

DIP TERM PROJECT REPORT

Group 15

B09902043 沈竑文 B09902041 陳盛緯 B09902070 蔡其翰

使用的PRE-TRAINED MODELS

- https://github.com/irasin/Pytorch_AdalN/
(https://github.com/irasin/Pytorch_AdalN/)
- <https://github.com/naoto0804/pytorch-AdaIN>
(<https://github.com/naoto0804/pytorch-AdaIN>)

單純使用ADAİN MODELS遇到的問題

因為AdaIN採用single-forward pass的作法，一次調整所有pixels導致大多情形下風格的轉移並不夠強烈，造成筆觸的變化無法轉移，或者是用色與理想相差甚遠，只產生了一張重新著色還畫得不怎麼樣的結果。

由於較無頭緒如何透過pre-processing與post-processing解決上色問題，我們決定基於上色較佳但筆觸極差的irasin's model下手去做調整。



PRE-PROCESSING

製造筆觸

起初想法是要透過pre-processing的方式，為content image 「人工加上筆觸」。利用smooth製造較大面積的色塊；利用sharpen彰顯細節而形成較細緻的筆觸，來應對不同style image所要營造的效果。

實驗結果為：smooth後達成的改善較好，因為原圖的結構遭到模糊，造成原先的線條(edges, noise等)擴散為較大的區塊，重新填色後油畫大面積上色的風格的確有轉移到成品上，甚至也能稍微減少光暈的範圍；sharpen達成的改善則有限，甚至還會放大瑕疵與光圈。

(smooth)



(sharpen)



然而smooth改進的效果有限，且使用smooth有其代價：強度過低效果不明顯，強度過高則是導致成品過度模糊，也是另一種層次的失真。

光圈消除

除了筆觸轉移外，我們也希望透過pre-processing處理掉光圈的問題。在組員透過photoshop檢查後發現武士雞周圍顏色和背景有些微不同，因此最初的猜測是「背景顏色不均」。所以我們打算從「顏色」下手，曾經透過調整content image飽和度改變色調、將content image轉灰階移除色彩，然而並無改善。後來也嘗試過對style image做處理，像是通過low pass filter / high pass filter，但也都有各自的問題。最後也嘗試透過單純柔邊、甚至使用photoshop來手動做自己任意想做的pre-processing，很遺憾都沒有成功處理掉光圈。

這些作法會失敗的原因，無非是我們當時並沒有想清楚光圈的成因，使用沒有改善癥結點的方法處理。

以下讓我們花一些篇幅來敘述，我們透過觀察與試驗，得出光圈的成因與猜測。



光圈

要分析光圈產生的原理，首先得回想AdaIN進行style transfer的步驟。AdaIN主要分成三部分，首先從pre-trained的VGG19中的不同層取出風格、內容的feature channels，再經過normalization將content的各channels之平均數、標準差對齊style。最終一個decoder，也就是將VGG19反過來做的generative network生成風格轉移的圖片結果。該decoder是經由training所得出的。我們猜測光圈產生的原因，可能發生於這三步驟中的任一階段，並以武士雞作為討論案例，列出幾個可能原因。

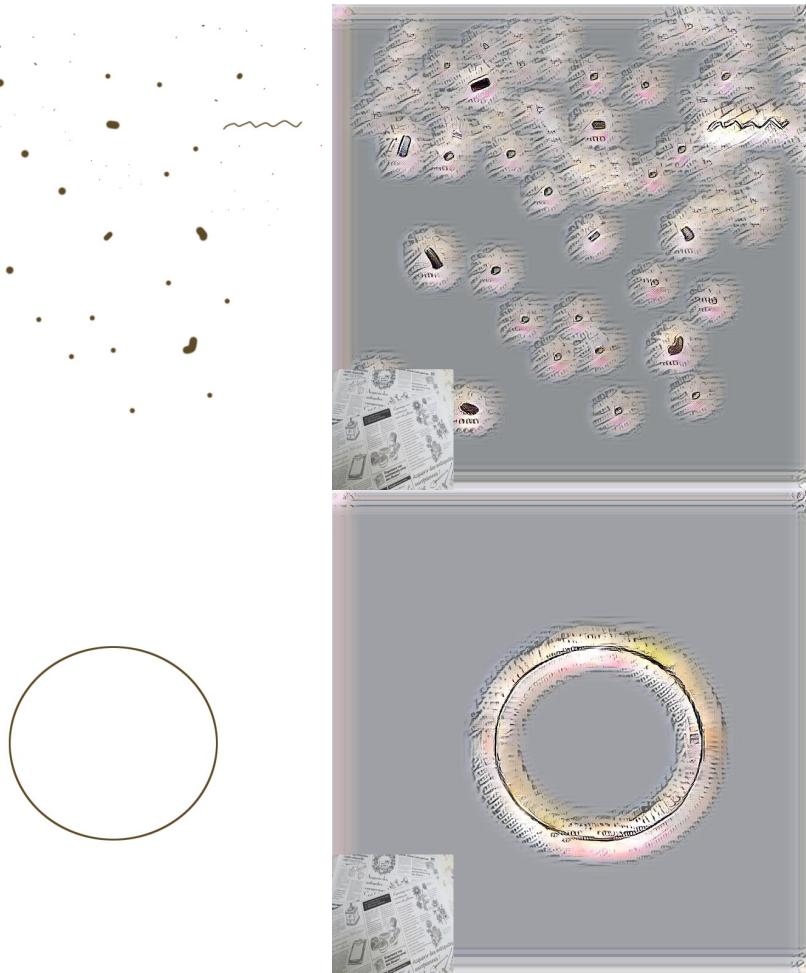
1. 兩塊大單一色塊間的edge的變化過於急遽，這理由正確，但這應該是結果而不是成因。我們實驗時將content模糊化再進行style transfer後光圈確實減少，可以推論 \times 模糊化使得這兩個色塊間的變化較平滑，才有減少光圈的效果。然而在湖景照或其他日常生活照片中，各種物件的邊緣也是變化相當急劇，卻沒有產生明顯光暈。或許是因為這類圖片不像武士雞面積大且顏色單調，而是由多變的小色塊組成，所以光圈小、並因為顏色與風格一致而看不太出來。
2. VGG19沒有學到如何提取，或decoder沒有學到如何生成這種大面積單一顏色的圖片。關於這個理由是否正確，在demo前我們沒有定論，但是報告時有其他組別將train decoder的dataset更換為動漫圖片，仍然產生明顯光圈，所以這並不是主要原因。
3. 因為VGG19是convolution neural network。在AdaIN中，提取feature及生成圖片時使用的VGG19由大量convolution layer組成，而我們猜測如同上課時教的將sinc等kernel拿來做convolution會產生ringing effect，AdaIN使用的

VGG19/decoder在training時所學習到的convolution

kernel應該是類似具有ripple的函數，導致提取feature時和生成結果時在邊緣處產生了風格(光圈)（也就是本光圈也是被轉換過來的風格的一部分，但是我們不想看到它），且剛好在武士雞這張變化較少的圖片上看起來很明顯，而在湖景這種細碎的圖片上被轉移的風格能夠連在一起而看不出來。不過這個問題並不好解決，雖然前處理 / 後處理能夠降低光圈影響，但邊緣與想要放上風格的地方不可分割，根本上還是要改動模型架構才有可能解決。

理解AdaIN的運作模式

為了更好的將風格轉移到內容圖片上，我們首先研究了AdaIN進行風格轉移的工作方式，並進行了一些實驗：



上圖為將一個純白背景，分佈零散斑點的content，以中右下角的報紙style進行風格轉移之結果。可以看出，AdaIN只會在輪廓的邊緣做風格轉移。此結論可以由下圖驗證，確實只在圓圈周遭有如報紙的文字。由此可以推論，如果沒有附著點，AdaIN便無法運作，因此若想要AdaIN的效果好，必須在content中加入有足夠的附著點（如加入noise等方法）。這也呼應武士雞身後大面積純色背景為什麼沒有任何風格，而

武士雞光圈的存在，即是因為AdaIN在邊界處進行了風格轉換；若除去這效果，風格轉換便無法進行，所以解法只能對背景做調整(ex.加入noise)，使得風格轉換產生更多效果，與光圈連接在一起後便難以看出。另外前面提及的smoothing之所以有效，則是因為其提供更大面積的附著點，因此產生更多的style效果。

MODEL TRAINING的影響

在尋找如何實作adaIN時，我們發現雖然同樣使用AdaIN搭配pre-trained VGG19提取feature，但是因為training的dataset或方法上的不同，以及使用提取feature的layer不同，產生的結果也有很大的差異。舉例來說，Keras官方提供的範例

(<https://keras.io/examples/generative/adain/>) 就因為只使用COCO相當小的subset並只train 30 epoch，而造成相當差的結果。而原始論文使用的是完整的MSCOCO dataset，並且訓練160000個iteration，因此效果好上不少。

各司其職一串接不同MODEL

基於以上原因，我們開始搜尋github上效果較好的model，希望能找到唯一解。其中較多star的兩個model分別為irasin及naoto0804的model。irasin的model可讓上色色調接近style image，但筆觸完全無法還原，使結果僅僅像是從原圖結構重新上色的結果；naoto0804的model更容易產生類似於style image的contour，但用色上並不是那麼精準，而且成品常有大幅度的扭曲，不夠精美。因為各個model都有不同程度的缺陷，也因為沒有單一model能夠做好，因此我們想，如果我們讓他們各自負責自己厲害的地方呢？另外，先前我們提及AdaIN只在輪廓邊緣進行風格變換，現在我們做多次的AdaIN，就會使得輪廓邊緣隨著每次style transfer稍微擴散，導致風格轉換的強度上升，也符合我們的目標。

因此，我們設計出這樣的流程：首先，將原style, content跑過naoto0804的model產生擁有理想輪廓的中間結果，再使用irasin的model將中間結果套上原始的style，得到將輪廓重新上色的最終產物。就像是畫家的創作過程一般，先勾勒骨架與結構，再進行上色，這個想法雖然直觀，但卻讓我們收

獲了突破性的成效。

(左：我們的方法，右：單純的one-pass)



由圖可見，用色極度符合原圖，對筆觸的轉移獲得了前所未有的提升，油畫的風格顯露無遺。

將架構融合PRE-PROCESSING

由於現在風格轉換被分成兩個階段，因此我們可以在最一開始以及兩個phase之間插入一些對圖片的處理，如smooth, high-pass, low-pass, log-transformation等，進一步將結果往我們想要的效果做調整。

例如我們嘗試過的作法是：餵進naoto0804的model前先讓style image經過一次high-pass filter和log transformation，把style image的顏色去掉單純留下輪廓，目的是為了讓產生的中間產物偏向黑白，產生只獲得了style輪廓但沒有受到顏色影響的武士雞。接著把從naoto08084 model中間結果當作餵進irasin model的content，而原始的style image作為這輪的style image輸入，最終得到上色準確且輪廓更多的結果。

結果

這裡將會比較最初單純使用irasin的model得到的成果（左圖），與使用我們作法得到的結果（右圖）。

武士雞



可見武士雞脖子處多出了一塊一塊、有如紅磚般的格狀紋路，成為一架「磚雞」。

Noised武士雞



這是有先加入noise產出的效果，可以看出我們的方法在背景產生了更像磚塊的塊狀紋理，連左下角的style image磚牆都快看不到了。之後的結果並沒有插入noise，是為了更單純的測試這方法的有效性。



圖片下方的樹風格轉移地最為明顯，顯現出原圖有稜有角、轉折強烈的幾何風格。上色方式也更像style image般乾淨、俐落。



在衣服處最為明顯，彰顯出style image粗大的筆觸。



有些時候我們的作法並不會一定比較好，像這批成品單純one pass的色調會更接近style image明亮的風格。

湖畔風景圖



可見湖邊兩側樹葉的輪廓更為抽象、奔放，接近星夜原畫迷幻中略帶扭曲的風格。



可見湖面多出了更多細小文字，而背景的樹葉也變得更加方格狀，更有報紙那種「井然有序的分區搭配文字密密麻麻的雜亂感」的獨特反差氛圍。



可見湖面的波光立體感變得更加強烈



從圖片各物體的邊界看來，多出了肖像畫中粗粗黑黑的邊線，更加強調了物體的輪廓

我們作法的優勢與結論

要產生更好的風格轉移結果，調整訓練的model架構或是直接換一個model會是更有效且更泛用的解法，但我們仍認為我們的做法有其價值。首先，AdaIN強調的是他的速度，而我們的做法並沒有增加太多額外的時間，跑兩次AdaIN也不會比跑一次多花太多時間，仍可以壓在real-time的影片生成時間內。若是改用其他model，或許可能得到更好的風格轉移效果，但是就失去AdaIN的速度優勢。其次，訓練AdaIN模型對於運算資源有很大的要求，光是MSCOCO + WiKiart合起來就將近40GB，VGG19又是相當深的CNN，為了調整一張圖片而重新訓練模型並不合理。因此我們的方法利用既有的model，結合各自的長處，設法加強style transfer的力道，這是一種較經濟的做法，也代表未來或許可以研究如何使用不同的dataset、不同的提取feature方式訓練幾個較小的模型，再將結果串接起來取得更好的效果，如此不但運算量下降，也能加入更多人為調整的參數，能夠更好的產生預期中style transfer的效果。

影片 STYLE TRANSFER

首先來介紹我們是如何轉換影片的：

1. 輸入影片
2. 用cv2的函式將影片分成多個幀，並將這些幀放入名為 frames 的資料夾

3. 輸入style image

4. 將 frames 裡的所有圖片轉移成style image的風格，並依前後順序將它們放入名為 stylized_frames 的資料夾

5. 利用 ffmpeg 將 stylized_frames 的圖片組合起來成影片檔並輸出

另外我們透過改變第四部份的轉移指令來決定是要用naoto的model還是irasin的model轉移影片。

由前面我們已知道naoto的model對於風格筆觸的轉移效果較好，而irasin的model則是善於轉移材質、顏色主題等，因此我們可視style image的視覺主效果來決定要用哪個model。

像是以《The muse》該風格為例，其呈現的主題是狂野的抽象筆觸，因此用naoto的效果較好。而若是以磁磚風格的話，其呈現重點是在於磁磚的紋理，所以用irasin的model能更好地帶出風格的主題。

透過前面的結論我們也知道善用兩種model可使轉移風格有更好的呈現效果，然而因為我們需將轉移後的圖片組合起來做影片，而轉移過一次的關係相鄰的幀之間已有不少的內容差異，若再轉移一次之後該差距會被拉大，使得影片有嚴重的閃爍效果。

除了試用兩次的風格轉移之外，我們還有嘗試對影片做post-processing，像是在 result/videos 裡面的 news1_irasin_denoise，因為雜訊的關係使得影片有不少的閃爍長條，而我們對其做降噪之後那些條狀glitch的數量下降了不少，但伴隨而之的是風格圖片所想呈現的報紙文字感也因此降低（成果見slide）。