中華民國第 58 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第一名

052511

運用 GAN 實現字體風格轉換

學校名稱：臺北市立建國高級中學

作者：	指導老師：
高二 邱泓翔	許雅淳

關鍵詞：GAN、pix2pix、cycleGAN

得獎感言

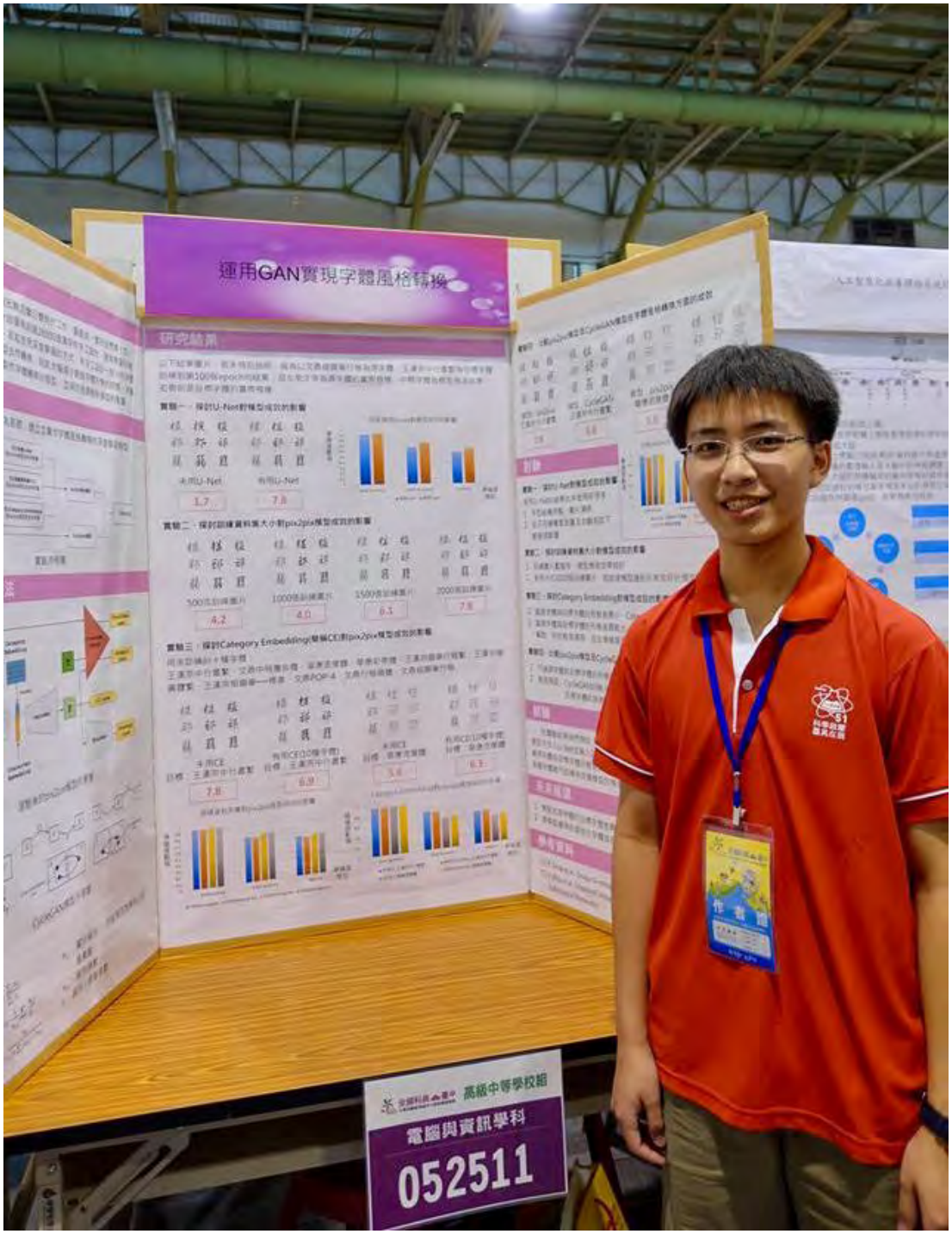
這是我第一次參加全國科展，我以前也有幾次參加科展的經驗，但都沒有晉級全國，因此我十分珍惜這次的機會，而我也確實在這趟旅程中交到一些同樣喜歡科學的朋友，得到了許多寶貴的經驗。

從升高二的暑假開始，我開始為了班上的成果發表進行個人的專題研究，還記得那些思考要訂甚麼題目、閱讀文獻資料的日子，雖然蠻辛苦的，但現在回想起來也自有一番趣味，而我也在看論文的過程中偶然發現了 GAN 的存在，因為覺得十分有趣，於是我將之用在字體風格轉換這個主題上，最終就有了現在的研究。

除了文獻探討外，實驗也是另一個我認為十分辛苦的部分，尤其我做的又是深度學習方面的研究，每跑一次實驗都需要花一天以上的時間，而在跑實驗之前的 debug 或一些繁瑣小事也總會讓我心煩不已。但現在回想這段過程，我想，努力也許不一定會有收穫，但不努力是一定不會成功的吧！正是這段辛苦的過程才會有現在的研究成果，更何況我也在這之中認識到許多人工智慧與深度學習的知識，也學會了相關程式的許多實作細節。

其實我並沒有想到自己會拿下電腦與資訊學科的第一名，在頒獎典禮聽到自己的名字時，心裡其實頗為震驚的。但我覺得除了得獎之外，認識了許多參展選手以及他們的研究才是我這次參賽的最大收穫，在評審第二天下午的公開展覽時，我去參觀了一些其他的參展作品，學會了蠻多新知識，也認識到許多新奇的想法，讓我不禁讚嘆科學的世界真的是廣大無邊、無窮無盡阿！

最後，我也要感謝在一路上幫助並鼓勵我的老師、家人及同學們，因為你們的支持，才讓我能夠度過在研究過程中一次次的困難，也才成就了我這次豐富精彩的科展旅程。



與展板合照

摘要

本研究以實作字體風格轉換的生成對抗網路模型為動機，將 Conditional GAN 當作模型的基礎，探討 pix2pix 模型及其他研究的一些方法對模型會產生甚麼影響，以得出能最優化預測成效的深度學習模型。

首先進行的是前處理的步驟，將字體的 truetype 檔案轉換成模型輸入的 jpeg 檔，再以生成器（Generator）和判別器（Discriminator）建立 Condional GAN 的基礎模型，然後探討加入 U-Net、Category Embedding 等方法，以及訓練資料集大小對模型造成的影響，最後實作整合的 pix2pix 模型和 CycleGAN 模型進行比較。

經過實驗後發現，U-Net 和 Category Embedding 都對模型的預測成果有所幫助，而使用越多字體進行訓練會有越好的成效。另外，對相似的字體而言，CycleGAN 的效果較好，而對兩種風格差異較大的字體則需用 Category Embedding 的方式，融入更多字體進行訓練以達到更好的成效。

壹、 研究動機

字體是每個人在文書處理上都會用到的工具，舉凡平常公文用的新細明體或標楷體，抑或簡報常使用的微軟正黑體，甚至是一些具有藝術性的字體，如行書體、碑體等等，都市我們在使版面整齊漂亮時經常使用的工具。

但我們所不知道的是，創造中文字體其實是一項困難且曠日費時的工作，根據統計，要創造一套符合標準（如：GBK）的中文字體，設計師要為超過 26000 個漢字作手工設計，通常需要好幾年的時間來完成。因此，若能使用深度學習的方式，先手工設計一部分的字，剩下的用深度學習的模型去作轉換，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（運用生成對抗網路 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的效果達到最佳。

貳、 研究目的

綜合上述討論，本研究目的歸納如下：

一、用 python 撰寫將字體檔案（truetype 檔案）轉換成 jpeg 檔案的程式

二、用 python 撰寫將 jpeg 檔案轉換成要輸入模型的 obj 或 csv 檔案的程式

三、用 tensorflow 實作 pix2pix 模型，並探討以下變因對模型的影響：

（一）探討 U-Net 對模型成效的影響

（二）探討 Category Embedding 對模型成效的影響

（三）探討訓練資料集大小對模型成效的影響

四、用 tensorflow 實作 CycleGAN 模型

五、分析 pix2pix 模型和 CycleGAN 模型對字體風格轉換的成效差異

參、 研究設備及器材

一、硬體

（一）筆記型電腦（CPU：Intel Core i7-7700HQ；GPU：GeForce GTX 1060）

二、軟體及工具

（一）ubuntu 16.04（作業系統）

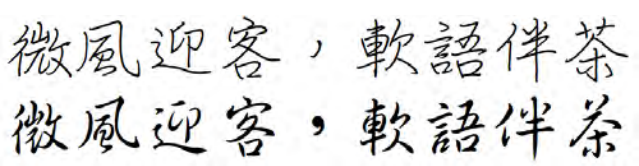
（二）Python 2.7（程式語言）

（三）CUDA 8.0 & cudnn（GPU 運算技術及深層神經網路原式函式庫）

（四）tensorflow 1.2.1（深度學習框架）

（五）相關套件：Pillow(PIL)、numpy、scipy、imageio

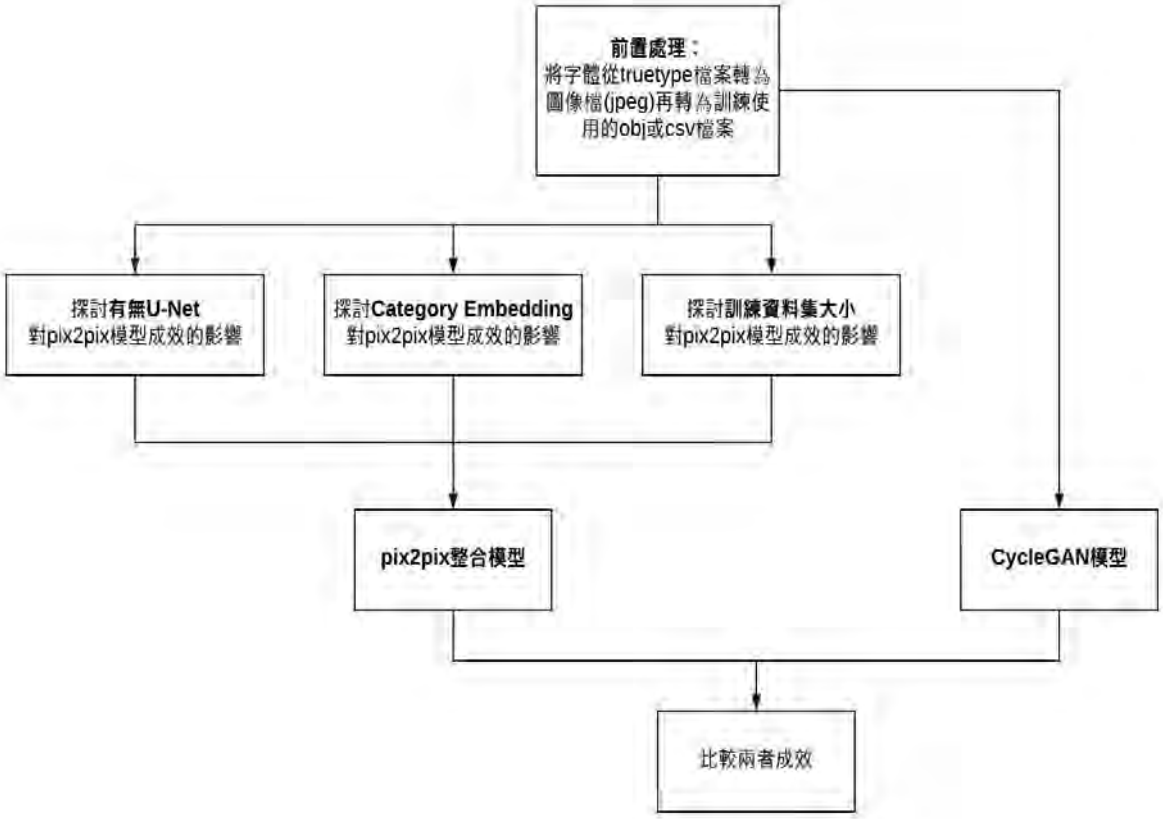
（六）字體的 truetype 檔案（本研究主要使用下列兩種字體，如圖一所示，其餘字體就不一一列舉）



圖一：本研究所用字體示意圖
上方字體：文鼎細鋼筆行楷
下方字體：王漢宗中行書繁
使用此二字體的主要原因在於兩字體有一定的相似特徵，卻又有滿多不同之處，較能有效鑑別兩種變因或方法的好壞

肆、研究過程與方法

一、研究架構



圖二：實驗流程圖

本研究的架構是先將字體檔案轉為模型接受的 obj 或 csv 檔案，並實作出原始的 pix2pix 模型，然後探討不同方法及變因對 pix2pix 模型的影響，最後建立整合模型和 CycleGAN 模型進行比較。以下將分則說明詳細內容。

二、前置處理

（一）pix2pix

1. 將 truetype 檔案轉為 jpeg 檔案（程式名：font2img）：

(1) 創建 cjk.json 的字元集合檔案

(2) 從 cjk.json 中隨機取出指定數量的字（例如：2000 字）

(3) 運用 Pillow 套件將每個字的源字體和目標字體形態畫在圖片上（如圖三所

示，左邊是目標字體，右邊是源字體）

圖三：圖片檔案示意圖（左右兩字屬於同一張圖片）

2. 將 jpeg 檔案轉為訓練使用的 obj 檔案（程式名：package）：

(1) 運用 cpickle 套件將 jpeg 檔案轉為 obj 檔案

（二）CycleGAN

1. 將 truetype 檔案轉為 jpeg 檔案（程式名：font2img）：

(1) 程式同 pix2pix 的 font2img 程式，但因 CycleGAN 處理的是不成對資料集的問題，故配對的兩個字是不同的，而且是兩張分開的圖片 。

圖四：圖片檔案示意圖（左右兩字是不同的兩張圖）

2. 將 jpeg 檔案轉為訓練使用的 csv 檔案（程式名：createCycleganDataset）：

(1) 運用 click 和 csv 套件將 jpeg 檔案轉為 csv 檔案

三、pix2pix

pix2pix 是針對圖像風格轉移領域的一種方法，其目標是在建立一個能處理絕大多數圖像風格轉移問題的大一統模型。pix2pix 的概念是以生成對抗網路（GAN）為基礎，並在模型上作些許調整所得出的一種方法。

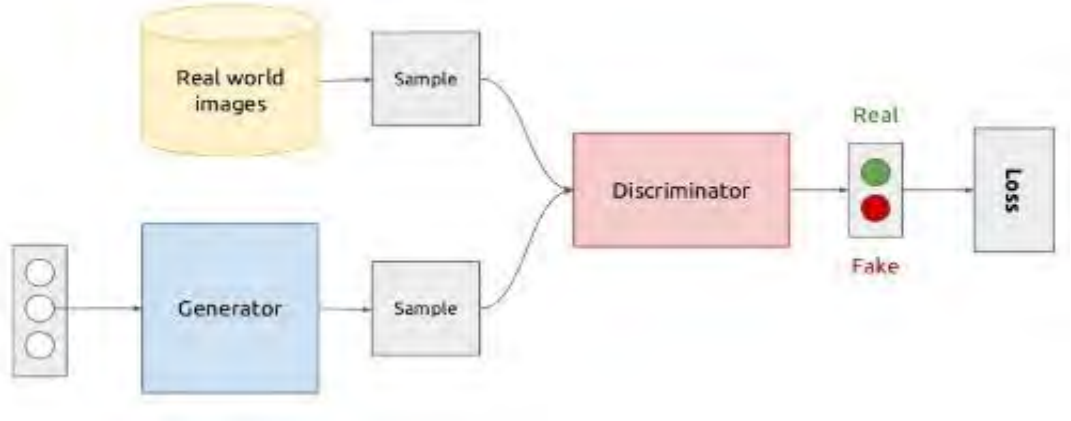
（一）生成對抗網路 Generated Adversarial Network（GAN）

生成對抗網路簡稱 GAN，其啟發自博弈論中的二人零和博弈，GAN 模型中包含兩位博弈者：生成器（generator）和判別器（discriminator）。生成器 G 捕捉樣本資料的分布，用服從某一分布（均勻分布、高斯分布等等）的雜訊 z 生成一個類似真實訓練資料的樣本，目標是越像真實樣本越好；判別器 D 則是一個二分類器，估計一個樣本來自於真實訓練資料的機率，若其判斷樣本來自真實訓練資料，就輸出大機率，反之則為小機率，目標是能判斷每一個樣本是真是假。

藉由生成器和判別器之間的對抗，可以讓生成器達到更佳的學習效果。

上述過程如下列公式及下圖所示：

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log 1 - D(G(z))] \quad -\text{Eq}(1)$$



圖五：生成對抗網路（GAN）模型示意圖

生成器 G 學習生成和訓練資料相似的圖片，判別器 D 學習判斷圖片的真偽

（二）模型

pix2pix 的模型與 Conditional GAN 大致相同，僅在幾處做了些許的改變：

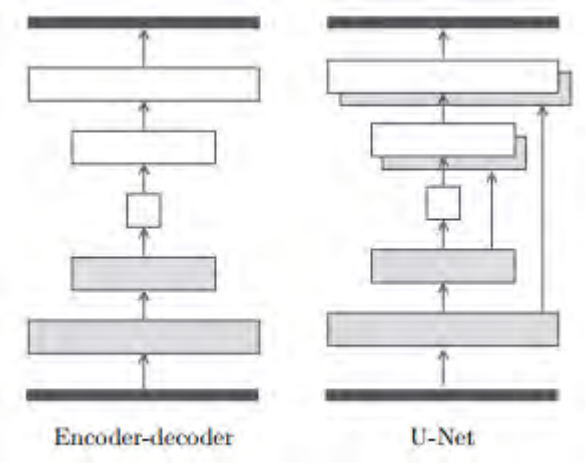
3. Conditional GAN（簡稱 cGAN）：

cGAN 與一般的 GAN 只有一點微小的差別，就是 GAN 的生成器是學習隨機雜訊 z 和輸出圖片 y 之間的映射函數（即 G : z → y），但 cGAN 是學習輸入圖片 x 及隨機雜訊 z 和輸出圖片之間的映射函數（即 G : {x, z} → y），其餘架構都與 GAN 相同。

4. 改變一──加入「跳躍連結」：

大部分 cGAN 的生成器都是使用編碼─解碼器的網路架構（encoder-decoder network），在此架構中，輸入會通過一連串逐漸縮小的神經層，直到通過瓶頸層之後又逐漸變大（如圖三左圖所示）。在這個網路中，每一項資訊都會流過所有的神經層，包括最小的瓶頸層。然而，對於圖像風格轉移問題，輸入與輸出的圖片勢必有些資訊是相同的，因此，若能讓這些資訊直接流過網路，而不必通過每一個神經層是較為合理的。實現此想法的具體作

法是加入「跳躍連結」，設 n 為神經層的總數，在第 i 層和第 n-i 層之間都加入一個「跳躍連結」，每一個跳躍連結可讓資訊直接流過。這個做法會讓模型變成「U-Net」的形式（如圖三右圖所示）。



圖六：兩種生成器的網路架構
左邊是編碼—解碼器的網路架構，為一般 cGAN 所使用；右邊則是 pix2pix 模型做出的改變：在以瓶頸層為對稱的兩神經層中都加入「跳躍連結」，形成「U-Net」的網路形式

5. 改變二──Markovian discriminator（PatchGAN）：

L1 loss 和 L2 loss 會造成生成出來的圖片十分模糊是已知的事實，但其實 L1 loss 固然在圖案較複雜的地方會造成模糊，在簡單的地方卻能準確的轉換，因此，在這裡只需要 GAN 的判別器去判斷圖案複雜處的真偽，剩下的只要依賴 L1 loss 即可。具體做法是設計一個判別器模型，它只會在 N x N 大小的方塊範圍內判別真偽，而此判別器會在圖片上循環遊走，進行判別。

（三）損失函數（loss）的設定

1. Adversarial Loss

(4) cGAN 的損失函數形式：

$$L_{cGAN}(G,D) = E_{x,y\sim p_{data}(x,y)}[\log D(x,y)] + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z))] \quad -Eq(2)$$

其中 G 的目標是讓 G(x)看起來越像 y 訓練集的資料越好，而 D 則是要分辨假樣本 G(x)和實際樣本，也就是說，G 想要最小化此損失函數，而 D_Y 卻想要最大化它，以數學式表示則有以下形式：

$$G^* = \arg\min_G \max_D L_{cGAN}(G,D) \quad -Eq(3)$$

(5) 判別器的輸入沒有真實圖片的損失函數形式：

$$L_{GAN}(G,D) = E_{y\sim p_{data}(y)}[\log D(y)] + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(G(x,z))] \quad -Eq(4)$$

2. L1 loss

由於前述提到在模型中加入 L1 loss 可以幫助生成低複雜圖圖案處的圖片，故此處也附上 L1 loss 的數學形式：

$$L_{L1}(G) = E_{x,y\sim p_{data}(x,y),z\sim p_z(z)}[\|y - G(x,z)\|_1] \quad -Eq(5)$$

3. 整體損失函數

此模型的整體損失函數由上述的 adversarial loss 和 L1 loss 組合而成，由 λ 控制兩者的比重。而此模型的目標是要讓生成器 G 和 F 學習到讓生成出的圖片和實際資料越像的特徵，故可用以下數學形式加以表達：

$$G^* = \arg\min_G \max_D L_{cGAN}(G,D) + \lambda L_{L1}(G) \quad -Eq(6)$$

（四）已做過的調整

1. Category Embedding

根據設計師設計字體的經驗，讓神經網路同時學會多種字體風格是十分重要的，因為同時對多種風格建模可以讓編碼器接觸到更多漢字，且不僅限於一個目標字體，還包括所有字體的組合，此外，這種作法也能讓解碼器從其他自體學會同一種偏旁的不同寫法。

然而，這種作法會面臨的問題是同一個漢字可以出現在多種字體當中，而原本的 pix2pix 模型並沒有解決這種一對多的關係。因此，在此處需要使用「類別嵌入」（category embedding）的作法，將不可訓練的高斯雜訊在資訊進入解碼器之前輸入神經網路，作為一種嵌入的風格（style embedding），如此一來，解碼器就會考慮原本的漢字以及此嵌入的風格來生成目標漢字。

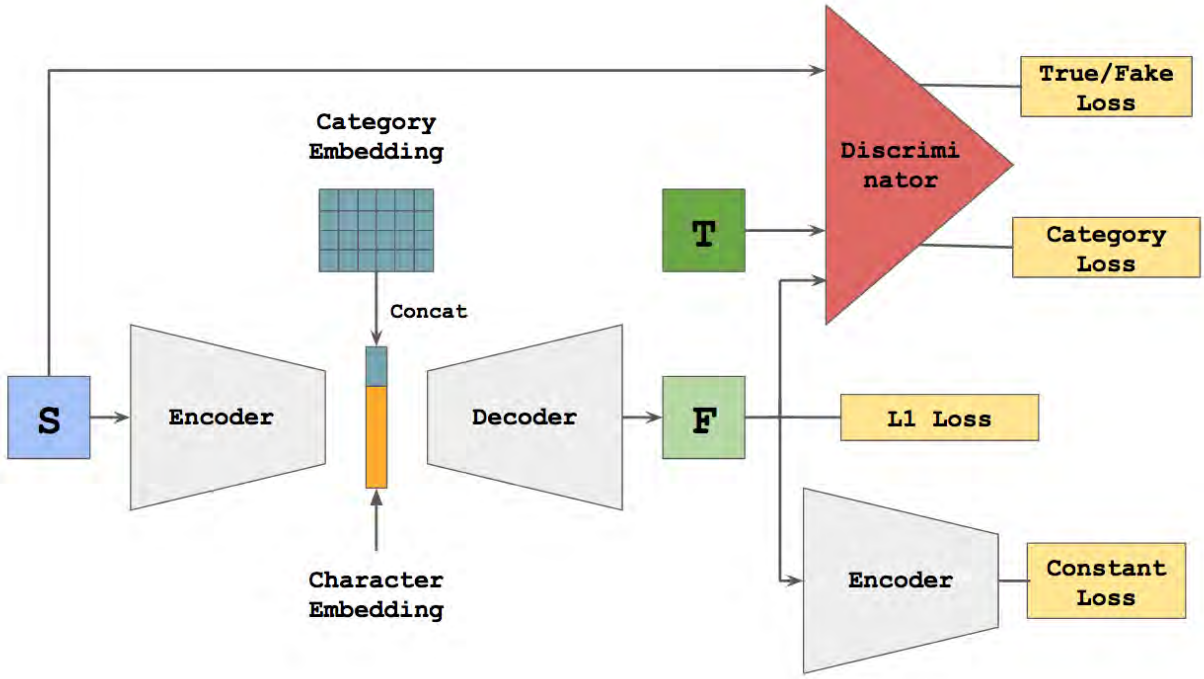
2. Multi-class Category Loss

利用 Category Embedding 就能同時對多種風格的字體建模，然而此作法會衍生一個新問題：模型開始將各種風格混淆在一起，導致生成的字體甚麼都不像。

因此針對這個問題，在此處要引入 AC-GAN 模型中的 multi-class category loss，將此損失函數加到判別器上，以此去「懲處」風格混淆的情況，就能有效地保存每一種風格。

3. Constant Loss

在模型中也引入了 DTN 神經網路中使用的 constant loss。此損失函數的基本原理就是：原本的字元和生成的字元對應的應該是同一個漢字，所以他們在圖片空間中的對應位置應該要十分相似。而結果證實 constant loss 能藉由讓編碼器保留生成漢字的識別性，縮小搜尋範圍，大大改善了收斂速度。

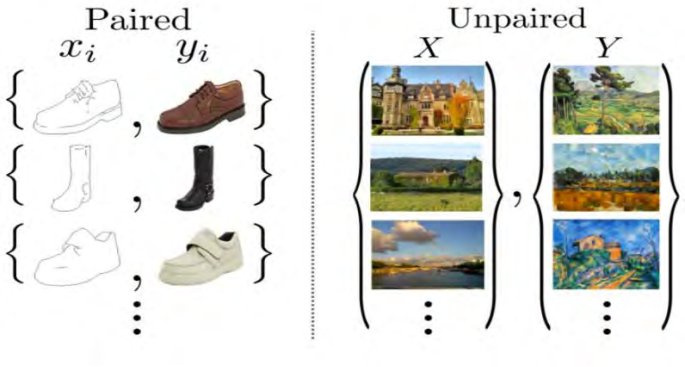


圖七：經調整後的 pix2pix 模型示意圖

整體架構與 cGAN 大致相同，而上述的三個調整：category embedding、category loss 和 constant loss 都加入到模型中，分別發揮各自的功能

四、Cycle-Consistent Adversarial Networks（CycleGAN）

圖像風格轉移是計算機視覺領域的一個重要分支，常見的作法是使用成對的訓練圖片集，讓機器去學習輸入圖片與輸出圖片之間的映射關係。然而，對於許多任務，成對的訓練圖片集可能不易或無法取得，CycleGAN 即為針對此一問題的一個解決方法。

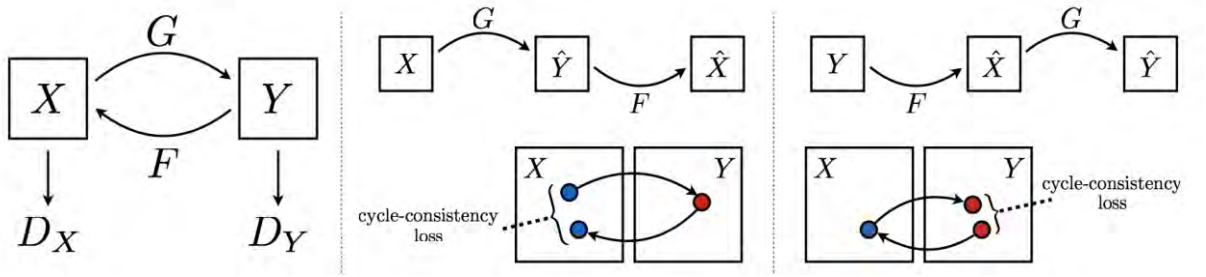


圖八：成對及非成對圖片示意圖
成對圖片間具有類似的特徵，如輪廓等等，而非成對圖片則否。

（一）Cycle Consistency

CycleGAN 和 pix2pix 一樣都是由 GAN 衍生而來，但由於訓練圖片集並非成對，故無法保證 GAN 訓練出的模型會學習到如預期中的映射關係，因此模型中必須加入一些其他架構。在這裡會應用到一種關係：一張圖片經過一次變換及一次逆變換之後應該要變回原本的圖片，也就是說，若這裡有兩個轉換器 $G : X \rightarrow Y$ 和 $F : Y \rightarrow X$，則 G 和 F 互為彼此的逆變換，會有 $F(G(x)) \approx x$ 和 $G(F(y) \approx y$ 的情形。此即 Cycle Consistency 的精髓，並應用到 CycleGAN 模型中得到很好的效果。

（二）模型



圖九：CycleGAN 模型示意圖

1. 此模型包含兩個映射函數 $G : X \rightarrow Y$ 和 $F : Y \rightarrow X$，以及相關的判別器 D_Y 和 D_X，D_Y 會激勵 G 生成近似於 Y 訓練集中資料的樣本，而 D_X 則會激勵 F 生成近似於 X 訓練及資料的樣本。
2. 為了更有效的訓練映射函數，模型中也應用了 Cycle Consistency 的原理而放入了兩個 cycle consistency loss：

   (1) 正向的 cycle consistency loss：$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$（如上中圖）

(2) 反向的 cycle consistency loss：y → F(y) → G(F(y)) ≈ y（如上左圖）

（三）損失函數（loss）的設定

1. Adversarial Loss

對於映射函數 G : X → Y 和其判別器 D_Y，其 adversarial loss 表示如下：

$$L_{GAN}(G, D_Y, X, Y) = E_{y\sim p_{data}(y)}[\log D_Y(y)] + E_{x\sim p_{data}(x)}[\log(1 - D_Y(G(x))] \quad -Eq(7)$$

其中 G 的目標是讓 G(x)看起來越像 Y 訓練集的資料越好，而 D_Y 則是要分辨假樣本 G(x)和實際樣本，也就是說，G 想要最小化此損失函數，而 D_Y 卻想要最大化它，以數學式表示則有以下形式：

$$\min_G \max_{D_Y} L_{GAN}(G, D_Y, X, Y) \ -Eq(8)$$

此損失函數也以同樣的方式應用在 F : Y → X 和其判別器 D_x 上。

2. Cycle Consistency Loss

對於前述所提的 cycle consistency 也定義了其損失函數，形式如下：

$$L_{cyc}(G, F) = E_{x\sim p_{data}(x)}[\|F(G(x)) - x\|_1] + E_{y\sim p_{data}(y)}[\|G(F(y)) - y\|_1] \quad -Eq(9)$$

3. 整體損失函數

此模型的整體損失函數由上述的 adversarial loss 和 cycle consistency loss 組合而成，形式如下：

$$\mathrm{L}(\mathrm{G}, \mathrm{F}, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F) \ -Eq(10)$$

其中 λ 控制了兩種損失函數的比重。而此模型的目標是要讓生成器 G 和 F 學習到讓生成出的圖片和實際資料越像的特徵，故可用以下數學形式加以表達：

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_y} L(G, F, D_X, D_Y) \ -Eq(11)$$

伍、研究結果

以下表格中的圖片，若未特別說明，皆為以文鼎細鋼筆行楷為源字體、王漢宗中行書繁為目標字體訓練到第 50 個 epoch 的結果，且左側文字為王漢宗中行書繁字體的實際模樣，右側文字為模型預測出來的風格。

一、探討 U-Net 對模型成效的影響

表一：實驗一結果圖片表格

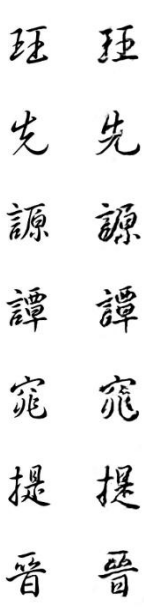
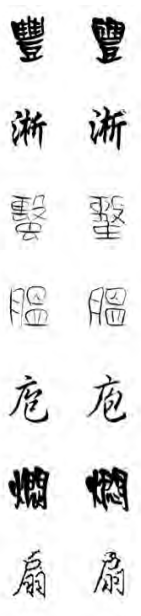
有無 U-Net	有	無
結果（圖片）		

二、探討 Category Embedding 對模型成效的影響

用來訓練的十種字體：

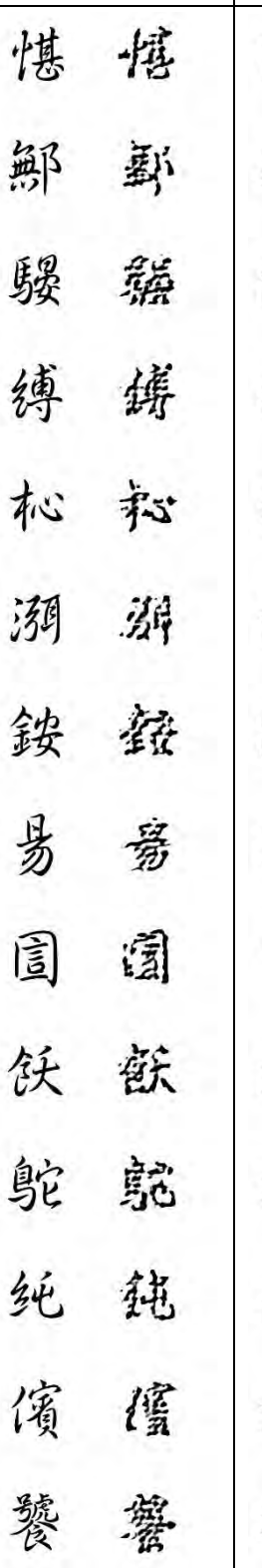
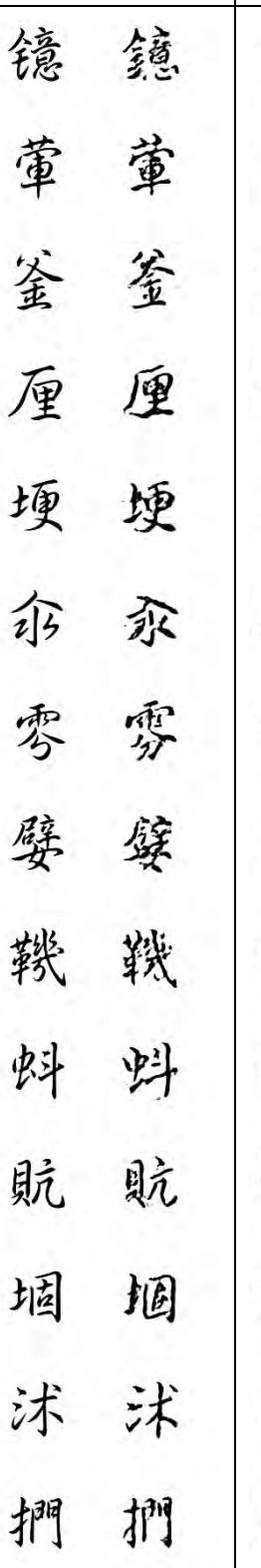
王漢宗中行書繁、文鼎中特廣告體、華康流葉體、華康彩帶體、王漢宗鋼筆行楷繁、王漢宗細圓體繁、王漢宗粗鋼筆——標準、文鼎 POP-4、文鼎行楷碑體、文鼎細鋼筆行楷

表二：實驗二結果圖片表格

有無進行 Category Embedding	無	無	無	有（4 種）	有（10 種）
源字體	文鼎細鋼筆行楷	文鼎細鋼筆行楷	文鼎細鋼筆行楷	文鼎細鋼筆行楷	文鼎細鋼筆行楷
目標字體	王漢宗中行書繁	文鼎中特廣告體	華康流葉體	上述前三種字體	上述前九種字體
結果（圖片）					

三、探討訓練資料集大小對模型成效的影響

表三：實驗三結果圖片表格

訓練圖片數（張）	500	1000	1500	2000
結果（圖片）				

四、比較 pix2pix 模型及 CycleGAN 模型在字體風格轉換方面的成效

由於 CycleGAN 並非使用成對圖片集作為訓練資料，因此輸出結果並不像其他部分一樣左邊是實際文字，右邊是模型預測的文字，而會是左半部的兩個文字是輸入的不成對圖片，右邊的兩個文字是模型輸出的結果，也就是說，模型會分別訓練兩種字體互相轉換的機制並輸出結果。

（一）源字體與目標字體形態差異較小的情況

源字體：文鼎細鋼筆行楷

目標字體：王漢宗中行書繁

表四：實驗四-1 結果圖片表格

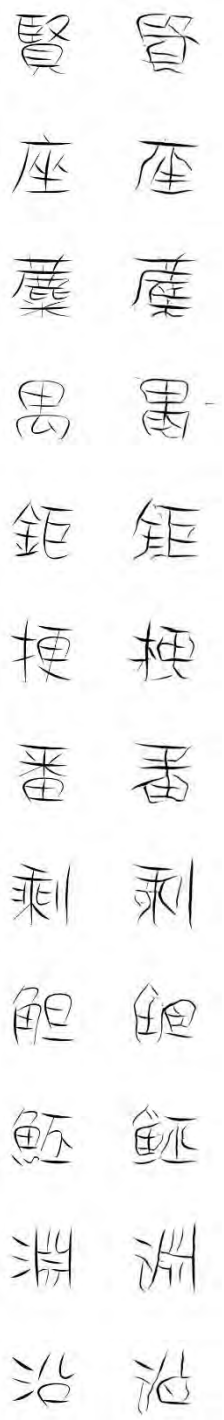
模型	pix2pix	CycleGAN
結果（圖片）		

（二）源字體與目標字體形態差異較小的情況

源字體：文鼎細鋼筆行楷

目標字體：華康流葉體

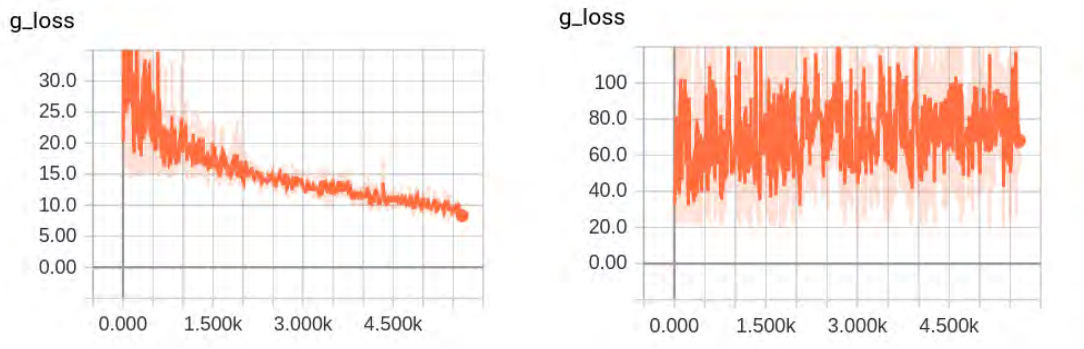
表五：實驗四-2 結果圖片表格

模型	pix2pix	CycleGAN
結果（圖片）		

陸、討論

一、探討 U-Net 對模型成效的影響

由結果圖片可看出，使用 U-Net 的結果明顯比未使用來的好很多，不僅字型的結構較為完整，預測的圖片也較為清晰。此外，觀察兩實驗生成器 loss 的趨勢圖（圖八）也可發現，有加入 U-Net 可讓 loss 逐漸變小，未加入則會讓 loss 不斷來回擺盪，無法有效降低 loss 值，達到較好的優化。由此我們可以推論，在 pix2pix 模型中，生成器在 encode 和 decode 的過程裡會使原有的一些特徵消失或模糊化，故在編碼器及解碼器之間加入 skip connection 使資訊連通能有效改善模型的成效。



圖十：U-Net 實驗的生成器 loss 趨勢圖
圖左是有使用 U-Net 的實驗，圖右則是未使用的，能明顯看出有加 U-Net 能使 loss 逐漸降低，未加 U-Net 則無法

二、探討 Category Embedding 對模型成效的影響

由結果圖片可看出，當源字體與目標字體的形態差異較小（如目標字體為王漢宗中行書繁的情況），是否使用 Category Embedding 對結果並沒有很大的影響；但當源字體與目標字體的形態差異較大（如目標字體為華康流葉體的情況），使用 Category Embedding 就有較大的幫助：可以看出在未使用 Category Embedding 的情況時，是無法輕易辨識轉換出來的是哪個字，但在有使用的情況下，這種現象有明顯的改善，雖然成效也並非很好，但至少轉換出來的字型在架構上更接近實際字體。此外，使用越多字體進行訓練會使效果更好。

三、探討訓練資料集大小對模型成效的影響

由結果圖片可看出，訓練圖片數越多，模型轉換的效果越好，訓練圖片只有 500 張

時，訓練出來的模型轉換效果是十分模糊的；而當訓練圖片有 1000 張時，我們比較可以看出轉換出的是甚麼字了，但每個字的細節部分還是很不清楚；用 1500 張圖片去訓練又比 1000 張要好一些；而用 2000 張圖片時，就已經不太會有整個字都很不清楚的情況了，雖然有些字的邊緣還是不太清晰，但大部分的字都已經有目標字體的很多特徵了。

四、比較 pix2pix 模型及 CycleGAN 模型在字體風格轉換方面的成效

由結果圖片可看出，在源字體與目標字體差異小時，CycleGAN 模型的效果比起 pix2pix 模型而言是較優的，因為 CycleGAN 轉換的文字圖片較清晰，邊緣的細節部分較沒有不清楚的情形，能清楚地展現每個字的架構。然而，CycleGAN 模型有一個問題：其訓練結果有與訓練資料過於相似的情形，例如「嘆」這個字的「共」部分（如圖十一），文鼎細鋼筆行楷有多了一撇，而實際上王漢宗中行書繁是沒有那一撇的，但預測結果卻是有那一撇的，這是需要改善的問題。另外，在源字體與目標字體差異大時，可以發現兩種模型轉換出來的效果都不佳，預測出來的字體的筆劃會在不正確的位置，導致整個字難以辨識，在這種情況下仍需以 Category Embedding 融入多種字體進行訓練會有較好的成果。

圖十一：CycleGAN 問題示意圖

柒、結論

一、U-Net 能顯著改善模型的成效。

二、Category Embedding 的成效在源字體和目標字體形態差異小時並不明顯，但在兩字體差異大時就有不錯的成效，且使用越多字體進行訓練會使效果更好。

三、訓練圖片數越多，模型轉換成效越好。

四、CycleGAN 的成效比 pix2pix 略優。

捌、參考資料及其他

[1] 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.

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Imageto-image translation with conditional adversarial networks. In CVPR, 2017.

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In CVPR, 2017.

[4] Augustus Odena, Christopher Olah, Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In stat.ML.

[5] Yaniv Taigman, Adam Polyak, Lior Wolf. Unsupervised cross-domain image generation. In cs.CV.

[6] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat(2017). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. cs.CL. arXiv:1611.04558v2.

[7] kaonashi-tyc(2017). zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. kaonashi-tyc.github.io/2017/04/06/zi2zi.html

[8] Sword York(2016). Generative Adversarial Networks. blog.slinuxer.com/2016/10/generative-adversarial-networks

【評語】052511

本研究以實作字體風格轉換的生成對抗網路模型為動機，將 Conditional GAN 當作模型的基礎，探討 pix2pix 模型及其他研究的一些方法對模型會產生甚麼影響，以得出預測成效的深度學習模型。該研究主題使得中文的字體可以更容易有各式各樣不同種的字型。

本研究具學術性，實驗與分析完整。該研究報告中有針對不同的設計（如 U-Net、Category Embedding）進行效果的量測和探討。

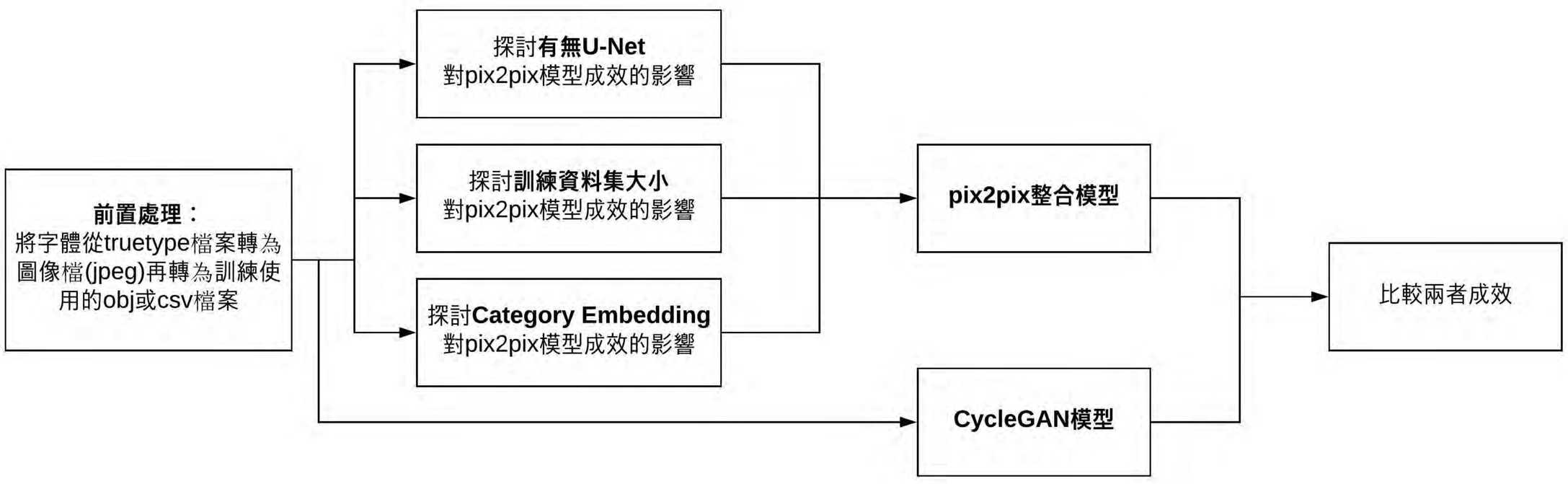
作品海報

研究動機

創造中文字體是一項困難且曠日費時的工作，要創造一套符合標準（如：GBK）的中文字體，設計師要為超過26000個漢字作手工設計，通常需要好幾年的時間來完成。因此，若能使用深度學習的方式，先手工設計一部分的字體，剩下的用深度學習的模型去作轉換，就能大幅減少創造字體所需的時間。本實驗的目的就是在建立並製作字體轉換的模型，並探討各調整對模型的影響

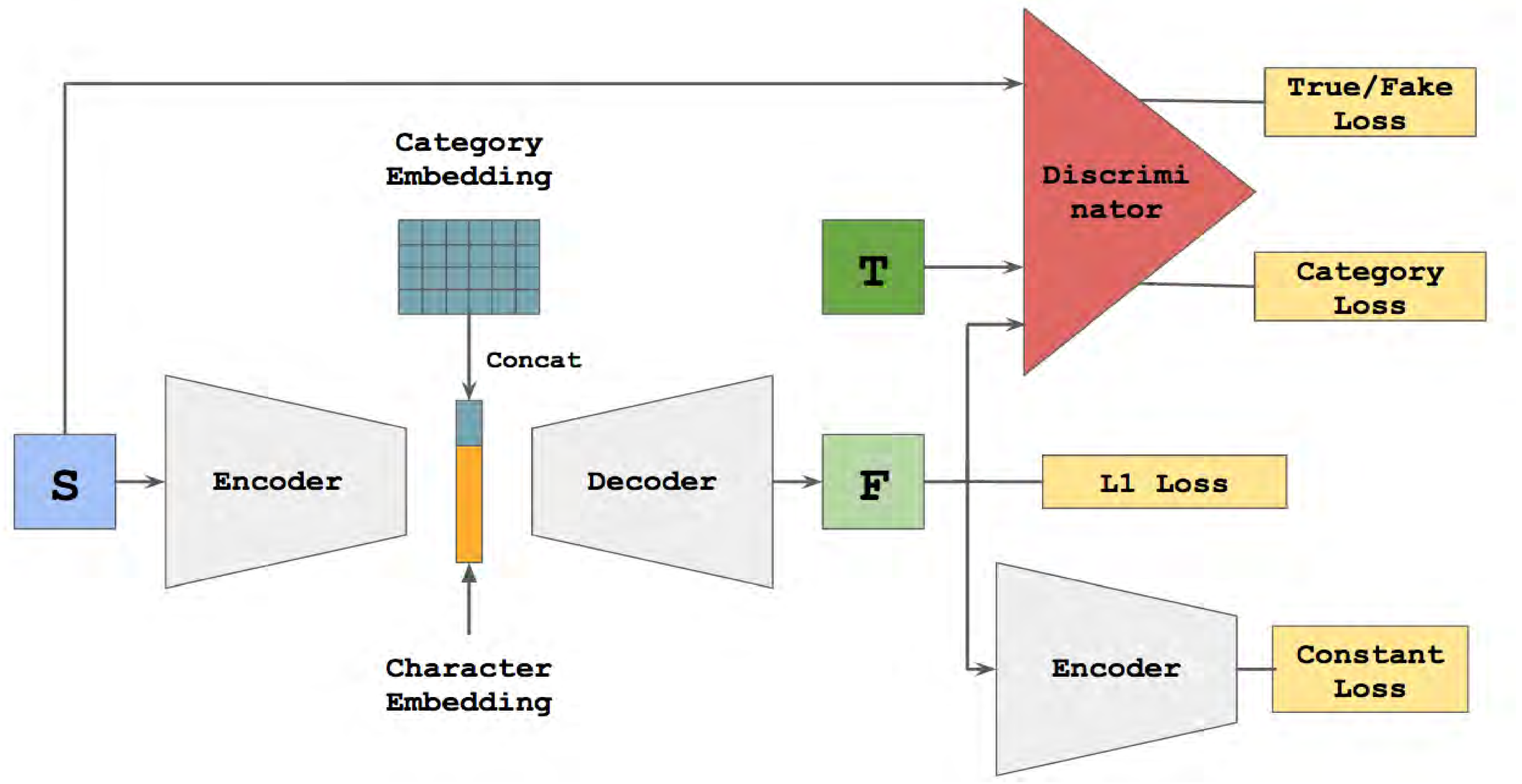
研究目的

以Conditional GAN為基礎，建立並實作字體風格轉換的深度學習模型：

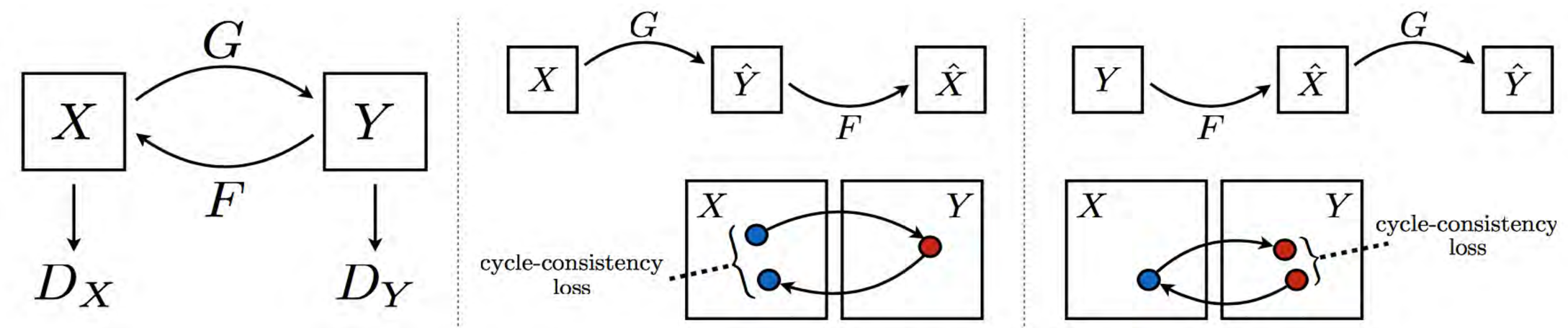


實驗流程圖

研究過程及方法



調整後的pix2pix模型示意圖



CycleGAN模型示意圖

準確度測量

- Pixel accuracy = $\frac{\sum_i n_{ii}}{\sum_i t_i}$
- Mean accuracy = $\frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i}$
- Mean IU = $\frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$

n_{ij}：屬於類別 i ，但被預測為類別 j 的像素數

n_{cl}：類別總數

t_i：類別 i 總像素數

研究結果

以下結果圖片，若未特別說明，皆為以文鼎細鋼筆行楷為源字體、王漢宗中行書繁為目標字體訓練到第100個epoch的結果，且左側文字為源字體的實際模樣，中間字體為模型預測結果，右側則是目標字體的實際模樣

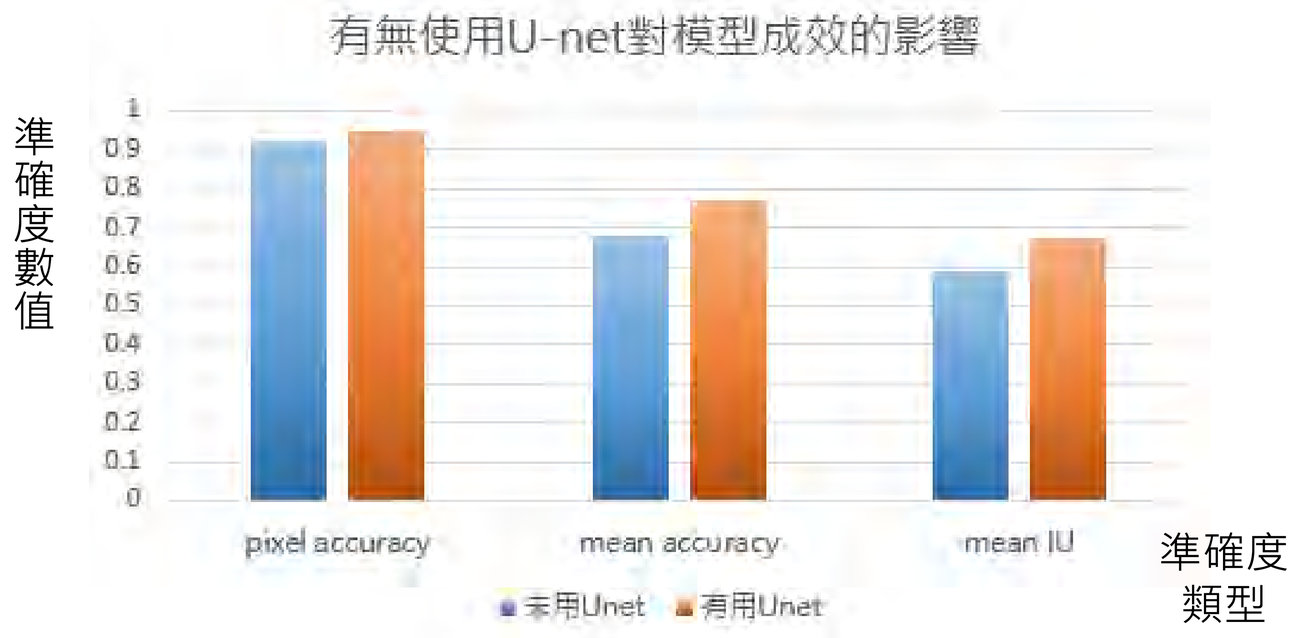
實驗一、探討U-Net對模型成效的影響

未用U-Net

1.7

有用U-Net

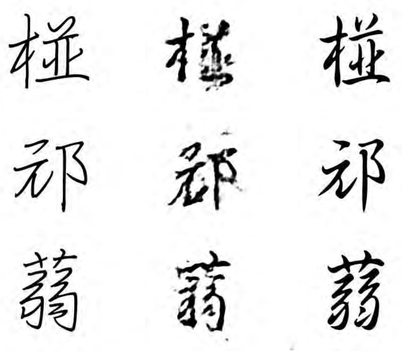
7.8



實驗二、探討訓練資料集大小對pix2pix模型成效的影響

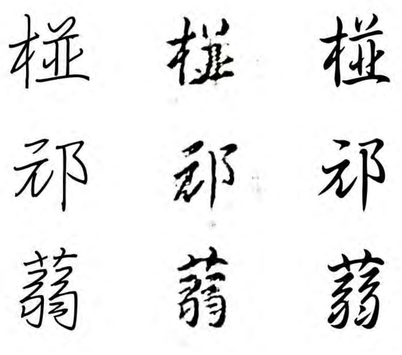
500張訓練圖片

4.2



1000張訓練圖片

4.0



1500張訓練圖片

6.1

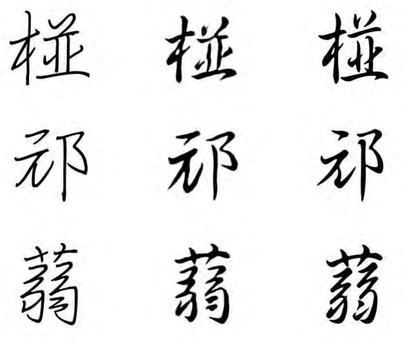
2000張訓練圖片

7.8

實驗三、探討Category Embedding(簡稱CE)對pix2pix模型成效的影響

用來訓練的十種字體：

王漢宗中行書繁、文鼎中特廣告體、華康流葉體、華康彩帶體、王漢宗鋼筆行楷繁、王漢宗細圓體繁、王漢宗粗鋼筆──標準、文鼎POP-4、文鼎行楷碑體、文鼎細鋼筆行楷



未用CE

目標：王漢宗中行書繁

7.8

有用CE(10種字體)

目標：王漢宗中行書繁

6.9

未用CE

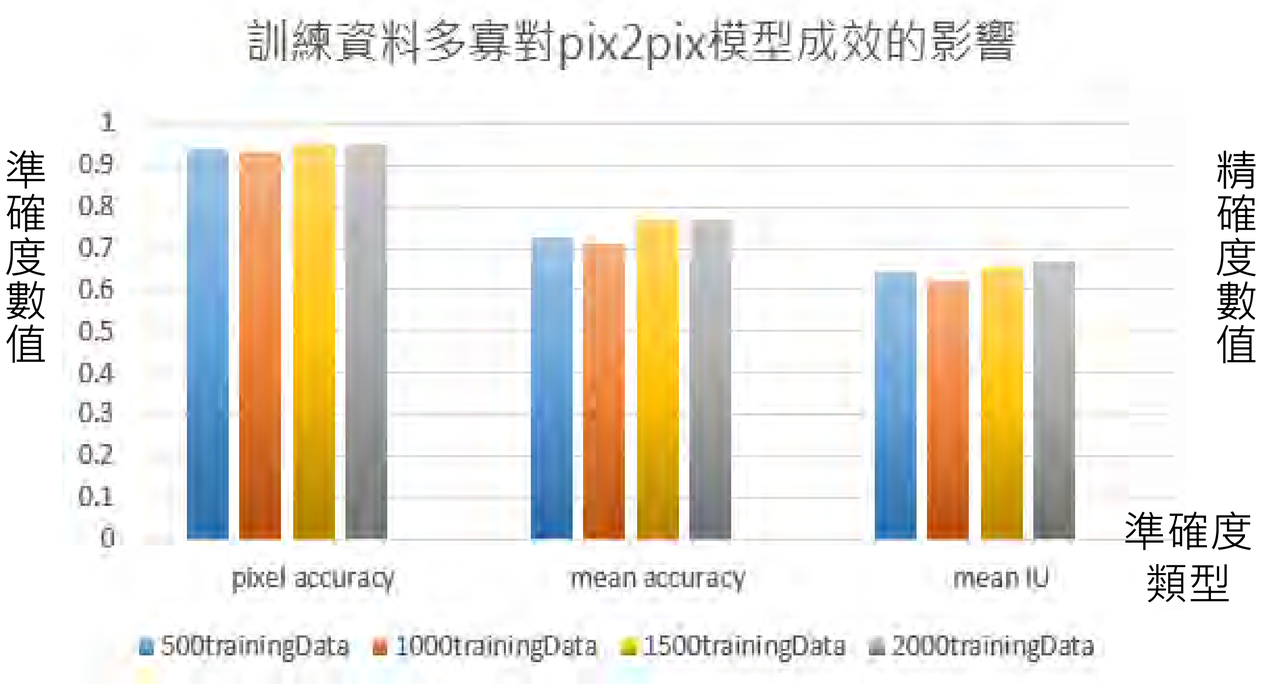
目標：華康流葉體

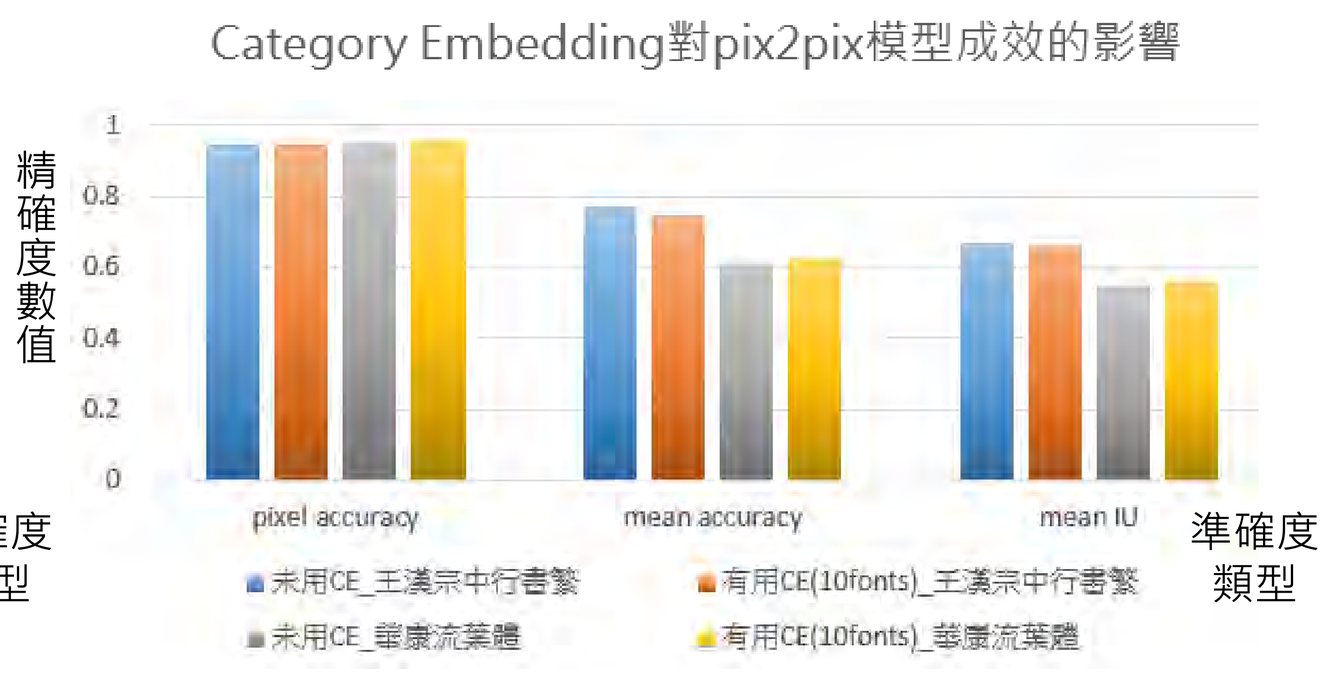
5.6

有用CE(10種字體)

目標：華康流葉體

6.3





實驗四、比較pix2pix模型及CycleGAN模型在字體風格轉換方面的成效

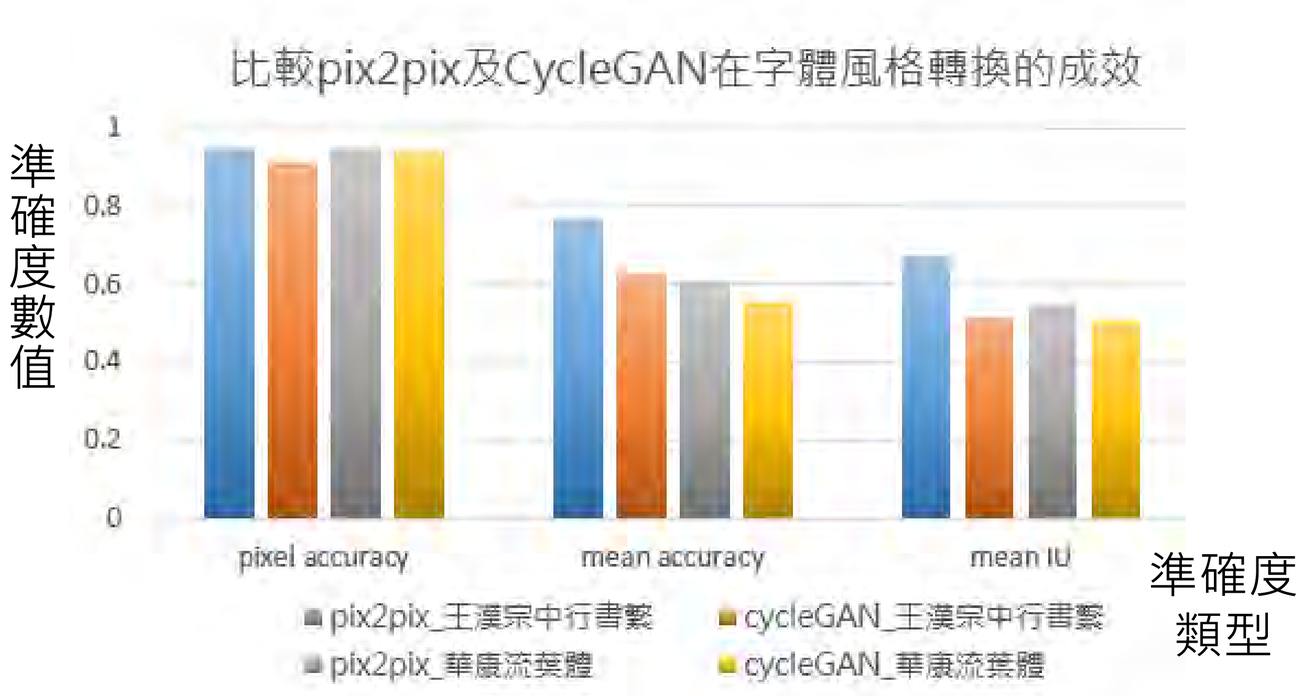
模型：pix2pix 王漢宗中行書繁	模型：CycleGAN 王漢宗中行書繁	模型：pix2pix 華康流葉體	模型：CycleGAN 華康流葉體
7.8	5.8	5.6	5.2

討論

實驗一、探討U-Net對模型成效的影響

使用U-Net的結果比未使用好很多：

1. 字型結構完整、圖片清晰
2. 在不同準確度測量及主觀測試下都表現較優



實驗二、探討訓練資料集大小對模型成效的影響

1. 訓練圖片數越多，模型預測效果越好
2. 使用大約2000張訓練圖片，就能使模型達到非常良好的預測結果

實驗三、探討Category Embedding對模型成效的影響

1. 當源字體與目標字體的形態差異小，Category Embedding對結果沒有大影響
2. 當源字體與目標字體的形態差異較大，使用Category Embedding就有較大的幫助，字形較為清楚、且在準確度測量及主觀測試上表現較佳

實驗四、比較pix2pix模型及CycleGAN模型在字體風格轉換方面的成效

1. 不論源字體與目標字體的形態差異大或小，pix2pix的成效都比CycleGAN優異
2. 推測原因：CycleGAN的輸入缺乏與源字體一對一對應的圖片，導致源字體與目標字體的映射關係較難建立，因此模型預測效果較差

結論

由實驗結果我們得知：在源字體與目標字體形態差異小的情況下，在pix2pix模型中加入U-Net並輸入2000張左右的訓練圖片可使模型產生優良的預測結果；而當源字體與目標字體形態差異大，則需要運用Category Embedding的概念，融入多種字體進行訓練來改善模型的預測成效。

未來展望

1. 模型在源字體於目標字體差異大時成效仍不佳，希望能想出其他方法進行改善
2. 將模型應用在個性化字體或是藝術字的創造

參考資料

[1] P. Isola et al. Image-to-image translation with conditional adversarial networks

[2] J. Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks