國立台北大學資訊工程學系專題報告

Text Logo Generator

專題組員：鄭卓欣、郭子熠、鍾秦陞、林琮翊

專題編號：PRJ-CSIE-112-001
執行期間：2023 年 7 月 至 2024 年 5 月

一. 摘要

本專題整合一個產生文字排版的系統。主要把產生字體、生成排版和加上材質三項功能做串接。分別參考三篇論文 Attr2Font[1]、TextLogo Layout[2]、Texture Reformer [3]和其提供的程式。

除了修改及銜接以上三個系統之外，我們也提供使用者介面，讓使用者能夠更容易操作我們所製作的系統。

二. 簡介

(一) 研製背景

近年來，人工智慧引起了人們的極大興趣。AI 設計 Text Logo 是一種很有價值的技術，它對促進設計師的任務和提高創建媒體內容的效率非常有幫助。

(二) 研究目標

我們希望完成一個系統架構。功能包括從產生字體、產生排版到加上材質，完成整體的系統。並提供使用者介面，使整體流程易於檢視和操作。

(三) 主要預期效益

透過使用這樣的系統，我們希望能提供一個讓使用者滿意的程式。使他可以根據想呈現的效果，通過調整字體屬性、挑選排版和材質，產生好看的 Text Logo。

三. 專題進行方式

(一)系統架構

本專題的系統流程如 Fig.1，可分為以下四個系統：(1)user interface、(2)Attr2Font、(3)TextLogo Layout、(4)Texture Reformer。

1. User Interface

介面(user interface)主要功能為：(1)設定屬性、(2)選擇來源字體、(3)輸入要排版的文字、(4)上傳欲更換的材質、(5)觀察過程與結果，執行整個流程。屬性藉由控制滑標控制(共 37 個)與其他決定字體材質功能都是藉由 HTTP POST 與兩個 flask server 進行通信。

2. Attr2Font

在這一階段會依照使用者在介面上設定 37 種屬性的值及選定的來源字體生成一組新的字體，結果字體的字體架構是以來源字體為基礎，而字體風格是依據使用者設定的屬性值去做生成，結果包含了 26 個英文字母的大寫、小寫及數字 0 到 9 字體圖片。

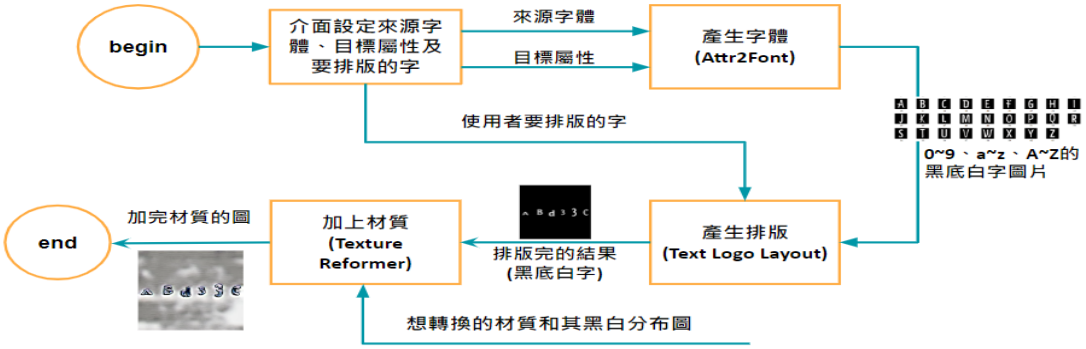

Fig.1 Text Logo Generator 系統總流程圖

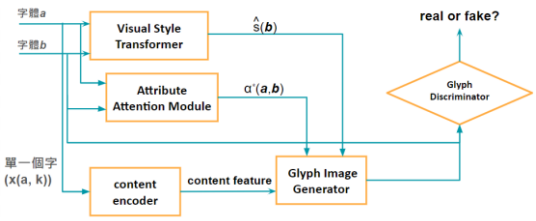

Fig. 2 Attr2Font 訓練流程圖

Attr2font[1]的訓練流程如 Fig. 2 所示，此模型是一個 GAN 架構，輸入是 2 種字體的圖片和他們各自的屬性表，屬性的部分是由 Attr2font[1]的資料集提供，以下使用 a 和 b 分別代表 2 種字體，訓練目的是希望字體 a 透過 2 字體屬性差生成出字體 b。

Glyph Image Generator 需要 3 個輸入：$\hat{s}(\boldsymbol{b})$、$\alpha^*(\boldsymbol{a},\boldsymbol{b})$及 content feature。

第一個輸入$\hat{s}(\boldsymbol{b})$由 Visual Style Transformer 得到，此部分需要字體 a 的圖片和字體 a、b 的屬性，首先會先從字體 a 的圖片取 m 張進到 CNN 得到字體 a 的風格特徵即$\hat{s}(\boldsymbol{a})$，這邊 m 越大越好，但由於電腦效能因素，沿用該論文的設定 m=4，接著將$\hat{s}(\boldsymbol{a})$和字體 a、b 屬性的差連接進到數個 residual blocks，得到預測的字體 b 的風格特徵即$\hat{s}(\boldsymbol{b})$。

第二個輸入 $\alpha^*(\boldsymbol{a},\boldsymbol{b})$從 Attribute Attention Module 獲得，此部分需要字體 a、b 的屬性，經過一些矩陣運算和捲積層，得出 $\alpha^*(\boldsymbol{a},\boldsymbol{b})$，這可以在生成圖片時，讓模型知道哪些屬性較為重要及屬性之間的相關性，助於優化生成結果。

第三個輸入 content feature 即為字體 a 的單個字的內容架構特徵，這部分需要字體 a 的單個字的圖片(x(a,k), k 表示 a 字體中的第 k 個字)當作輸入，進到 content encoder 得到 content feature，此 encoder 為 CNN 架構，得到的結果會在後續生成字體圖片時更精確重構出該字形。

最後將$\hat{s}(\boldsymbol{b})$、$\alpha^*(\boldsymbol{a},\boldsymbol{b})$及 content feature 輸入進 generator 中就可以得到字體風格為$\hat{s}(\boldsymbol{b})$的新字體了，結果如 Fig. 3 所示。

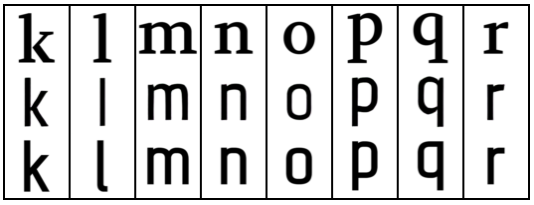

Fig. 3 訓練的部分結果，字體 a(上)、字體 b(下)、模仿結果(中間)

3. TextLogo Layout

TextLogo Layout[2]的訓練流程如 Fig. 4 所示，此模型是一個 GAN 的架構，但比較特別的是會用 1 個 generator 和 2 個 discriminator 來進行訓練。

首先 generator 要準備 2 組輸入，第一組輸入——condition feature 需要先取得從 Attr2Font 的字體圖以及 character embedding(f^e)，然後將字體圖透過以 CNN 為基礎製作的 visual encoder 得到 Image Feature(f^v)，再來將 f^e和 f^v輸入進以 RNN 製作的 condition encoder 就能獲得 condition feature(f^c)；第二個輸入是從 standard normal distribution 中取樣出來的 z，有了這兩個輸入 generator 就能生成每個字的中心座標(x^c, y^c)、寬度以及高度，再來就只要透過 differentiable composition 並根據這些參數將每個字做適當的平移及縮放就可以得到排版完的圖了。

而這兩個 discriminator 的用途也不太相同。首先 sequence discriminator 的任務是分析根據這一組組的參數進行排版後是否會造成字與字重疊，或是能否流暢閱讀。但由於 sequence discriminator 是用 RNN 實作的，因此需要再利用 CNN 實作的 image discriminator 去分析結果圖的細節，如筆畫或是字本身是否

扭曲變形等等。

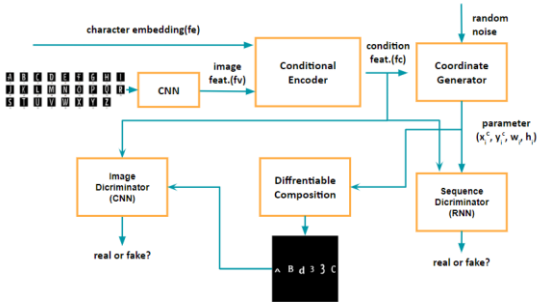

Fig.4 TextLogo Layout 訓練流程圖

4. Texture Reformer

經過上一個階段，我們得到排版完且黑底白字的圖。而在這一階段，目的是把剛剛排版完的圖，根據給定的材質產生加完材質的圖。

使用的是 Texture Reformer [3]提供的程式，流程如 Fig.5 所示。輸入需三張圖片，一張是要被加上材質的圖(T^{sem})也就是剛剛排版完的圖，還有材質(S^{sty})和其分布圖(S^{sem})。總共會執行三個階段，每一階段的成果為 T^{sty}。

第一階段(Global View Structure Alignment)和第二階段(Local View Texture Refinement)，都是使用 Semantic-Guided Texture Warping(SGTW)，目的是為了用 S 去決定 T。概念上 SGTW 的作法可以想成是把 T(Target)的圖和 S(Source)的圖都切成一塊塊。T 的每一塊都根據相似度和 S 的圖對照。之後把 S 的每一塊根據對照，貼到 T 上。就產生出這一階段的結果。(以上都是大概的介紹，實際上有很多細節。可參考原論文[3])

第一階段和第二階段的差別是在於切的大小不一樣。第一階段是為了先產生大致的輪廓，之後在第二階段的比對會比較準確。故第一階段方塊切的大小較大，第二階段為了處理細節，就切的較小。

第三階段(Holistic Effect Enhancement)就是強化對比度，亮度

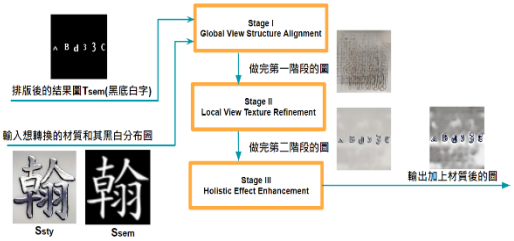

Fig.5 Texture Reformer 的實作流程及例子，其中參數為預設值。

等細節。

而在 Texture Reformer [3]提供的程式裡有提供不少參數可供調整。我們最後決定把 coarse_alph 設為 0、enhance_alpha 設為 0.4、semantic 設為 concat_ds。其他參數則維持原樣。

coarse_alph 代表的意思是第一階段做完的結果和原圖(T^{sem})混和的比例，範圍為 0~1。0 代表維持原圖、1 代表完全採用第一階段做完的結果。我們發現在 T^{sem} 和 S^{sem} 差很多的情況下，背景會嚴重受到 S^{sty} 的影響，如 Fig.6 所示。故我們把 coarse_alph 設為 0，使背景比較乾淨。不過副作用是若背景也想有特定的紋理會無法呈現。

enhance_alpha 為第三階段做完的結果和前一階段產出的結果(T^{sty})混合的比例，範圍為 0~1。0 代表維持前一階段產出的結果、1 代表完全採用第三階段做完的結果。我們發現有時第三階段加強的效果太強了，反而比較不美觀。經過測試，認為設成 0.4 在我們測試的所有 texture 上有比較好的表現。

最後的 semantic 是在運算過程中把 T^{sem} 和 T^{sty} 或 S^{sem} 和 S^{sty} 接起來的方式，細節可參考原論文[3]。程式有提供三個選項，為 add、concat 和 concat_ds。最後選用 concat_ds 的原因為，他雖然比起 concat，背景更容易受到 S^{sty} 的影響。但他能比較好的呈現字的輪廓。而背景看起來很糟這件事，已透過前

面把 coarse_alph 設為 0 解決。故最後採用 concat_ds。

Fig.6 Texture Reformer 的例子，其中參數 enhance_alpha=0.4、semantic=concat_ds。左圖 coarse_alph=0，右圖 coarse_alph=1。

(二) 主要困難與解決之道

1.另尋能夠貼上 texture 的方法

最初在研究 Textlogo Layout 的論文時以為程式本身已經具備了貼上 texture 的功能(Fig.7)，但實際卻只能夠產出白底黑字的輸出。由於論文所使用的方法並未開源，所以我們找了其他替代方案以解決沒有貼上 texture 的問題。

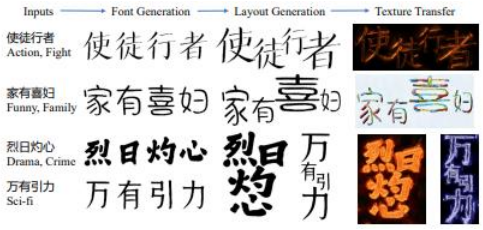

Fig.7 原本預計 TextLogo Layout 的執行結果會如圖中右邊的樣子(Texture Transfer)，但實際上只有對文字做排版(Layout Generation)

最先找到的兩個方法是使用 TET-GAN[4]及 CFITT[5]，但兩者都有一些問題需要解決：TET-GAN 在為一個以上的字貼上 texture 結果通常都不太優秀(Fig.8)，會有字變模糊的情況；相比之下，CFITT 在大多數情況下都能有較好的效果，但排版若太緊湊會有缺字的問題(Fig.8)，另外 CFITT 是用 matlab 實作的，所以無法直接和用 Python 實作的 TextLogo Layout 做串接。和教授討論過後決定先以 CFITT 做為串接對象，但是後來又找到了 CFITT 的改良版——Texture Reformer[3]，不僅是用 Python 實作，產出結果的速度和品質也比 CFITT 好上許多，所以就以 Texture Reformer 作為最終的串接對象。

Fig.8 貼上 texture 的方法比較，由左至右分別為 TET-GAN、CFITT、以及 Texture Reformer

2.使用自己的手寫輸入

在專題研究過程中，曾想過在我們的系統加入使用者自己的手寫輸入，讓使用者能有更彈性的使用方式，也有在網路上找到 Handwrite[6]的插件，這能幫助我們以掃描文件的方式，擷取使用者的手寫輸入並轉換成圖片格式，但很可惜經過 Attr2font[1]後，生成的字體圖片通常不符合理想，這是因為手寫字的屬性需要被正確地給值才能穩定使用，而給值的過程要很久的一段時間，因此我們未在最終的系統裡加入此功能。

四.主要成果與評估

以下將分別展示 Attr2Font、TextLogo Layout 以及 Texture Reformer 三個系統的輸出。

1.Attr2Font

輸出結果如 Fig.9 所示，展現了 3 種使用方式(1).調整部分屬性值(在 Fig.9 裡將 thin 變小、wide 變大、italic 變大)、(2).對 37 種屬性都給一個隨機值、(3).使用其他字體的屬性做模仿，可以發現結果都大致符合我們預期，也可以發現一些特別的事情，在結果(3)可以看到「6」這個字的圖片頂端的橫線，是由來源字體保留下來的，因此來源字體的選擇，也會影響輸出結果。

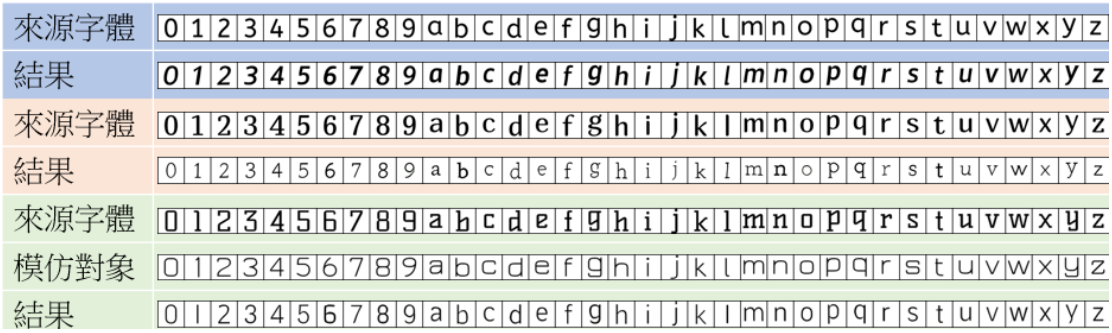

Fig.9 Attr2Font 輸出部分結果

2. TextLogo Layout

輸出結果如 Fig.10 所示，可以看到 TextLogo Layout 不僅在橫向或縱向的排版生成都不會過於單調，也能夠做出較為複雜的排版，如 Fig.10 右邊的排版結果。需要注意的是單字太多或是太長的單字都可能使排版的結果不甚理想，所以我們建議排版時輸入最多 8 個單字且每個單字長度不超過 6 個英文字母。

Fig.10 TextLogo Layout 輸出結果

3. Texture Reformer

Fig.11 為我們把排版完的結果去加上不同的材質。可以發現加上材質後的效果通常會隨著字變大而變得更好。

五. 結語與展望

我們整合了 Attr2Font、TextLogo Layout 以及 Texture Reformer 三篇論文的成果，完成了一個 Text Logo 的生成系統，並且透過此系統改善在設計 Text Logo 需要花費的時間及精力。

未來我們希望能夠再精進生成字體、排版、以及貼上 texture 三個方向的功能和實際結果。

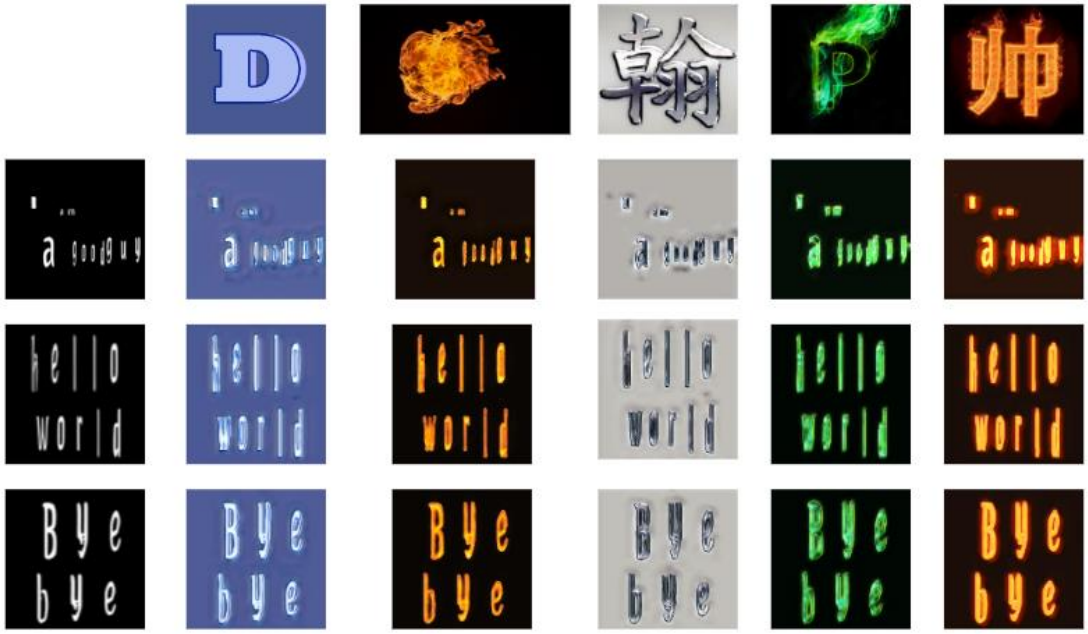

Fig.11 Texture Reformer 的各種輸出結果

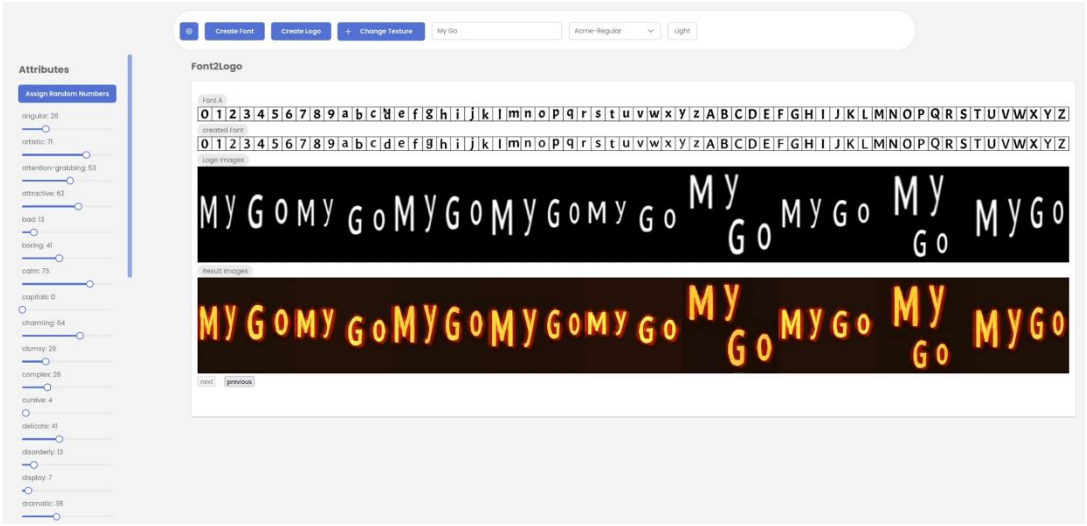

Fig.12 Text Logo Generator 介面展示

六.銘謝

感謝指導教授在專題中的指導和指引研究方向，協助我們完成專題。

七.參考文獻

[1]Wang, Yizhi, Yue Gao, and Zhouhui Lian. "Attribute2font: Creating fonts you want from attributes." *ACM Transactions on Graphics (TOG)* 39.4 (2020): 69-1.

[2]Wang, Yizhi, et al. "Aesthetic text logo synthesis via content-aware layout inferring." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[3]Wang, Zhizhong, et al. "Texture reformer: Towards fast and universal interactive texture transfer." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 3. 2022.

[4]Yang, Shuai, et al. "TET-GAN: Text effects transfer via stylization and destylization." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. No. 01. 2019.

[5]Men, Yifang, et al. "A common framework for interactive texture transfer." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[6]https://www.builtree.org/handwrite/api/sheettopng