(Discriminator,D)不斷比較，進而讓生成網路學習到資料的分布，以圖片為例，訓練完成後，生成網路可以從一段隨機數中生成出逼真的圖像，好比兩個角色，一個是偽造者，他不斷製造假鈔，另一個角色是鑑賞家，不斷從偽造者拿取假鈔票，判斷是

真是假，偽造者就根據鑑賞家判斷結果的回饋，不斷改良，最後假鈔票變成真假難辨，這就是 GAN 的概念。G、D 主要的功能：G 是生成的網路，收到一個隨機數，通過隨機數生成影像 D 是判別網路，判斷一張影像是否為真實的，它的輸入為一張圖片，輸出為數字，此數字代表此影像是否真實的概率，最高為 1，代表百分之百為真實圖像，輸出為 0，代表完全不為真實圖像。

在 GAN 架構下，G 就稱為生成模型（generative model），D 稱為判別模型（discriminative model），簡單架構如圖 1。

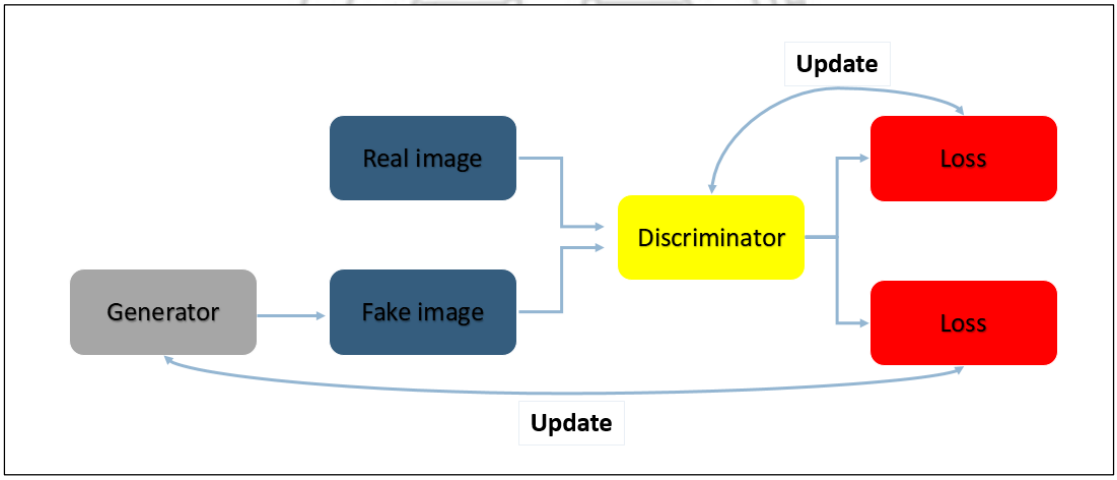


圖 1: GAN 架構圖

接下來介紹目標函數，如公式所示，G 代表著生成器，D 則代表鑑別器，y 為真實數據，Pdata 為真實數據的分布，x 為隨機輸入的數據，從鑑別器的角度為了區分真實數據 y 與假圖像 G(x)，鑑別器 D 希望原始數據 Pdata 出來的判斷越大越好，而 D(G(x))要越小越好，

也就是說希望 V(D,G)越大越好，而站在生成器的角度來看，生成器希望生成出來的假圖像 G(x)可以騙果鑑別器 D，也就是 D(G(x))可以盡可能的越大越好，也就是說 V(D,G)盡可能的越小越好，GAN 就是在這兩個模型中互相對抗，最後達到最佳的效果。

$$\min_G \max_D V(G,D) =$$
$$E_{y\sim Pdata(y)}[\log D(y)] + E_{x\sim Pdata(x)}[\log D(1-D(G(x)))] \quad (1)$$

2.1.2 Pix2Pix

Pix2Pix 是第一個做圖像轉換的方法，2017 由 Phillip Isola 等人[15]提出基於生成對抗網路圖像轉換的研究。是 GAN 網路中的一種，主要是採用 CGAN [12]網路的結構，它依然包括了一個生成器和一個判別器。生成器採用的是一個 U-net [14]的結構，其結構有點類似 Encoder-decoder [3]，總共包含 15 層，分別有 8 層卷積層作為 encoder，7 層反卷積層作為 decoder，與傳統的 encoder-decoder 不同的是引入了一個叫做“skip-connect”的技巧，即每一層反卷積(deconvolution)層的輸入都是：前一層的輸出加上與該層對稱的卷積層的輸出，從而保證 encoder 的資訊在 decoder 時可以不斷地被重新記憶，使得生成的影像儘可能保留原影象的一些資訊。

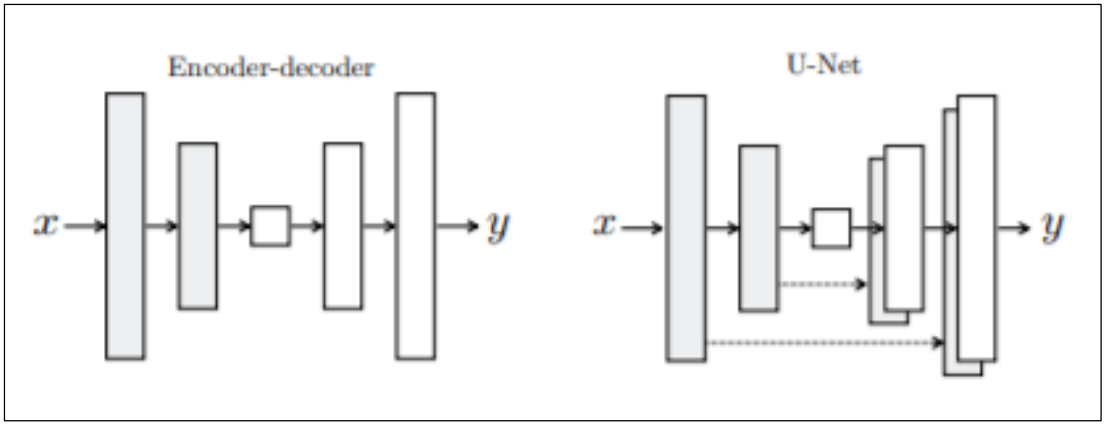


圖 2: U-Net 架構圖[15]

圖 2 中，U-Net [14]也是 Encoder-Decoder 模型。Encoder 和 Decoder 是對稱的。所謂的 U-Net 是將第 i 層拼接到第 n-i 層，這樣做是因為第 i 層和第 n-i 層的影像大小是一致的，可以認為他們承載著類似的資訊。架構如圖 3 所示，為 Pix2pix 的架構圖。

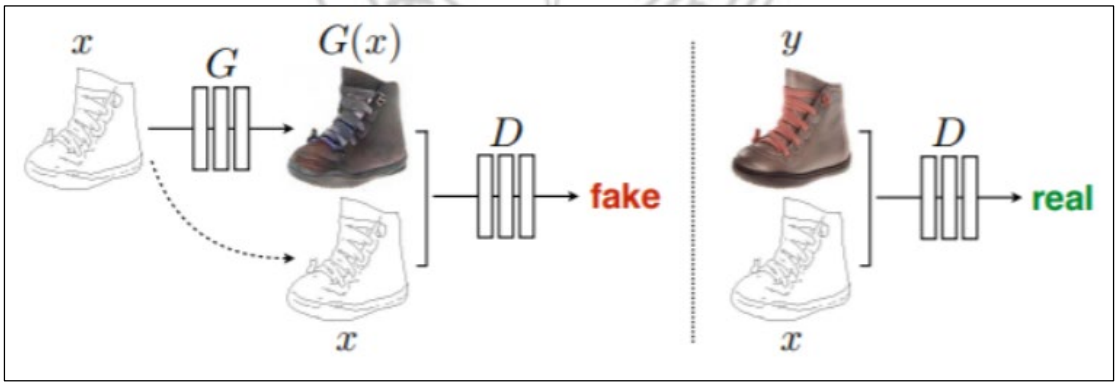


圖 3: Pix2pix 架構圖[15]

Pix2pix[15]判別網絡採用 PatchGAN 結構，也就是說把圖像等分成多個固定大小的 Patch，分別判斷每個 Patch 的真假，最後再取平均值作為 D 最後的輸出，如圖 4 所示。也使用了 L1-Loss 來減少輸出圖

像和訓練圖像之間的差異。Pix2Pix [15]的限制是需要成對的圖像來訓練模型，很難實用在物體轉換上。

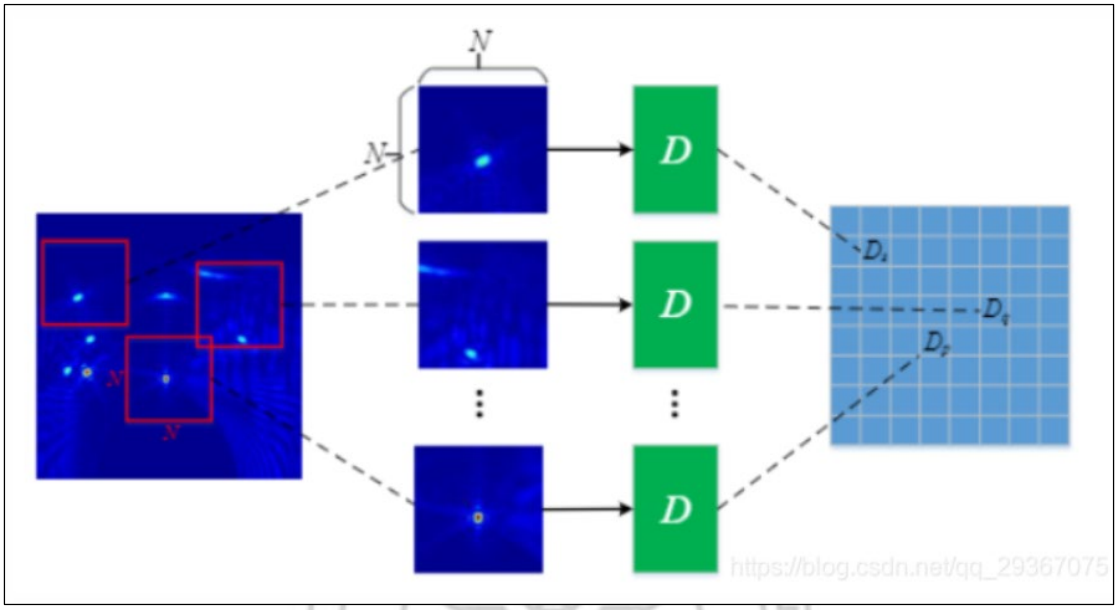


圖 4: PatchGAN 圖形表示[15]

Pix2pix 在生成網路上所使用的 L1-Loss 如下公式所示:

$$\mathcal{L}_{L1}(\mathrm{G}) = \mathbb{E}_{x,y,z}[\|y - \mathrm{G}(x,z)\|_1] \tag{2}$$

並綜合了 cGAN 架構公式如下所示:

$$\mathrm{G}^* = \arg \min_{G} \max_{D} \mathcal{L}_{cGAN}(\mathrm{G},\mathrm{D}) + \lambda \mathcal{L}_{L1}(\mathrm{G}) \tag{3}$$

其中 G 試圖最小化目標，而鑑別器 D 試圖最大化目標。

2.1.3 CycleGAN

2017 由 Zhu Park 與 Efros [9]提出迴圈生成對抗網路的研究，不僅為計算機圖形學等領域的研究人員所用，也成為視覺藝術家廣泛使用的工具。Pix2Pix [15]能生成出與訓練風格類似的影像，但無法指定具體的圖像，CycleGAN 可以有效解決這個問題，CycleGAN 有一項重大的發展，以往資料通常需要成對的影響，而一般視覺問題上很難找到成對高質量的影像來提供模型學習，由於一般生成對抗模型需依賴配對的影像，除非有目的的產生成對影響，否則無法訓練，並且很容易造成生成之數據偏差，一般成對以及非成對影像說明，如圖 5 所示。

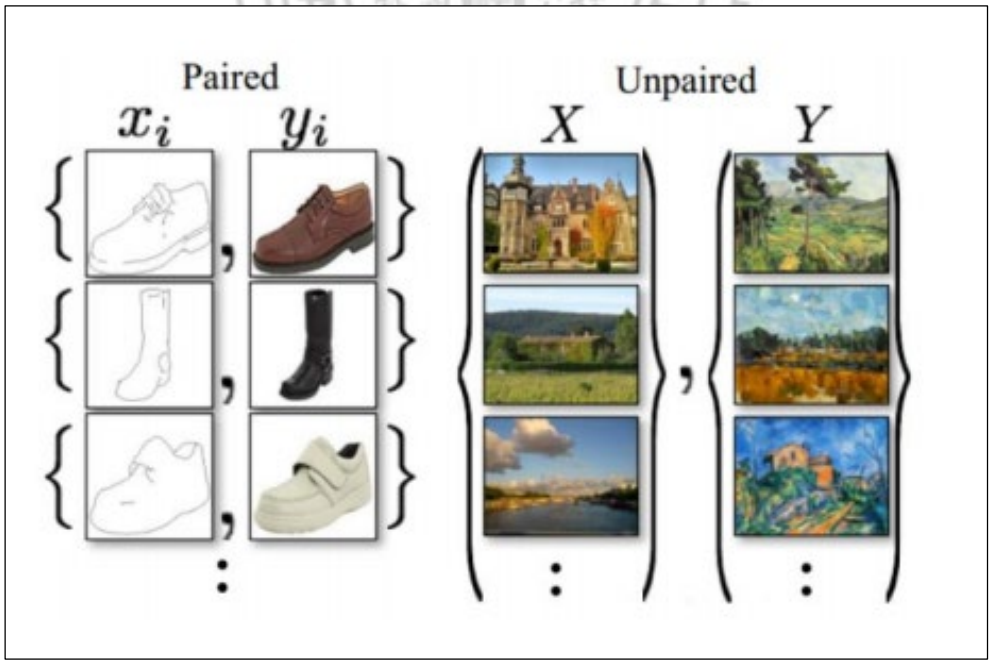


圖 5: CycleGAN 概念圖[9]

CycleGAN 之原理圖解，如圖 6 所示，輸入一個真實圖像 A，透過生成器 G_AB 生成出想轉換成 B 影像之假影像 B，再由判別器 D_B 判斷是否與真實 B 影像相似，為了避免生成出的影像過度學習目標而上師了原本真實影像 A 的特徵，需要一個生成器再將升成的假影像 B 生成出與真實影像相似的假影像 A，假影像 A 與輸入真實影像 A 越相似越好，代表還是保留原影像的特徵。

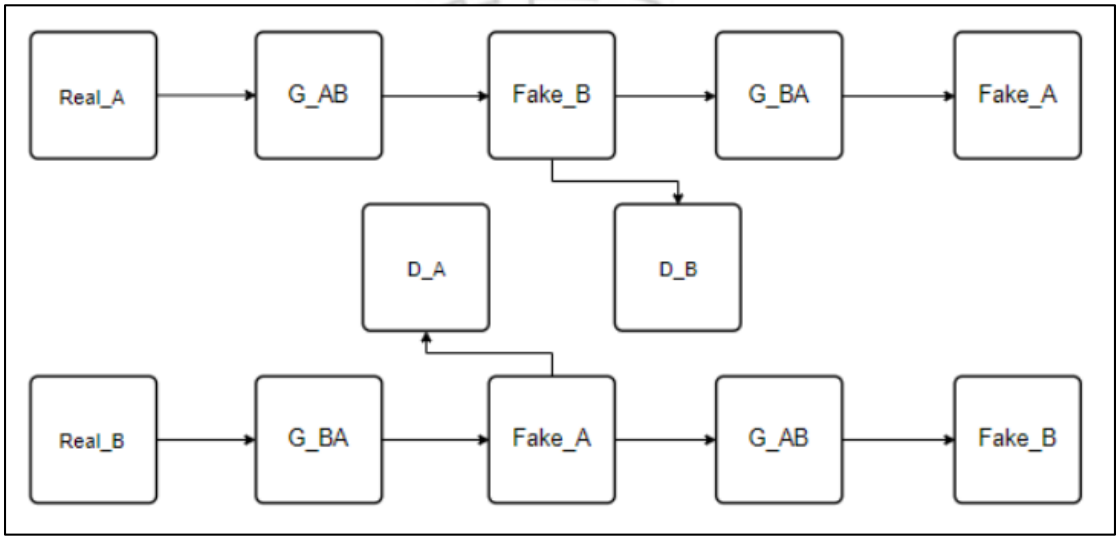


圖 6: CycleGAN 架構圖[9]

CycleGAN 損失函數主要分為兩個部分，分別是對抗損失以及循環一致性損失函數，需先定義以下兩個輸入分別為 X 與 Y，兩個鑑別器為 Dx 與 Dy，兩個生成器為 G 與 F。

$LGAN(G,D_y,X,Y)=$

$$E_{y\sim Pdata(y)}[\log Dy(y)] + E_{x\sim Pdata(x)}[\log(1-Dy(G(x)))] \quad (4)$$

生成器 G 會生成出 G(x)，盡可能讓他接近於從 Y 取出的資料，而 Dy 則會試著分辨出 G(x)與真實分布的資料 y，在訓練過程中，生成器 G 的目標是最小化這個損失函數，Dy 則相反的想要最大化這個損失函數，對於生成器 F 也是相同的原理模式。

最終的損失函數結合了生成器 G 與生成器 F 的對抗損失函數以及循環一致性損失函數如下：

$$Lcyc(G,F)=E[\|F(G(x))-x\|_1]+E[\|G(F(y))-y\|_1] \quad (5)$$

2.2 Zi2Zi 漢字生成法

Zi2Zi [18]是第一個使用 GAN 生成漢字的方法，2017 由 Github 用戶 Kaonashi-tyc 提出的方法，Pix2Pix[15] 模型並沒有解決這種一對多的關係。作者受「谷歌零數據機器翻譯」論文(Zero-shot GNMT)[13]的啟發，想出來了類別嵌入(category embedding)的方法，將不可訓練的高斯噪聲作為風格嵌入(style embedding)與漢字嵌入(character embedding)串聯起來，之後再一併進入解碼器。這樣，解碼器仍舊將

同一個漢字映射為同一個向量，但是，解碼器會同時考慮漢字和風格兩個嵌入來生成目標漢字(target character)。如圖 7 所示。

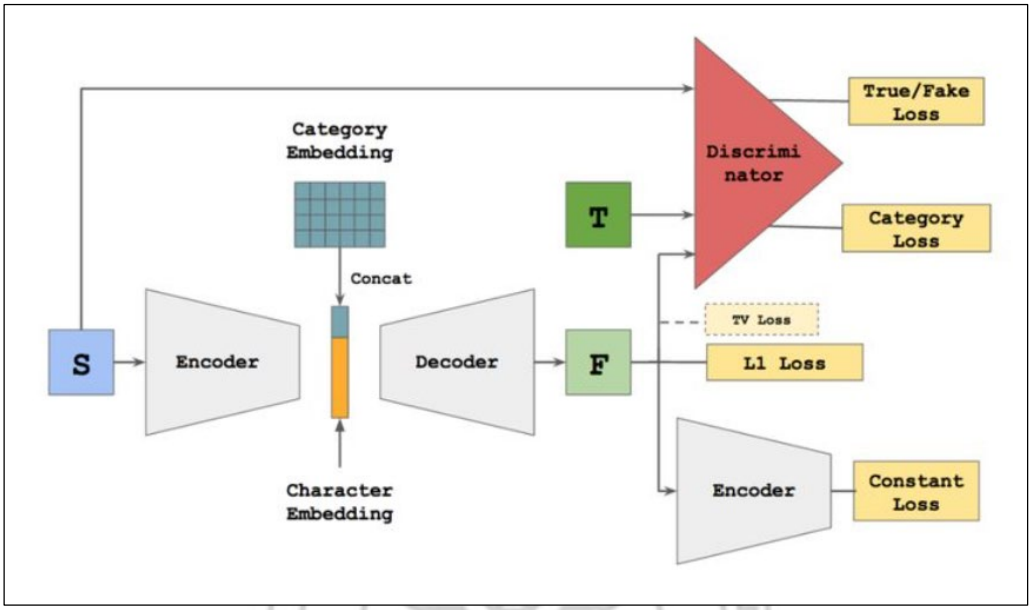


圖 7: Zi2Zi 架構圖[18]

但是，作者發現，這樣做之後又出現了一個新的問題：模型開始將各種風格弄混淆並且混合在一起，生成的漢字什麼也不像了。因此採用了 AC-GAN [1]模型中的 multi-class category loss，把這個 loss 加到判別器上面，一旦出現混淆或者風格混合，就「懲罰」判別器。也採用了 DTN [19]中的 constant loss 得以提高輸出質量。

2.3 AEGG 漢字生成法

Zi2Zi 是一個開園項目，並未發表論文，而第一篇使用 GAN 來生成漢字的論文是 AEGG，2017 由 Pengyuan Lyu 等人[16]提出漢字書法字體生成自動編碼器的研究，也是基於 Pix2Pix [15]，新增了 Encoder-Decoder，在訓練過程中進行監督工作，並提供訊息，與 Zi2Zi [18]不同於，AEGG 僅只能單一訓練，如圖 8 所示。

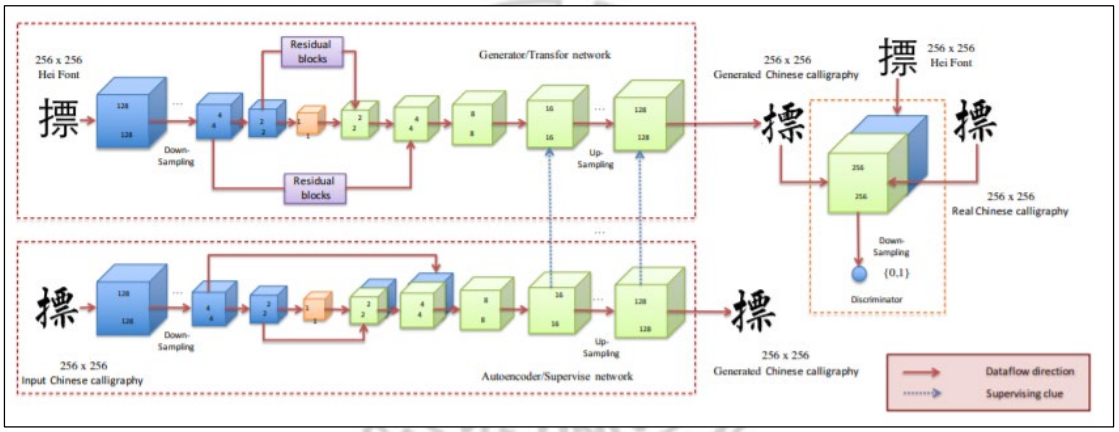


圖 8: AEGG 架構圖[16]

2.4 HAN 漢字生成法

緊追其後的還有一篇使用 GAN 來生成漢字的論文是 HAN，2017 由 Jie Chang 等人[10]提出基於層次生成對抗網路之中文字體轉換。也是基於 Pix2Pix [15]，對於字體轉換任務，不同的字體可能共享相同的部首。這是字體轉換方法可以利用的一個很好的特性，即學習源樣式和目標樣式之間部首的直接映射。但是，有時在一種特定字體中，

相同的部首在不同的字符中可能會出現完全不同的情況，如圖 9 所示。

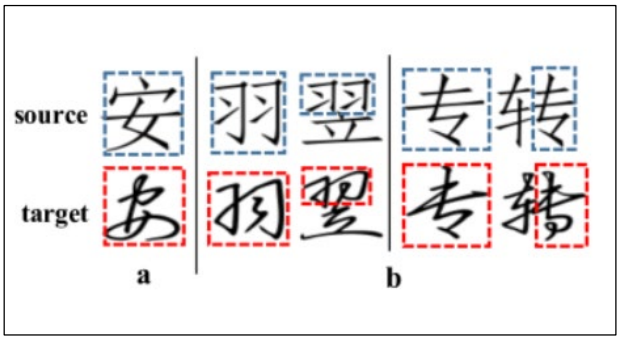


圖 9: 部首比較[10]

藉由 GoogLeNet [4]的思想，提出了一種分層解碼器，它在多個解碼層中生成人工圖像，有望幫助解碼器在其隱藏層中學習更好的表示。分層解碼器嘗試最大限度地保留不同解碼層中的全局拓撲結構，同時考慮隱藏層中解碼的局部特徵，從而使傳輸網絡能夠生成接近真實的字體圖像，架構如下圖 10 所示。

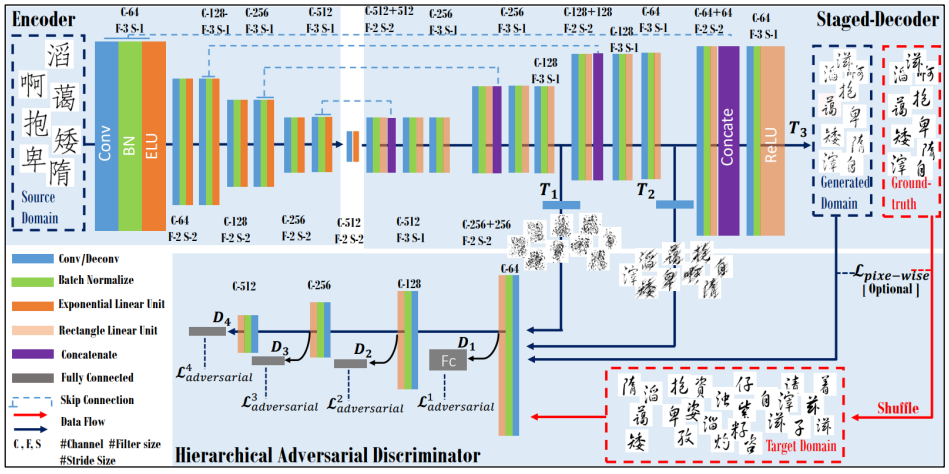


圖 10: HAN 架構圖[10]

2.5 CalliGAN 漢字生成法

在 2020 年由台大所發表的一篇也是使用 GAN 來生成漢字的論文，2020 由 Shan-Jean Wu 等人[17]提出風格與結構之中國書法生成器，基於 Pix2Pix [15]，在即中也使用了 U-Net [14]作為生成器的架構，由於考慮到漢字是由許多筆畫組成的，與學習給定語言之間的翻譯模式不同，EMD [20]和 SA-VAE [5]都是獨立的內容和樣式作為兩個不相關的域，並使用兩個獨立的編碼器對它們進行建模。然而，它們的科技細節是不同的。EMD [20]在雙線性混合網絡中混合風格和內容的潛在特徵，生成輸出影像通過影像解碼器。因此，它的訓練樣本很特別。一個樣本包括兩組訓練圖像，一個用於內容，另一個用於樣式。相反，SA-VAE [5]採用順序方法。它首先認識到從給定圖像中選取字元，然後將識別出的字元編碼成特殊程式碼。SA-VAE [5]表明，漢字的主要知識有助於提高輸出影像的質量。因此將漢字分解了 517 個部件，並進行編碼配置，利用 LSTM [6]將各個部件特徵提取，當作生成條件做訓練，架構如圖 11 所示。

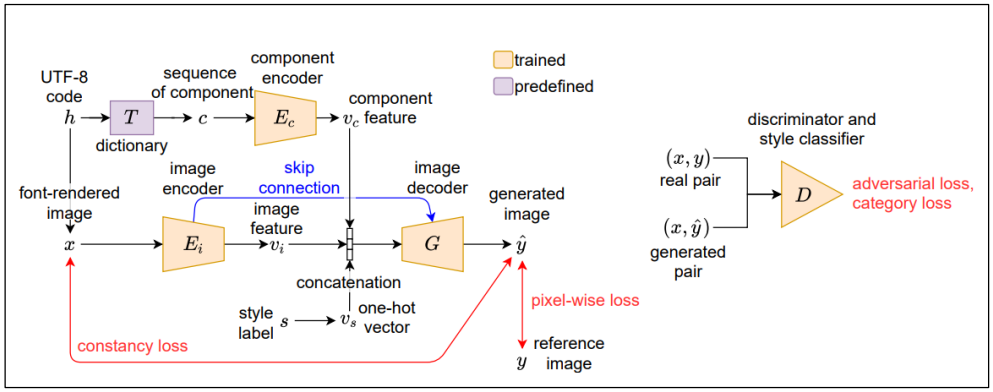


圖 11: CalliGAN 架構圖[17]

給定一個 h，通過渲染圖像 x，在通過圖像編碼器 Ei 生成出特徵向量 vi，同時也通過字典 T 獲取組件序列 c，在通過編碼器 Ec 生成出組件特徵，最後將 vc，vi 以及樣式標籤 S 轉換的向量 vs 一起通過解碼器 G 生成圖像 y，而綜合公式如下：

$$\mathcal{L}=\log \mathrm{D}(x,y)+\log(1-\mathrm{D}(x,\hat{y}))+\lambda_{\mathrm{p}}\mathrm{L}_{\mathrm{p}}+\lambda_{\mathrm{c}}\mathrm{L}_{\mathrm{c}}+\lambda_{\mathrm{s}}\mathrm{L}_{s} \qquad (6)$$

第三章 研究方法

在本論文中，我們應用了 CalliGAN [17]中所使用的資料集，來自網路上所提供的陳忠建書法字體，並基於 Pix2Pix [15]，在其中結合了 HAN [10]的分層解碼器外，也結合了 Zi2Zi [18]中的類別嵌入，來提高訓練品質。圖 12 為本論文研究方法介紹章節。

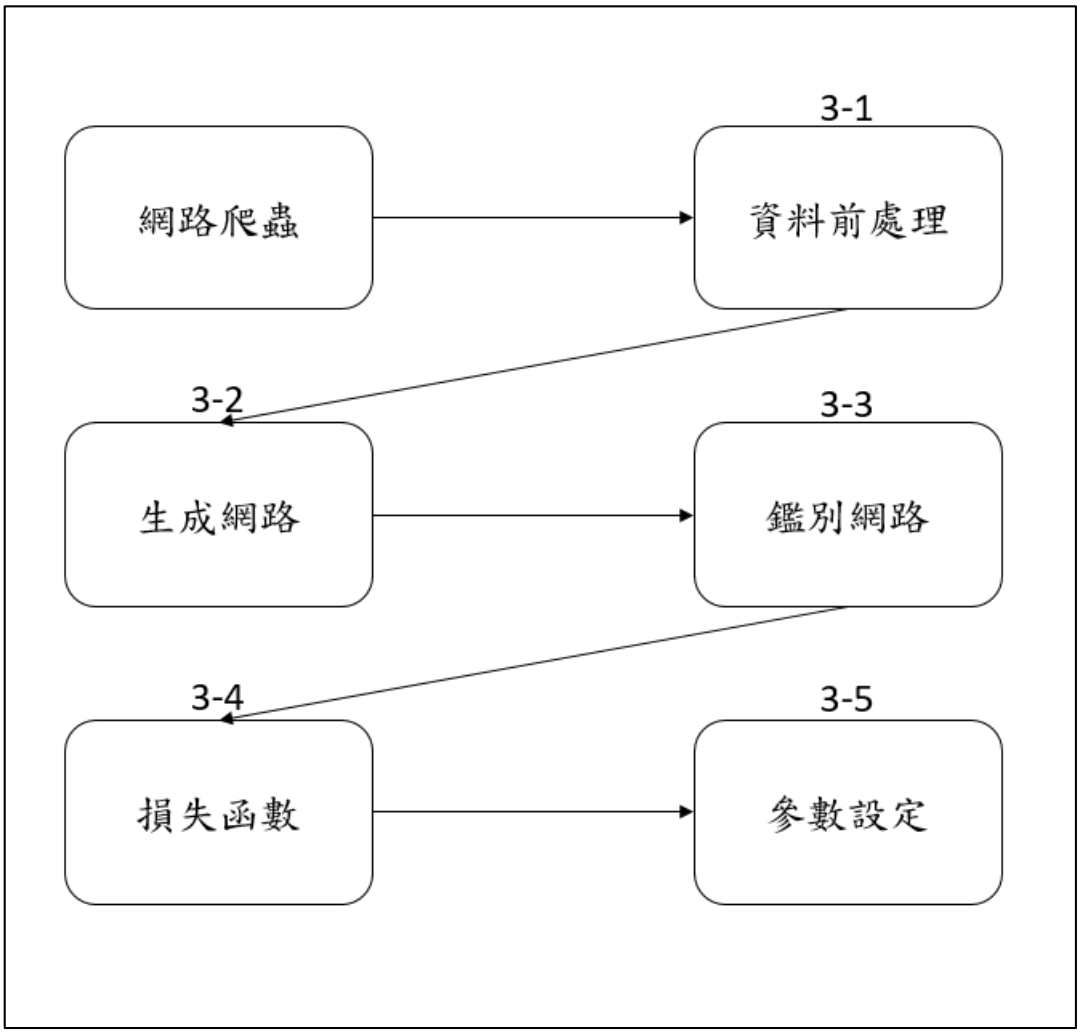


圖 12: 實驗介紹章節

首先，在 CalliGAN 中所使用的資料庫，將作為本論文的資料庫，是以陳忠建書法資料庫中的 7 種字體作為本論文訓練資料，由圖 13 所示。

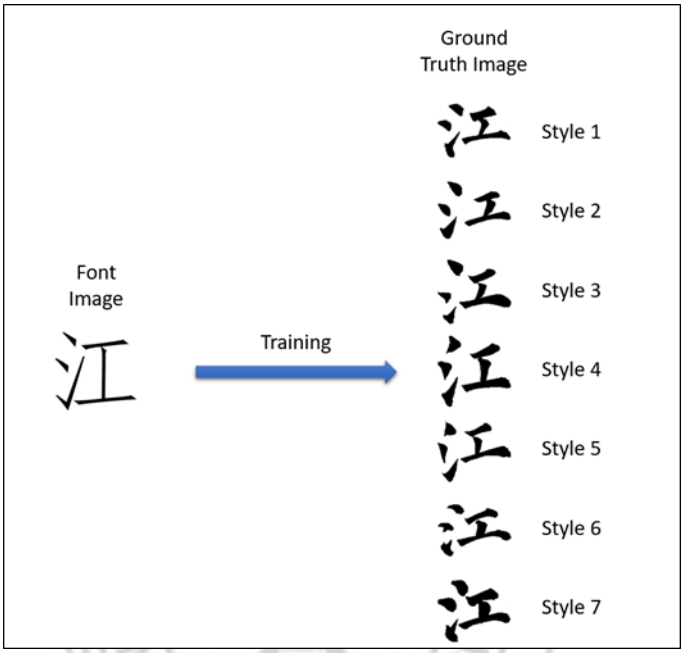


圖 13: 源字體（仿宋）以及 7 種字體

由圖 14 表示，利用網路爬蟲，將陳忠建書法資料庫中的 7 種字體抓取，並做資料前處理，將圖像轉換成 256 x 256，在其中 7 種字體字數並非一致，因此須找出共同擁有的字體。

將書法字體格式 TTF 檔轉換成圖像格式 JPG 檔，並將輸入字體和目標字體合併成對應圖像 256 x 512，再轉換格式至訓練格式 OBJ 檔，訓練資料與測試資料占比 9：1，並將資料進入各個模型之中做訓練。

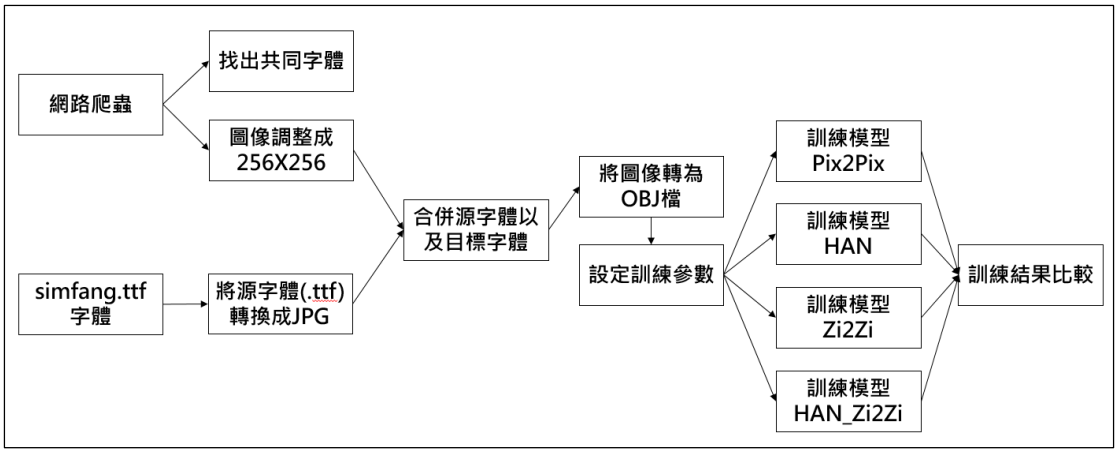


圖 14: 實驗流程圖

而在 3-1 中會介紹本論文資料庫的前處理方式，以及訓練與測試的占比。其後，在 3-2 中會介紹生成網路的方法與架構。接者，在 3-3 中會介紹對抗網路的方法與架構，以及相關的損失函數。在 3-4 中會介紹相關的損失函數。最後，3-5 中會介紹本論文所使用的相關參數。

3.1 資料前處理

本論文所使用的是陳忠建書法字體，在這些書法圖像中的每一個圖像大小均不相同，此資料庫的圖像固定常邊均為 140，且均為白底黑字，為了讓圖像能有相同的固定長度與寬度，因此在每一張圖像背後加入了一張相同長度與寬度的 140X140 的白色背景圖，如圖 15 所示。

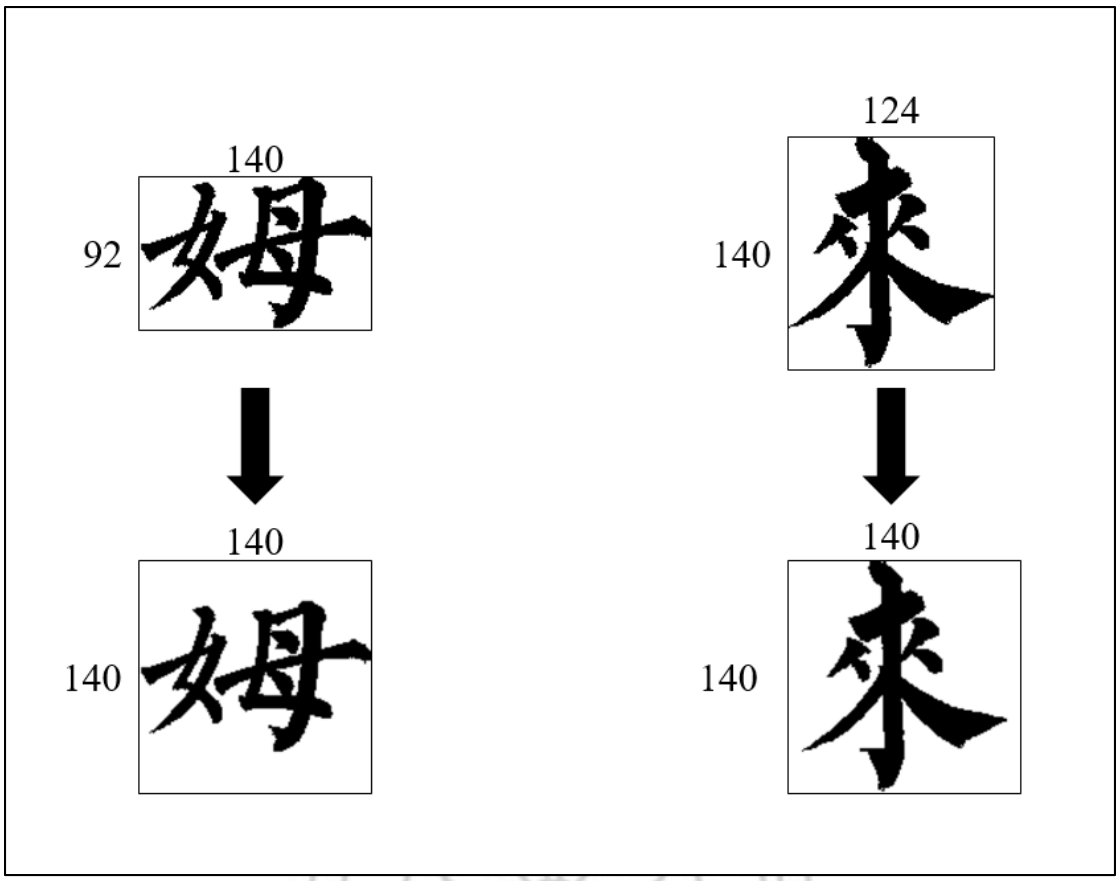


圖 15: 圖像前處理

再將圖像等比例放大至 256x256，在 7 種字體中，每一種字體的個數均不一致，且有些部分的字體均有重複，為了統一訓練比例以及比較，本文將每一種字體中的共有字體抓取出，共有 5000 個字，如表 1 所示。

表 1: 訓練資料個數

	Style 1	Style 2	Style 3	Style 4	Style 5	Style 6	Style 7
原資料	6171	7008	7159	6901	6999	6308	7006
共有字數	5000	5000	5000	5000	5000	5000	5000

本文的方法是目標圖像與源圖像做成對的訓練，因此在源字體部分也要輸出為圖像，如圖 16 所示，左邊為源字體，右邊為目標字體，合併為訓練資料。

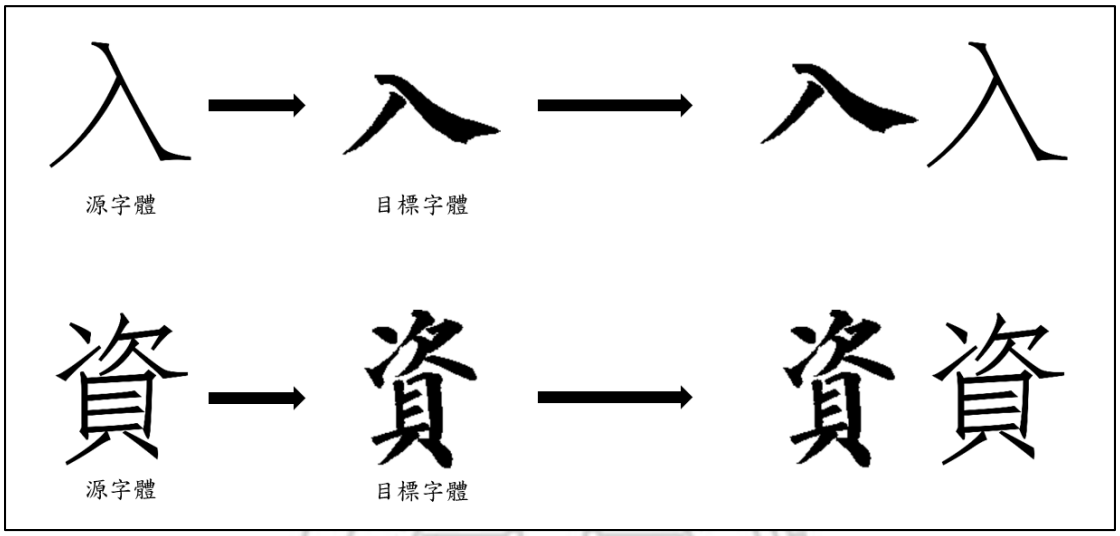


圖 16: 訓練資料合併

3.2 生成網路

本文在生成網路結構主要在 U-Net 的基礎上做一些修改，因為在任何圖像的相對位置特徵資訊對漢子都是至關重要的，因此使用 U-Net 來合併圖像的特徵資訊，主要減少尺寸並保留重要資訊。然而，在某些情況下會導致部分特徵資訊的流失，因此生成網路中額外選取了特徵圖像做訓練。

而本文在解碼器(Decoder)中，主要運用了 HAN 的基礎方法，利用解碼器中間的特徵資訊野生成了圖像，並與最後一個生成圖像，一起發送到鑑別器(Discriminator) 。然而，本文只測量了最後生成的圖

像與目標圖像之間的差異。與此同時，中間生成出的特徵圖像損耗可以助於優化生成網路，並可以對生成網路中的參數進行正規化處理，也可以解決一定程度上的過擬合問題。

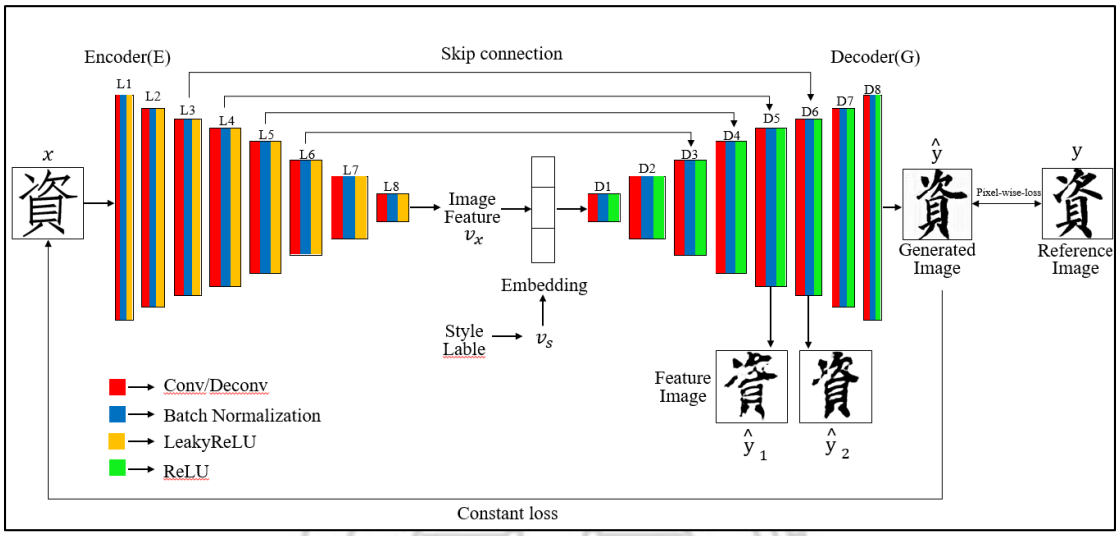


圖 17: 生成網路流程

由上圖 17 所示，輸入 x 為 256x256 的圖像,輸入至編碼器(Encoder, E)中，本文使用 8 個卷積層，其中在每一層卷積都使用了 LeakyReLU 以及 BatchNormalize，最後得到特徵向量，LeakyReLU 的負數斜率為 0.2。為了產生 7 個樣式的字體，本文運用了 Zi2Zi 的 Category Embedding 做字體樣式的分類，在與特徵向量一起送入至解碼器中。而在解碼器中，使用了 8 層卷積層，其中也在每一層卷積都加入了 ReLU 以及 BatchNormalize，最後輸出了生成圖像 y，ReLU 的斜率為 0.2，每個生成圖像 y 都會與目標圖像做點數差異(pixe-wise)，再參考了 DTN [19]網路中的概念，加入了 constant loss，輸入圖像和生成圖

像十分相似，因此在嵌入空間中的位置也十分相近，所以 constant loss 迫使解碼器保留生成漢字的內容，加快收斂數度。在解碼器的上採樣中，在 L3,L4,L5,L6 做 Skip connection 合併保留特徵，並在 D5,D6 兩層的特徵輸出成圖像，與最後生成的圖像送入鑑別器中。

表 2: Encoder 詳細架構

Layer	Feature Size	Number of Filters
Input	256x256	1
Conv2d_1	128x128	64
Conv2d_2	64x64	128
Conv2d_3	32x32	256
Conv2d_4	16x16	512
Conv2d_5	8x8	512
Conv2d_6	4x4	512
Conv2d_7	2x2	512
Conv2d_8	1x1	512

表 3: Decoder 詳細架構

Layer	Feature Size	Number of Filters
DeConv_1	2x2	512
DeConv_2	4x4	512
Conv2d_6+DeConv_2	4x4	1024
DeConv_3	8x8	512
Conv2d_5+DeConv_3	8x8	1024
DeConv_4	16x16	512
Conv2d_4+DeConv_4	16x16	1024
DeConv_5	32x32	256
Conv2d_3+DeConv_5	32x32	512
DeConv_6	64x64	128
DeConv_7	128x128	64
DeConv_8	256x256	1

Encoder 與 Decoder 的架構會如表 2，表 3 所示，在 Encoder 部分，會先經過一個 Convolution layer，然後經過 Batch normalization 步驟，最後再經過 LeakyReLU 層，而在 Decoder 部分，會先經過一個 Deconvolution layer，然後經過 Batch normalization，最後在經過 ReLU 層。

3.3 鑑別網路

本文在對抗網路中，為了解決類別嵌入導致的混淆問題，也參考了 AC-GAN [1]中的 Mutli-class category loss，加入在鑑別器上，一旦出現混淆或是風格混和，就懲罰鑑別器。

由於本文的生成對抗中，也生成出第 5 層與第 6 層的特徵圖像作為對抗目標，因此對抗網路中也多了兩組的對抗資料，如圖 18 所示。

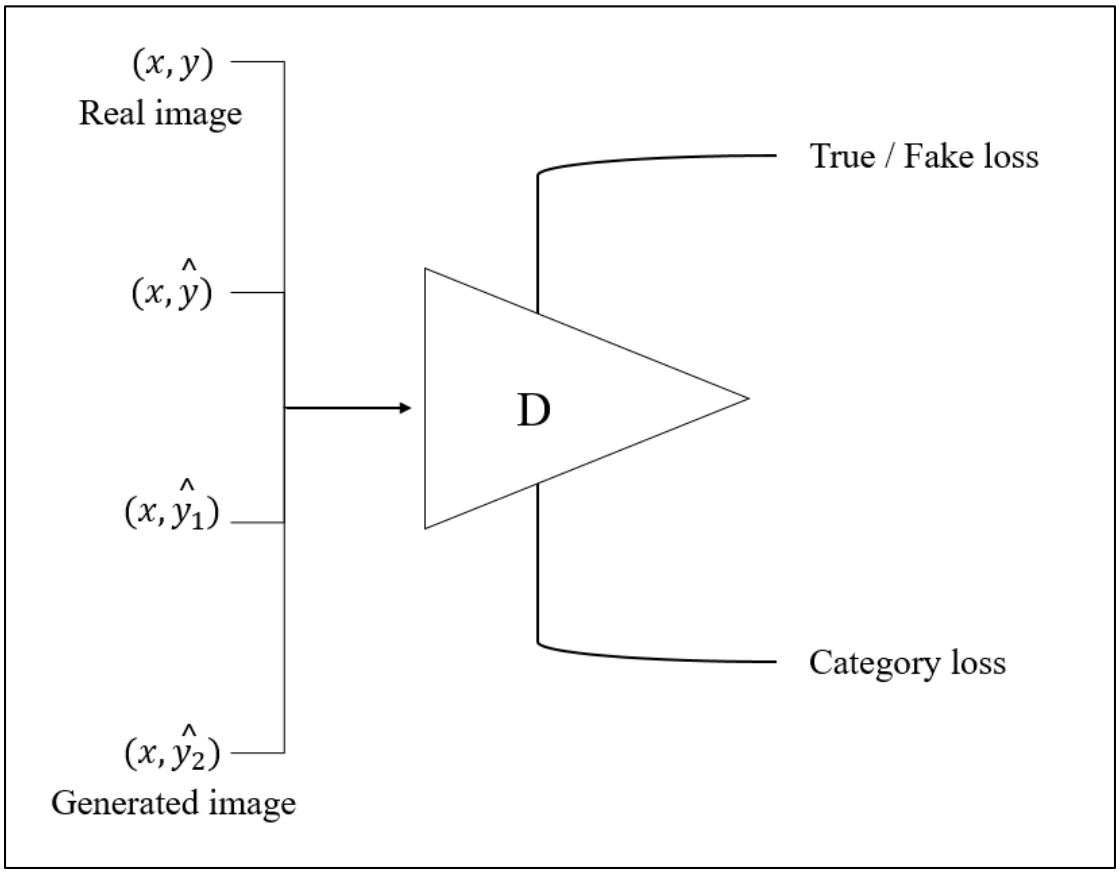


圖 18: 鑑別器流程

然而，為了要能對影像的局部生成得更清晰，所以本文使用的是 PatchGAN 的結構，如圖 19 所示，就是把影像分成 Patch，分別判斷每個 Patch 的真偽，如此更能生成清晰的影像。

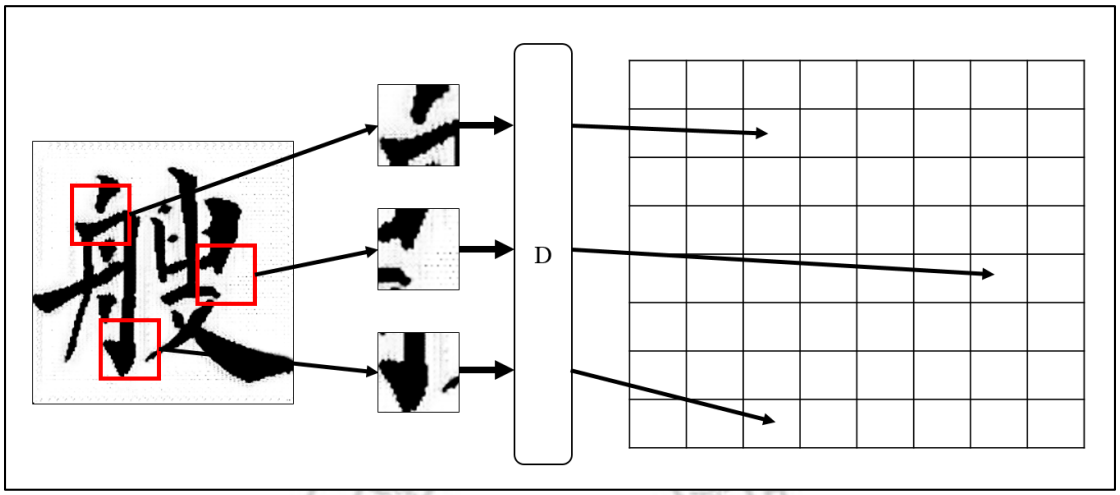


圖 19: PatchGAN 架構圖

Discriminator 的架構會如表 4 所示，會先經過一個 Convolution layer，然後經過 Batch normalization 步驟，最後再經過 LeakyReLU 層，LeakyReLU 的負數斜率為 0.2。

表 4: Discriminator 架構

Layer	Feature Size	Number of Filters
Input	256x256	1
Conv2d_1	128x128	64

Conv2d_2	64x64	128
Conv2d_3	32x32	256
Conv2d_4	30x30	512
Conv2d_5	30x30	1

3.4 損失函數

本文研究損失函數的設計，包含了 4 個損失函數來訓練模型，首先為對抗網路，運用了條件損失函數(Conditional GAN, cGAN)。

$$\mathcal{L}_{cGAN}=[\log \mathrm{D}(x,y)] + [\log (1\text{-}\mathrm{D}(x, \hat{y}\, \hat{y}_1\, \hat{y}_2))] \quad (7)$$

在生成網路中，有多生成出兩個特徵圖像，為了生成網路能夠獲取更多更有效的特徵資訊，因此做為訓練目標。其中 x 是 condition，y 為真實圖像，z 為輸入的隨機雜訊，作為鑑別器的便是結果，圖像像素等級的損失是生成對抗網路中基本的損失函數，目的是為了讓生成圖像與訓練圖有想似的圖樣，因此本文使用 Pixed-wise Loss。

$$\mathcal{L}_{\mathrm{p}} = [\|y - G(x,z)\|_1] \quad (8)$$

本文所輸入 x 與輸出的圖像 y 是相同的字，因此運用了 constant Loss，來促進兩個圖像有相同的特徵向量。

$$\mathcal{L}_{\mathrm{c}} = [\|\mathrm{E}_i(x) - \mathrm{E}_i(\mathrm{G}(x,z)\|_1] \tag{9}$$

由於在生成網路中，使用了類別嵌入(Category Embedding)來進行一對多目標訓練，因此會出現圖像混淆的問題，為了讓生成的圖像保留原有的分配方式，運用了 Category Loss。

$$\mathcal{L}_{\mathrm{s}} = [\log(D_s\,(S|y)) + \log(\,D_s(S|G(x,z)))] \tag{10}$$

而將以上四種 Loss 相加，並加入一個方便調整的權重 k，就是本研究的 Loss Function。

$$\mathcal{L}_{\text{total}} = \left[\mathcal{L}_{\text{cGAN}} + \lambda_{\mathrm{p}}\,\mathcal{L}_{\mathrm{p}} + \lambda_{\mathrm{c}}\,\mathcal{L}_{\mathrm{c}} + \lambda_{\mathrm{s}}\,\mathcal{L}_{\mathrm{s}}\right] \tag{11}$$

3.5 參數設定

本研究用使的 Optimizer 是 Adam 優化器。一般對優化器來說 learning rate 相當重要，太小會花費過多時間學習，而太大將會有 overfitting 的可能。如表 5 所示，本論文使用 learning rate 為 0.0002，而 Epoch 為 200，Batch size 為 32，LeakyReLU 的負數斜率為 0.2，ReLU 的斜率為 0.2。

表 5: 參數設定

訓練參數名稱	參數值
epoch	200
batch size	32
LeakyReLU,ReLU 的斜率	0.2
Optimizer	Adam
learning rate	0.0002

第四章 實驗結果與分析

為了模型訓練，本文訓練資料從陳忠建書法資料庫中下載，所有的圖像都是臨摹古代著名的書法家的書法字體，該字庫涵蓋了 29 種字體，本文使用以下 7 種字體風格：

1. 柳公權
2. 虞世南
3. 褚遂良
4. 歐陽詢-九成宮
5. 歐陽詢-皇甫誕
6. 顏真卿多寶塔體
7. 顏真卿-顏勤禮碑

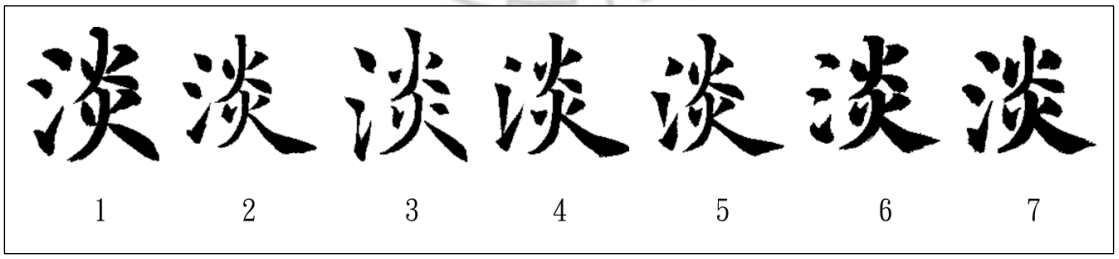


圖 20: 陳忠建書寫的 7 種字體

如圖 20 所示，展示了 7 種字體風格。如表 6 所示，本文使用了每種字體資料集，由於 7 種字體個數不一，並將 7 種共同字體取出來當作訓練資料。

表 6: 訓練資料數

風格	1	2	3	4	5	6	7	Total
Train	4500	4500	4500	4500	4500	4500	4500	31500
Test	500	500	500	500	500	500	500	3500
Total	5000	5000	5000	5000	5000	5000	5000	35000

資料庫中的字體寬度與長度式不一致的，長邊是固定 140，本文將短邊部分是用白底背景補成 140 x 140，再放大圖像至 256 x 256 圖像最為真實圖像，而圖像顏色深度為 1，因此不做任何改動．本文也使用了 simfang 字體做為輸入圖像 x，如前面圖 13 所示。

表 7: 實驗環境表

CPU	GPU	Ram	Python	Pytorch	CUDA	Cudnn
Intel i9-9900k	GTX 2080Ti	64GB	3.7	1.5.1	10.2	8.0.3

4.1 實驗環境

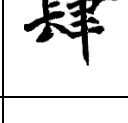
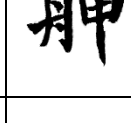
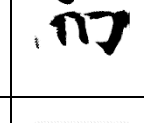
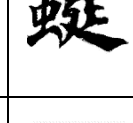
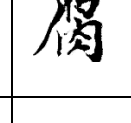
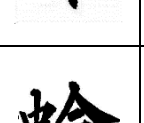
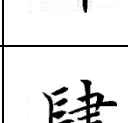
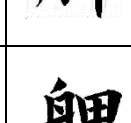
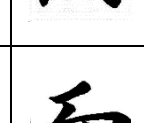
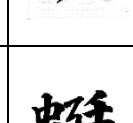
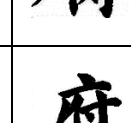
本文進行實驗所使用的硬體設備，如表 7 所示，顯示卡為 GTX 2080Ti ，記憶體總共 64 GB，處理器使用 Intel i9-9900K CPU 3.00 GHz，

軟體方面使用 Python 3.7、Pytorch 1.5.1、CUDA 10.2、cudnn 8.0.3。資料總預訓練時間為 74 小時。

4.2 訓練資料大小比較

在本篇論文中，使用陳忠建書寫的 7 種字體作為訓練資料，其中訓練資料共 35000 張圖像，主要以三種訓練資料大小作為訓練集，分別為 7000、21000 和 35000，如表 8 所示，訓練集大小隊於實驗結果具有較大的相關性。

表 8: 訓練資料大小比較

風格 / 訓練集	1	2	3	4	5	6	7
7000							
21000							
35000							
Target							

4.3 實驗評分指標

為了評比論文的實驗結果，本論文使用結構相似性指數(SSIM)作為評分標準之一，來確定真實圖像與生成圖像之間的相似度。本文也使用了主觀評估作為評分標準，對大學生以及親戚朋友進行調查。

4.3.1 結構相似性指數評分

結構相似性指數(Structural Similarity Index, SSIM) [23]是一種衡量兩幅圖像相似度的指標。該指標首先由德州大學奧斯丁分校的圖像和視頻工程實驗室（Laboratory for Image and Video Engineering）提出。SSIM 使用的兩張圖像中，一張爲未經壓縮的無失真圖像，另一張爲失真後的圖像。

對 SSIM 算法，其輸入是兩張圖片，即要求結構相似性的兩張圖片。其中一張是未經壓縮的無失真圖像（即 ground truth），另一張就是需要與無失真圖片對比的圖片。物體表面的亮度信息與照度和反射係數有關，且場景中的物體的結構與照度是獨立的，反射係數與物體有關。我們可以通過分離照度對物體的影響來探索一張圖像中的結構信息。這裏，把與物體結構相關的亮度和對比度作爲圖像中結構信息的定義。因爲一個場景中的亮度和對比度總是在變化的，所以我們可以通過分別對局部的處理來得到更精確的結果。定義如下：

$$SSIM(x,y)=[l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{r} \tag{12}$$

$$l(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2\mu_y^2+C_2} \tag{13}$$

$$\mathrm{c}(x,y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2\sigma_y^2+C_2} \tag{14}$$

$$\mathrm{s}(x,y)=\frac{2\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3} \tag{15}$$

其中，$l(x,y)$比較 x 和 y 的亮度(luminance)，$c(x,y)$比較 x 和 y 的對比度(contrast)，$s(x,y)$比較 x 和 y 的結構(structure)，$\alpha>0$、$\beta>0$、$\gamma>0$，為調整 $l(x,y)$、$c(x,y)$、$s(x,y)$相對重要性的參數，μ_x 及 μ_y、σ_x 及 σ_y 分別為 x 和 y 的平均值和標準差，σ_{xy} 為 x 和 y 的共變異數，C_1、C_2、C_3 皆為常數，用以維持 $l(x,y)$、$c(x,y)$、$s(x,y)$的穩定。結構相似性指標的值越大，代表兩個影像的相似性越高。

本文從柳公權字體實驗結果圖像和真實圖像中隨機選取 8 個圖像字體，如圖 21 所示，並與其他模型 Zi2Zi、Pix2Pix、HAN 進行 SSIM 評分，如表 9 所示，在此評分中，本論文的 SSIM 評分些許的較其他模型高分。

圖 21: 各實驗結果比較

表 9: SSIM 比較結果數據

評分	MAX	MIN	平均
Pix2Pix	0.4739	0.3516	0.4137
Zi2Zi	0.6549	0.5262	0.6031
HAN	0.6028	0.5399	0.5828
Proposed	0.6631	0.6157	0.6459

4.3.2 主觀評分

本文受試者為 18-22 歲之間的大學生 60 位，以及 27-60 之間的親戚朋友 40 位，從本次實驗生成圖像和真實圖像中隨機選取 8 個字體圖像，並與其他模型 Zi2Zi、Pix2Pix、HAN 中的生成圖像，一同給受試者作評分，如表 10 所示，評分標準為 1-10 分，給字體圖像做相似度的評分。

由這次的主觀評分分數來看，較大多數的人對於本論文實驗結果相似度評分較高，這也表示本論文的方法是有用處的。

表 10: 主觀比較結果數據

評分	MAX	MIN	平均
Pix2Pix	7	2	4.7
Zi2Zi	8	6	7.1
HAN	8	4	6.3
Proposed	9	6	8.7

4.4 實驗結果與 CalliGAN 比較

CalliGAN 是 2020 年所提出的，在臺灣算是第一篇以書法的生成系統作為論文發表，因此其中的實驗結果也具有相當高的質量，在此論文中，雖然有提供相關程式碼，但仍然有部分的程式碼尚未提供，因此未能將以實作。由於此論文實驗結果具有相當高的質量，因此想藉由此論文實驗結果與本論文作為比較。由圖 22 為 CalliGAN 論文中與 AEGG [16]做比較所提供的實驗結果。

圖 22: 與 AEGG、CalliGAN 比較結果圖

第五章　結論與未來展望

本論文研究以基於 U-Net 的生成器套用了 HAN 中的分層解碼器，以及套用了 Zi2Zi 中的類別嵌入所組成的，該方法有效提升生成圖像的品質，再由人體評分與 SSIM 評分表示，本論文生成圖像與真實圖像具有較高的相似度。

在研究當中，也有許多問題需要解決，如在書法家裡，有許多書法字體具有較大的差異性，如在字體的每一個點與線以及每一個比畫的長度與寬度，都跟輸入的字體具有較大的差異，導致訓練效果不佳，9 大大降低生成字體圖像的相似度，期望在未來的研究路途中，可以研究出能夠解決差異性較大的字體，也期望自己能夠在未來在這方面有所突破。

參考文獻

[1] Augustus Odena, Christopher Olah, and Jonathon Shlens. “Conditional image synthesis with auxiliary classifier GANs.” In ICML, 2017.

[2] A. Radford, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks.” Computer Science, 2015.

[3] Bart van Merrienboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk and Yoshua Bengio. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.” Computation and Language, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions.” In Computer Vision and Pattern Recognition, pages 1–9, 2015.

[5] Danyang Sun, Tongzheng Ren, Chongxun Li, Hang Su, and Jun Zhu. “Learning to write stylized Chinese characters by reading a handful of examples.” In IJCAI, 2018.

[6] F. A. Gers., J. Schmidhuber. and F. Cummins., “Learning to forget: Continual prediction with LSTM.” *9th International Conference on Artificial Neural Networks,* Institution of Engineering and Technology, vol.5, pp. 850–855, 1999.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio(2014). “Generative Adversarial Networks.” stat.ML. arXiv:1406.2661v1.

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of Wasserstein GANs.” In NeurIPS, 2017.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired image-to-image translation using cycle consistent adversarial networks.” In ICCV, 2017.

[10] Jie Chang, Yujun Gu, Ya Zhang, and Yan-Feng Wang. “Chinese handwriting imitation with hierarchical generative adversarial network.” In BMVC, 2018.

[11] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein GAN.” arXiv preprint arXiv:1701.07875, 2017.

[12] M. Mirza and S. Osindero. “Conditional generative adversarial nets.” Computer Science, pages 2672–2680,2014.

[13] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen and Nikhil Thorat. “Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation.” Computation and Language,2017.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “UNet: Convolutional networks for biomedical image segmentation.” In MICCAI, 2015.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks.” In CVPR, 2017.

[16] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. “Auto-encoder guided GAN for Chinese calligraphy synthesis.” In ICDAR, 2017.

[17] Shan-Jean Wu, Chih-Yuan Yang and Jane Yung-jen Hsu. “CalliGAN: Style and structure-aware Chinese calligraphy character generator.” In CVPR, 2020.

[18] Yuchen Tian. “zi2zi: Master Chinese calligraphy with conditional adversarial networks.” https://kaonashi-tyc. github.io/2017/04/06/zi2zi.html, 2017.

[19] Yaniv Taigman, Adam Polyak, and Lior Wolf. “Unsupervised cross-domain image generation.” arXiv preprint arXiv:1611.02200, 2016.

[20] Yexun Zhang, Ya Zhang, and Wenbin Cai. “Separating style and content for generalized style transfer.” In CVPR, 2018.

[21] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. “DCFont: an end-to-end deep Chinese font generation system.” In SIGGRAPH Asia. 2017.

[22] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. “StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation.” In CVPR, 2018.

[23] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. “Image quality assessment: from error visibility to structural similarity.” TIP, 13(4):600–612, 2004.

附錄一 英文論文

Calligraphy Character Generation System Based on the Generative Adversarial Networks Model

Yu-Jui Lien (連禹睿), Wen-Bing, Horng (洪文斌)

Department of Computer Science and Information Engineering

Tamkang University

Taipei, Taiwan

E-mail: horng@mail.tku.edu.tw, 608420062@ gms.tku.edu.tw

Abstract:Calligraphy is the artistic expression of Chinese character writing, with rich shapes and variations. This research attempts to use the Generative Adversarial Network to generate the calligraphy characters of famous calligraphers. We use calligraphy font images of famous calligraphy masters as training materials for learning. In this research, based on the Zi2Zi method, we implemented a generational confrontation network model for calligraphic font style conversion, and explored the impact of some other research methods on the model, and tried to arrive at an optimized deep learning model. We discuss methods such as adding U-Net, Category Embedding, and HAN, as well as the impact of the size of the training data set on the model. In the final experiment, we compared the pix2pix model, Zi2Zi model, HAN model, and HAN-Zi2Zi model. After experimentation, it is found that adding U-Net and Category Embedding are helpful to the results of the model, and the more fonts are used for training, the better the results will be. In addition, HAN-Zi2Zi works best.

Keywords:**Deep Learning, GAN, ZI2ZI, HAN, Calligraphic**

1. Introduction

Due to the advancement of science and technology, artificial intelligence (AI) technology has risen again. In recent years, scientists have invested in the field of artificial intelligence research. In the past, computer hardware equipment

could not be effectively provided to researchers to realize theories. Therefore, in the past research, artificial intelligence has risen several times but finally fell to the bottom. In recent years, due to the huge improvement in hardware technology, people rely on the powerful performance of hardware, and artificial intelligence research has once again entered the peak period.

In recent years, Generative Adversarial Networks (GAN) has gradually attracted everyone's attention. Many researchers have begun to join in the research one after another, and develop many various models. It is already a comprehensive technology. And there are many familiar applications, the most well-known application is a variety of image style conversion App and the face change that caused many celebrities a while back. Technically speaking, GAN is indeed a very important part of the development of deep learning in recent years.

2. Related Work

In the past literature, there are many Chinese character generation methods. The method proposed in this paper is the problem of image conversion. The related technologies are discussed as follows.

2.1 Image-to-image translation

Image-to-image translation is a kind of visual problem, which is to learn the mapping function between input and output. There are many applications of using this kind of method model [15,8,11,9,22,2,21],Many methods are mostly based on the conditions of GAN [7].

2.1.1 generative adversarial networks

In 2014, Goodfellow et al. [7] proposed the concept of Generative Adversarial Network (GAN), which greatly reduced the amount of data required for deep learning. GAN mainly uses the generation network (Generator, G) and the discriminant network (Discriminator, D) to continuously compare, so that the generation network can learn the distribution of data. Taking pictures as an example, after the training is completed, the generation network can start from a random period. Realistic images are generated in the numbers, like two characters, one is a counterfeiter, who keeps making fake banknotes, the other is a connoisseur, who constantly takes fake banknotes from the counterfeiter, and judges whether it is true or not. The counterfeiter is based on The feedback of the connoisseur's judgment results has been continuously improved, and finally the fake banknotes become difficult to distinguish between true and false. This is the concept of GAN. The main functions of G and D: G is a generated network, which receives a random number, and generates an image through the random number. D is a judgment network to determine whether an image is real. Its input is a picture, and its output is Number, this number represents the probability of whether the image is real or not, the highest is 1, which means

100% is a real image, and the output is 0, which means it is not a real image at all.

Under the GAN architecture, G is called a generative model, and D is called a discriminative model. The simple architecture is shown in Figure 1.

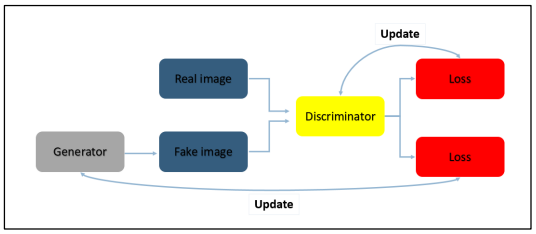


Figure 1. GAN architecture

Next, the objective function is introduced. As shown in the formula, G represents the generator, D represents the discriminator, y is the real data, Pdata is the distribution of the real data, and x is the random input data. From the perspective of the discriminator, in order to distinguish the real The data y and the fake image G(x), the discriminator D hopes that the larger the judgment from the original data Pdata, the better, and the smaller D(G(x)), the better, that is to say, the more V(D,G) The bigger the better, and from the generator's point of view, the generator hopes that the generated fake image G(x) can deceive the discriminator D, that is, D(G(x)) can be as large as possible, the better, That is to say, V(D,G) is as small as possible, the better, GAN is to fight each other in these two models, and finally achieve the best effect.

$$\min_G \max_D V(G,D) = E_{y\sim Pdata(y)}[\log D(y)] + E_{x\sim Pdata(x)}[\log D(1-D(G(x)))] \quad (1)$$

2.1.2 Pix2Pix

Pix2Pix is the first method to do image conversion. In 2017, Phillip Isola et al. [15] proposed a research based on generating and fighting network image conversion. It is a kind of GAN network, mainly adopts the structure of CGAN [12] network, it still includes a generator and a discriminator. The generator uses a U-net [14] structure, which is somewhat similar to Encoder-decoder [3]. It contains a total of 15 layers, with 8 layers of convolutional layers as encoders and 7-layer deconvolutional layers as decoders, which are different from the traditional The difference of the encoder-decoder is that it introduces a technique called "skip-connect", that is, the input of each deconvolution layer is: the output of the previous layer plus the output of the convolutional layer symmetrical to that layer, This ensures that the information of the encoder can be continuously re-memorized during the decoder, so that the generated image retains some information of the original image as much as possible.

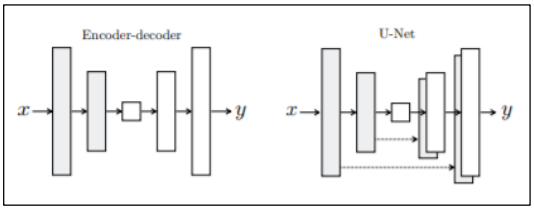


Figure 2. U-Net architecture[15]

In Figure 2, U-Net [14] is also an Encoder-Decoder model. Encoder and Decoder are symmetrical. The so-called U-Net splices the i-th layer to the n-i-th layer. This is done because the image sizes of the i-th and n-i-th layers are the

same, and it can be considered that they carry similar information. The architecture is shown in Figure 3, which is the architecture diagram of Pix2pix.

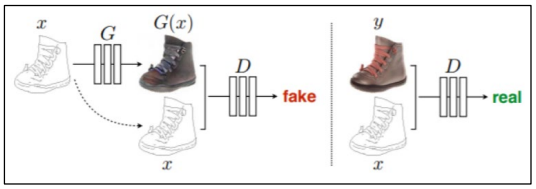


Figure 3. Pix2pix architecture[15]

The Pix2pix[15] discriminant network adopts the PatchGAN structure, which means that the image is equally divided into multiple fixed-size Patches, and the authenticity of each Patch is judged separately, and finally the average value is taken as the final output of D, as shown in Figure 4. . L1-Loss is also used to reduce the difference between the output image and the training image. The limitation of Pix2Pix [15] is that it requires paired images to train the model, which is difficult to apply to object conversion.

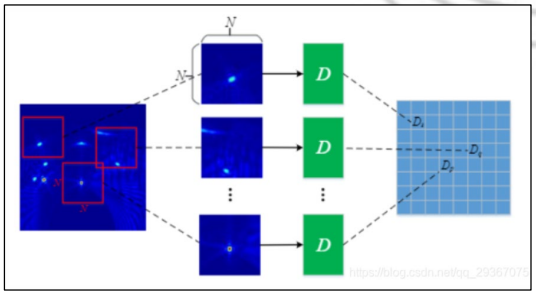


Figure 4. PatchGAN graphical representation [15]

The L1-Loss used by Pix2pix on the generation network is shown in the following function:

$$\mathcal{L}_{L1}(\mathrm{G}) = \mathbb{E}_{x,y,z}[\|y - \mathrm{G}(x,z)\|_1] \quad (2)$$

And the integrated cGAN architecture function is as follows:

$$\mathrm{G}^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(\mathrm{G},\mathrm{D}) + \lambda \mathcal{L}_{L1}(\mathrm{G}) \quad (3)$$

Where G tries to minimize the target, and the discriminator D tries to maximize the target.

2.1.3 CycleGAN

In 2017, Zhu Park and Efros [9] proposed the research of loop generation confrontation network, which is not only used by researchers in computer graphics and other fields, but also has become a tool widely used by visual artists. Pix2Pix [15] can generate images similar to the training style, but cannot specify a specific image. CycleGAN can effectively solve this problem. CycleGAN has a major development. In the past, data usually requires paired influence, and general visual problems It is difficult to find pairs of high-quality images to provide model learning. Generally, the generation of confrontation models relies on paired images. Unless paired effects are purposefully produced, they cannot be trained, and it is easy to cause deviations in the generated data. Generally paired And the description of unpaired images, as shown in Figure 5.

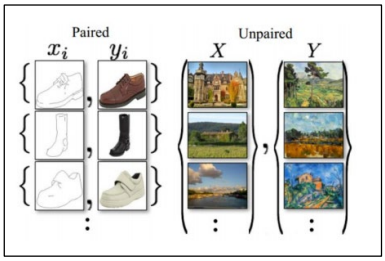


Figure 5. CycleGAN concept[9]

The principle diagram of CycleGAN, as shown in Figure 6, input a real image A, through the generator G_AB to generate a fake image B that you want to convert into a B image, and then the discriminator D_B determines whether it is similar to the real B image, in order to avoid generating The output image over-learns the target and the teacher learns the characteristics of the original real image A. A generator is needed to generate a fake image A that is similar to the real image. The fake image A is more similar to the input real image A. The better, it means that the characteristics of the original image are retained.

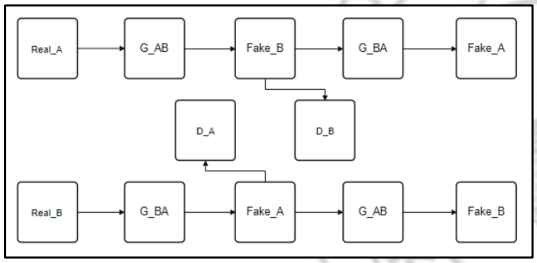


Figure 6. CycleGAN architecture[9]

The CycleGAN loss function is mainly divided into two parts, namely the confrontation loss and the cycle consistency loss function. The following two inputs must be defined as X and Y, the two discriminators are Dx and Dy, and the two generators are G and F.

$$L_{GAN}(G,D_y,X,Y)= E_{y\sim P_{data}(y)}[\log D_y(y)] + E_{x\sim P_{data}(x)}[\log(1-D_y(G(x)))] \quad (4)$$

The generator G will generate G(x) as close as possible to the data taken from Y, while Dy will try to distinguish G(x) from the real distributed data y. During the training process, the generator G's The goal is to minimize this loss function, Dy wants to maximize this loss function on the contrary, the same principle pattern is also applied to generator F.

The final loss function combines the confrontation loss function of generator G and generator F and the cycle consistency loss function as follows:

$$L_{cyc}(G,F)=E[\|F(G(x))-x\|_1] + E[\|G(F(y))-y\|_1] \quad (5)$$

2.2 Zi2Zi Chinese character generation

Zi2Zi [18] is the first method to use GAN to generate Chinese characters. The method proposed by Github user Kaonashi-tyc in 2017, the Pix2Pix [15] model does not solve this one-to-many relationship. Inspired by the "Zero-shot GNMT" paper (Zero-shot GNMT) [13], the author came up with a category embedding method, using untrainable Gaussian noise as style embedding and Chinese character embedding (character embedding) are connected in series, and then enter the decoder together. In this way, the decoder still maps the same Chinese character to the same vector, but the decoder will also consider the two embeddings of the Chinese character and the style to generate the target character (target character). As shown in Figure7.

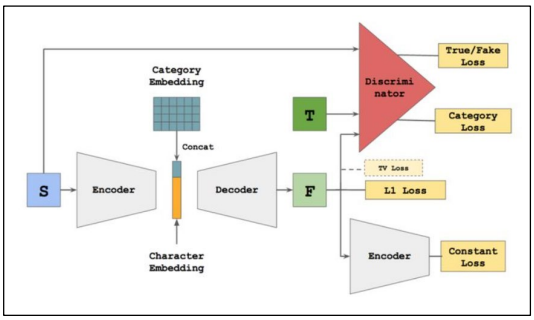


Figure 7. Zi2Zi architecture[18]

However, the author found that after doing so, a new problem appeared: the model began to confuse and mix various styles, and the generated Chinese characters were nothing like. Therefore, the multi-class category loss in the AC-GAN [1] model is adopted, and this loss is added to the discriminator. Once confusion or style mixing occurs, the discriminator is "punished". The constant loss in DTN [19] is also used to improve the output quality.

2.3 AEGG Chinese character generation

Zi2Zi is an open park project and has not published a paper. The first paper using GAN to generate Chinese characters is AEGG. In 2017, Pengyuan Lyu et al. [16] proposed a research on the automatic encoder for Chinese calligraphy font generation, which is also based on Pix2Pix [15] , Encoder-Decoder is added, which supervises and provides information during the training process. Unlike Zi2Zi [18], AEGG can only be trained in a single manner, as shown in Figure 8.

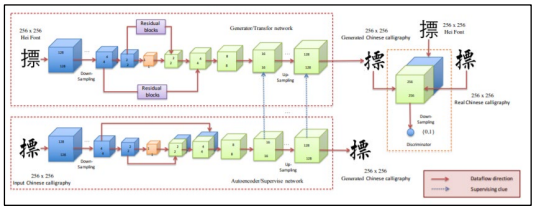


Figure 8. AEGG architecture[16]

2.4 HAN Chinese character generation

Another paper that uses GAN to generate Chinese characters is HAN. In 2017, Jie Chang et al. [10] proposed Chinese font conversion based on hierarchical generation against the Internet. Also based on Pix2Pix [15], for font conversion tasks, different fonts may share the same radicals. This is a good feature that the font conversion method can take advantage of, that is, learn the direct mapping between the source style and the target style. However, sometimes in a particular font, the same radical may appear completely different in different characters, as shown in Figure 9.

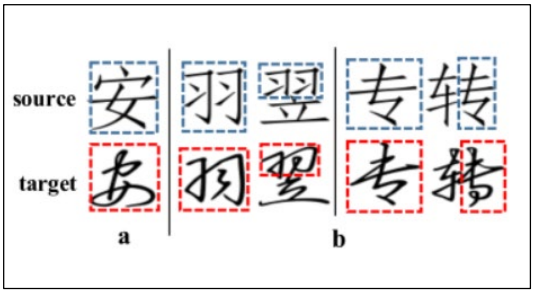


Figure 9. Radical comparison[10]

With the idea of GoogLeNet [4], a layered decoder is proposed, which generates artificial images in multiple decoding layers, which is expected to help the decoder learn better representations in its hidden layers. The hierarchical decoder tries to preserve the global topological structure in different decoding layers as much as possible, while taking into account the local characteristics of the decoding in the hidden layer, so that the transmission network can generate close to real font images. The architecture is shown in Figure 10 below.

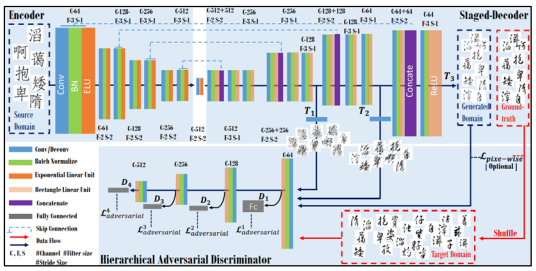


Figure 10. HAN architecture[10]

2.5 CalliGAN Chinese character generation

In 2020, a paper published by National Taiwan University also uses GAN to generate Chinese characters. In 2020, Shan-Jean Wu et al. [17] proposed a Chinese calligraphy generator for style and structure, based on Pix2Pix [15], which is also used immediately U-Net [14] is used as the architecture of the generator, considering that Chinese characters are composed of many strokes, which is different from learning the translation mode between a given language, EMD [20] and SA-VAE [5] are both independent The content and style of the two are regarded as two unrelated domains, and two independent encoders are used to model them. However, their technical details are different. EMD [20] mixes the potential features of style and content in a bilinear hybrid network, and generates an output image through an image decoder. Therefore, its training samples are very special. A sample includes two sets of training images, one for content and the other for style. In contrast, SA-VAE [5] uses a sequential approach. It first recognizes the selection of characters from a given image, and then encodes the recognized characters into a special code. SA-VAE [5] shows that the main knowledge of Chinese characters helps to improve the quality of output images. Therefore, the Chinese characters are decomposed into 517 parts, and the coding configuration is performed, and the features of each part are extracted using LSTM [6] as the generation

Condition for training, the architecture is shown in Figure 11.

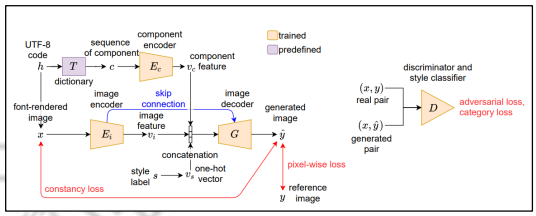


Figure 11. CalliGAN architecture[17]

Given an h, by rendering the image x, the feature vector vi is generated by the image encoder Ei, and the component sequence c is also obtained by the dictionary T. The component features are generated by the encoder Ec, and finally vc, vi and The vector vs converted by the style label S is passed through the decoder G to generate the image y, and the comprehensive formula is as follows:

$$\mathcal{L} = \log \mathrm{D}(x, y) + \log(1\text{-}\mathrm{D}(x, \hat{y})) + \lambda_p L_p + \lambda_c L_c + \lambda_s L_s \quad (6)$$

3. Proposed Method

In this paper, we apply the data set used in CalliGAN [17], from the Chen Zhongjian calligraphy font provided on the Internet, and based on Pix2Pix [15], which incorporates the layered decoder of HAN [10] In addition, it also combines the category entry in Zi2Zi [18] to

improve the quality of training. Figure 12 is the introduction chapter of the research method of this thesis.

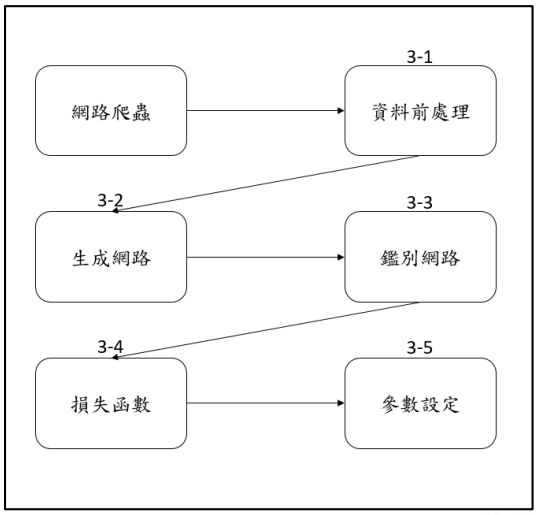


Figure 12. Experiment introduction chapter

First of all, the database used in CalliGAN will be used as the database of this thesis. The 7 fonts in the calligraphy database of Chen Zhongjian are used as the training materials of this thesis, as shown in Figure 13.

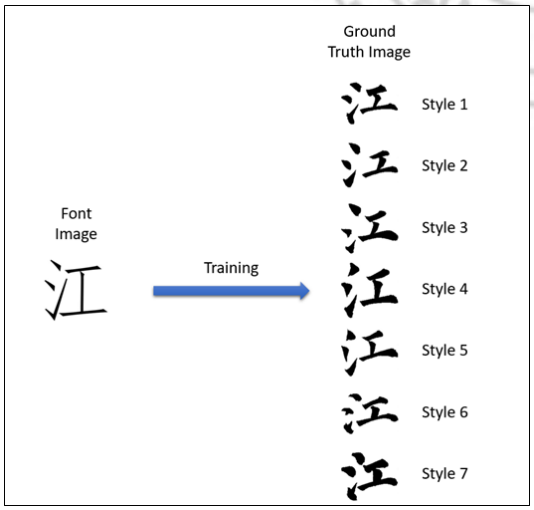


Figure 13. Source font (simfang) and 7 fonts

As shown in Figure 14, the 7 fonts in Chen Zhongjian's calligraphy database are captured using a web crawler, and the data is pre-processed to convert the image to 256 x 256. The number of words in the 7 fonts is not the same, so It is necessary to find the fonts that are jointly owned.

Convert the calligraphy font format TTF file to the image format JPG file, merge the input font and target font into the corresponding image 256 x 512, and then convert the format to the training format OBJ file, the training data and test data account for 9:1, And enter the data into each model for training.

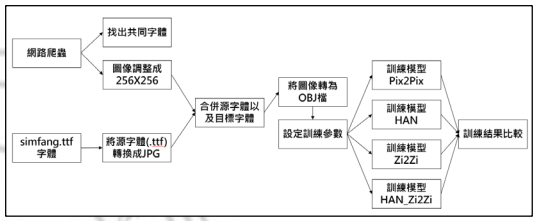


Figure 14. Experimental flowchart

In 3-1, the pre-processing methods of the database of this paper will be introduced, as well as the proportion of training and testing. After that, the method and architecture of generating the network will be introduced in 3-2. Then, in 3-3, we will introduce the method and architecture of the confrontation network, as well as the related loss function. The related loss function will be introduced in 3-4. Finally, the relevant parameters used in this paper will be introduced in 3-5.

3.1 Data preprocessing

This paper uses Chen Zhongjian's calligraphy font. Each of these calligraphy images has a different size. The images in this database have a fixed constant edge of 140, and they are all black characters on a white background.

Images can have the same fixed length and width, so a 140X140 white background image with the same length and width is added behind each image, as shown in Figure 15.

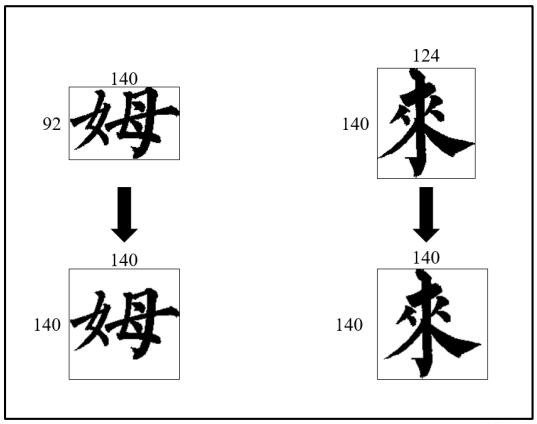


Figure 15. Before image processing

Then enlarge the image to 256x256 in equal proportions. Among the 7 fonts, the number of each font is inconsistent, and some parts of the font are repeated. In order to unify the training ratio and comparison, this article will share the common fonts in each font Grab and take out, a total of 5000 words, as shown in Table 1.

Table 1. The number of training data

	Style 1	Style 2	Style 3	Style 4	Style 5	Style 6	Style 7
原資料	6171	7008	7159	6901	6999	6308	7006
共有字數	5000	5000	5000	5000	5000	5000	5000

The method in this article is to train the target image and the source image as a pair, so the source font part should also be output as an image, as shown in Figure 16, with the source font on the left and the target font on the right, which are merged into training data.

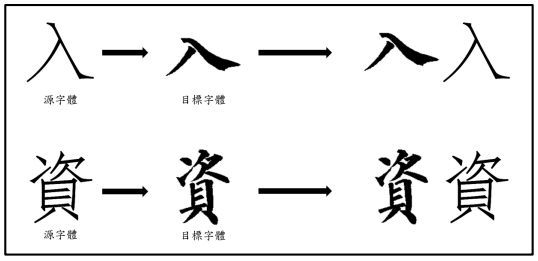


Figure 16. Training data merge

3.2 Generate network

In this paper, the network structure generated in this paper is mainly based on U-Net. Because the relative position feature information of any image is very important to the man, U-Net is used to merge the feature information of the image. Mainly reduce the size and keep important information. However, in some cases, some feature information will be lost, so feature images are additionally selected for training in the generation network.

And herein decoder, the main basis of the method using the HAN, using the feature information has become wild intermediate image decoder, and generates a final image, transmitted together discriminator. However, this article only measures the difference between the final generated image and the target image. At the same time, the loss of the feature image generated in the middle can help optimize the generation network, and can normalize the parameters in the generation network, and can also solve the problem of overfitting to a certain extent.

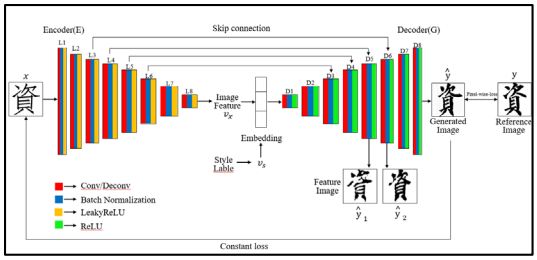


Figure 17. Generate network process

As shown in Figure 17 above, the input x is an image of 256x256, which is input to the encoder (Encoder, E). This article uses 8 convolutional layers, of which LeakyReLU and BatchNormalize are used in each layer of convolution, and finally the feature vector is obtained. , The negative slope of LeakyReLU is 0.2. In order to generate 7 styles of fonts, this article uses Zi2Zi's Category Embedding to classify font styles, which are sent to the decoder together with the feature vector. In the decoder, 8 layers of convolutional layers are used. ReLU and BatchNormalize are added to each layer of convolution. Finally, the generated image y is output. The slope of ReLU is 0.2. Each generated image y will match the target The image is pixe-wise, and then referring to the concept of the DTN [19] network, adding constant loss, the input image and the generated image are very similar, so the position in the embedding space is also very similar , So the constant loss forces the decoder to retain the content of the generated Chinese characters, speeding up the convergence several times. In the up-sampling of the decoder, skip connections are made in L3, L4, L5, and L6 to merge the reserved features, and the features in the D5 and D6 layers are output as an image, and the final generated image is sent to the discriminator.

Table 2. Encoder detailed architecture

Layer	Feature Size	Number of Filters
Input	256x256	1
Conv2d_1	128x128	64
Conv2d_2	64x64	128
Conv2d_3	32x32	256
Conv2d_4	16x16	512
Conv2d_5	8x8	512
Conv2d_6	4x4	512
Conv2d_7	2x2	512
Conv2d_8	1x1	512

Table 3. Encoder detailed architecture

Layer	Feature Size	Number of Filters
DeConv_1	2x2	512
DeConv_2	4x4	512
Conv2d_6 +DeConv_ 2	4x4	1024
DeConv_3	8x8	512
Conv2d_5 +DeConv_ 3	8x8	1024
DeConv_4	16x16	512

Conv2d_4 +DeConv_ 4	16x16	1024
DeConv_5	32x32	256
Conv2d_3 +DeConv_ 5	32x32	512
DeConv_6	64x64	128
DeConv_7	128x128	64
DeConv_8	256x256	1

The architecture of Encoder and Decoder will be as shown in Table 2 and Table 3. In the Encoder part, it will first go through a Convolution layer, then through the Batch normalization step, and finally through the LeakyReLU layer, and in the Decoder part, it will go through a Deconvolution layer first, Then go through Batch normalization, and finally go through the ReLU layer.

3.3 Discrimination network

In this paper, against the network in order to solve the problem confuse embedded into the category of lead, also made reference to AC-GAN Mutli-class category loss [1] is added in the discriminator, once confused or mixed up styles, to punish Discriminator.

Since the feature images of the 5th and 6th layers are also generated in the generated confrontation in this paper as the confrontation target, there are also two sets of confrontation data in the confrontation network, as shown in Figure 18.

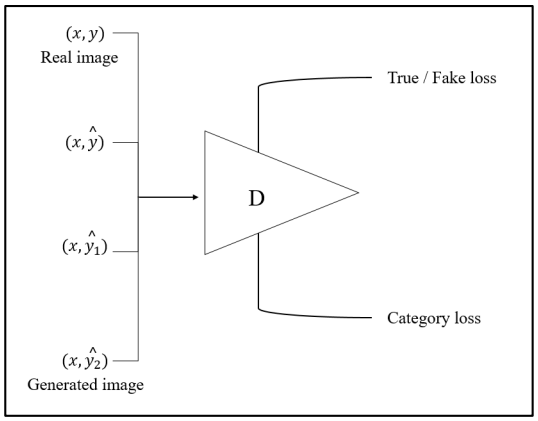


Figure 18. Discriminator network process

However, in order to be able to generate a clearer part of the image, this article uses the structure of PatchGAN, as shown in Figure 19, which divides the image into Patches, and judges the authenticity of each Patch separately, so that a clearer can be generated image.

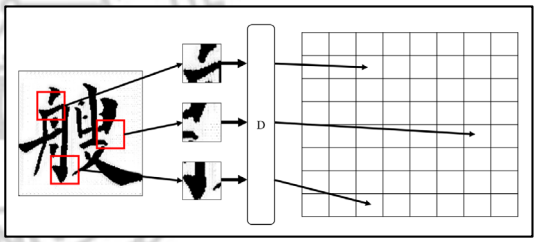


Figure 19. PatchGAN architecture

Discriminator's architecture will be as shown in Table 4. It will first go through a Convolution layer, then through the Batch normalization step, and finally through the LeakyReLU layer. The negative slope of LeakyReLU is 0.2.

Table 4. Discrimination network architecture

Layer	Feature Size	Number of Filters
Input	256x256	1

Conv2d_1	128x128	64
Conv2d_2	64x64	128
Conv2d_3	32x32	256
Conv2d_4	30x30	512
Conv2d_5	30x30	1

3.4 Loss function

This paper studies the design of the loss function, including 4 loss functions to train the model. First, for the confrontation network, the conditional loss function (Conditional GAN, cGAN) is used.

$$\mathcal{L}_{cGAN}=[\log \mathrm{D}(\mathrm{x},\mathrm{y})] + [\log (1\text{-}\mathrm{D}(\mathrm{x}, \hat{\mathrm{y}}\ \hat{\mathrm{y}}_1\ \hat{\mathrm{y}}_2))] \quad (7)$$

In the generation network, two more feature images are generated. In order to generate more effective feature information, the network can be used as a training target. Where x is the condition, y is the real image, z is the random noise input, and the result is the discriminator. The loss of the image pixel level is to generate the basic loss function in the confrontation network, and the purpose is to generate the image It looks like a pattern similar to the training image, so this article uses Pixed-wise Loss.

$$\mathcal{L}_{\mathrm{p}} = [\|y - G(x,z)\|_1] \quad (8)$$

In this paper, the input x and the output image y are the same words, so constant loss is used to promote the two images to have the same feature vector.

$$\mathcal{L}_{\mathrm{c}} = [\|\mathrm{E}_i(x) - \mathrm{E}_i(\mathrm{G}(x,z)\|_1] \quad (9)$$

Because in the generation network. Use the category Embedding to many target training, so there will be a problem of confusing images, in order to allow the generated image to retain the original allocation, the use of Category Loss.

$$\mathcal{L}_{\mathrm{s}} = [\log(D_s(S|y)) + \log(D_s(S|G(x,z)))] \quad (10)$$

Loss while adding the above four, and adding a convenient adjustment of the right weight k, of the present study is the Loss Function.

$$\mathcal{L}_{\text{total}} = [\mathcal{L}_{\text{cGAN}} + \lambda_{\mathrm{p}}\mathcal{L}_{\mathrm{p}} + \lambda_{\mathrm{c}}\mathcal{L}_{\mathrm{c}} + \lambda_{\mathrm{s}}\mathcal{L}_{\mathrm{s}}] \quad (11)$$

3.5 Parameter setting

The Optimizer used in this study is the Adam optimizer. Generally speaking, the learning rate is very important for the optimizer. If it is too small, it will take too much time to learn, and if it is too large, there is a possibility of overfitting. As shown in Table 5, this paper uses a learning rate of 0.0002, an Epoch of 200, a Batch size of 32, the negative slope of LeakyReLU is 0.2, and the slope of ReLU is 0.2.

Table 5. Parameter setting

訓練參數名稱	參數值
epoch	200
batch size	32

LeakyReLU,ReLU 的斜率	0.2
Optimizer	Adam
learning rate	0.0002

4. Experimental Results

For model training, the training materials of this article are downloaded from Chen Zhongjian's calligraphy database. All the images are copied from the calligraphy fonts of famous calligraphers in ancient times. The font database covers 29 fonts. This article uses the following 7 font styles:

1. Liu Gongquan,
2. Yu Shinan,
3. Chu Suiliang,
4. Ouyang Xun–Inscription on Sweet Wine Spring at Jiucheng Palace,
5. Ouyang Xun–Huangfu Dan Stele,
6. Yan Zhenqing–Stele of the Abundant Treasure Pagoda,
7. Yan Zhenqing–Yan Qinli Stele.

Figure 20. 7 fonts written by Chen Zhongjian

As shown in Figure 20, 7 font styles are displayed. As shown in Table 6, each font data set is used in this article. Since the number of 7 fonts is different, the 7 common fonts are taken out as training data.

The font width in the database is inconsistent with the length style. The long side is fixed at 140. This article uses a white background to fill the short side to 140 x 140, and then enlarge the image to 256 x 256, which is the most real image. The image color depth is 1, so no changes are made. This article also uses simfang font as the input image x, as shown in Figure 13 above.

Table 6. Number of training data

風格	1	2	3	4	5	6	7	Total
Train	4500	4500	4500	4500	4500	4500	4500	31500
Test	500	500	500	500	500	500	500	3500
Total	5000	5000	5000	5000	5000	5000	5000	35000

4.1 Experimental Results

The hardware equipment used in this experiment is shown in Table 7. The graphics card is GTX 2080Ti, the total memory is 64 GB, the processor uses Intel i9-9900K CPU 3.00 GHz, and the software uses Python 3.7, Pytorch 1.5.1, CUDA 10.2, cudnn 8.0.3. The total pre-training time of the data is 74 hours.

Table 7. Experimental environment table

CPU	GPU	Ram	Python	Pytorch	CUDA	Cudnn
Intel i9-9900k	GTX 2080Ti	64GB	3.7	1.5.1	10.2	8.0.3

4.2 Compare the size of the training data

In this paper, 7 types of fonts written by Chen Zhongjian are used as training data. Among them, there are 35,000 images of training data. Three training data sizes

are mainly used as training sets, which are 7000, 21000 and 35000 respectively, as shown in Table 8. , The training set size has a greater correlation with the experimental results.

Table 8. Compare the size of the training data

風格 / 訓練集	1	2	3	4	5	6	7
7000							
21000							
35000							
Target							

4.3 Experiment scoring index

In order to compare the experimental results of the paper, this paper uses the Structural Similarity Index (SSIM) as one of the scoring criteria to determine the similarity between the real image and the generated image. This article also uses subjective evaluation as a scoring standard to survey college students and relatives and friends.

4.3.1 Structural Similarity Index

The Structural Similarity Index (SSIM) [23] is an index that measures the similarity of two images. This indicator was first proposed by the Laboratory for Image and Video Engineering of the University of Texas at Austin. Among the two images used by SSIM, one is an uncompressed, undistorted image, and the other is a distorted image.

For the SSIM algorithm, the input is two pictures, that is, two pictures that require structural similarity. One is an uncompressed, undistorted image (that is, ground truth), and the other is an image that needs to be compared with an undistorted image. The brightness information of the surface of the object is related to the illuminance and reflection coefficient, and the structure of the object in the scene is independent of the illuminance, and the reflection coefficient is related to the object. We can explore the structural information in

an image by separating the influence of illuminance on the object. Here, the brightness and contrast related to the structure of the object are used as the definition of the structure information in the image. Because the brightness and contrast in a scene are always changing, we can get more accurate results by processing the parts separately. It is defined as follows:

$$SSIM(x,y)= [l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{r} \quad (12)$$

$$l(x,y) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2\mu_y^2+C_2} \quad (13)$$

$$c(x,y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2\sigma_y^2+C_2} \quad (14)$$

$$s(x,y) = \frac{2\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3} \quad (15)$$

Among them, l(x, y) compares the brightness of x and y, c(x, y) compares the contrast of x and y, s(x, y) compares the structure of x and y, α>0, β>0, γ>0, are the parameters to adjust the relative importance of l(x, y), c(x, y), s(x, y), μx and μy, σx and σy are x respectively And y mean and standard deviation, σxy is the covariance of x and y, C1, C2, and C3 are all constants to maintain l(x, y), c(x, y), s(x, y))'S stability. The larger the value of the structural similarity index, the higher the similarity between the two images.

This article randomly selects 8 image fonts from the experimental result image of Liu Gongquan font and the real image, as shown in Figure 21, and performs SSIM scoring with other models Zi2Zi, Pix2Pix, and HAN, as shown in Table 9, in this scoring , The SSIM score of this paper is slightly higher than other models.

Figure 21. Comparison of experimental results

Table 9. SSIM comparison results

評分	MAX	MIN	平均
Pix2Pix	0.4739	0.3516	0.4137
Zi2Zi	0.6549	0.5262	0.6031
HAN	0.6028	0.5399	0.5828
Proposed	0.6631	0.6157	0.6459

4.3.2 Subjective scoring

The subjects in this article are 60 college students between the ages of 18-22 and 40 relatives and friends between 27 and 60. 8 font images are randomly selected from the images generated in this experiment and the real images, and the The generated images in the other models Zi2Zi, Pix2Pix, and HAN were scored together with the subjects. As shown in Table 10, the scoring standard is 1-10

points, and the font images are scored for similarity.

Judging from the subjective scoring scores this time, more people have a higher score for the similarity of the experimental results of this paper, which also indicates that the method of this paper is useful.

Table 10. Subjective comparison result data

評分	MAX	MIN	平均
Pix2Pix	7	2	4.7
Zi2Zi	8	6	7.1
HAN	8	4	6.3
Proposed	9	6	8.7

4.4 The results compared with CalliGAN

CalliGAN was proposed in 2020. It is considered the first in Taiwan to publish a calligraphy generation system as a paper. Therefore, the experimental results are of very high quality. In this paper, although the relevant code is provided, there are still Part of the code has not yet been provided, so it cannot be implemented. Since the experimental results of this paper are of very high quality, I would like to compare the experimental results of this paper with this paper. Figure 22 shows the experimental results provided in the CalliGAN paper for comparison with AEGG [16].

Figure 22. Comparison result graph with AEGG and CalliGAN

5. Conclusions

In this thesis, the U-Net-based generator applies the hierarchical decoder in HAN and the category embedding in Zi2Zi. This method effectively improves the quality of the generated image, and then the human body score and SSIM score are used. Indicates that the generated image in this paper has a high degree of similarity with the real image.

In the research, there are also many problems that need to be solved. For example, in the calligraphy house, there are many calligraphy fonts with great differences. For example, the length and width of each point and line of the font and the length and width of each stroke are all the same as the input font. There are large differences, which leads to poor training effects. 9 greatly reduces the similarity of the generated font images. I hope that in the future research road, I can study fonts that can solve the large differences. I also hope that I can be in

the future. There has been a breakthrough in this area.

REFERENCE

[1] Augustus Odena, Christopher Olah, and Jonathon Shlens. “Conditional image synthesis with auxiliary classifier GANs.” In ICML, 2017.

[2] A. Radford, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks.” Computer Science, 2015.

[3] Bart van Merrienboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk and Yoshua Bengio. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.” Computation and Language, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions.” In Computer Vision and Pattern Recognition, pages 1–9, 2015.

[5] Danyang Sun, Tongzheng Ren, Chongxun Li, Hang Su, and Jun Zhu. “Learning to write stylized Chinese characters by reading a handful of examples.” In IJCAI, 2018.

[6] F. A. Gers., J. Schmidhuber. and F. Cummins., “Learning to forget: Continual prediction with LSTM.” *9th International Conference on Artificial Neural Networks,* Institution of Engineering and Technology, vol.5, pp. 850–855, 1999.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio(2014). “Generative Adversarial Networks.” stat.ML. arXiv:1406.2661v1.

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of Wasserstein GANs.” In NeurIPS, 2017.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired image-to-image translation using cycle consistent adversarial networks.” In ICCV, 2017.

[10] Jie Chang, Yujun Gu, Ya Zhang, and Yan-Feng Wang. “Chinese handwriting imitation with hierarchical generative adversarial network.” In BMVC, 2018.

[11] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein GAN.” arXiv preprint arXiv:1701.07875, 2017.

[12] M. Mirza and S. Osindero. “Conditional generative adversarial nets.” Computer Science, pages 2672–2680,2014.

[13] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen and Nikhil Thorat. “Google’s Multilingual Neural Machine

Translation System: Enabling Zero-Shot Translation.” Computation and Language,2017.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “UNet: Convolutional networks for biomedical image segmentation.” In MICCAI, 2015.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks.” In CVPR, 2017.

[16] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. “Auto-encoder guided GAN for Chinese calligraphy synthesis.” In ICDAR, 2017.

[17] Shan-Jean Wu, Chih-Yuan Yang and Jane Yung-jen Hsu. “CalliGAN: Style and structure-aware Chinese calligraphy character generator.” In CVPR, 2020.

[18] Yuchen Tian. “zi2zi: Master Chinese calligraphy with conditional adversarial networks.” https://kaonashi-tyc. github.io/2017/04/06/zi2zi.html, 2017.

[19] Yaniv Taigman, Adam Polyak, and Lior Wolf. “Unsupervised cross-domain image generation.” arXiv preprint arXiv:1611.02200, 2016.

[20] Yexun Zhang, Ya Zhang, and Wenbin Cai. “Separating style and content for generalized style transfer.” In CVPR, 2018.

[21] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. “DCFont: an end-to-end deep Chinese font generation system.” In SIGGRAPH Asia. 2017.

[22] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. “StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation.” In CVPR, 2018.

[23] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. “Image quality assessment: from error visibility to structural similarity.” TIP, 13(4):600–612, 2004.