中華民國第 61 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

第三名

052508

利用深度學習將黑白影片色彩化

學校名稱：國立臺南第一高級中學

作者：	指導老師：
高二 莊庭瑋 高二 簡敦賢 高二 鄭仲耘	顏永進

關鍵詞：深度學習、影片色彩化、影像處理技術

摘要

本研究旨在解決以往無法將影片色彩化的問題，設計方法分為兩大部分：首先，我們在基礎的生成對抗網路上加進了多個模組如：殘差模組以及特徵提取模組，我們實作多種不同組合的生成對抗網路，並比較其成效。其中，我們利用 COCO dataset 來訓練我們的的基礎模型。本研究以風景影片為主，因為風景影片的震盪和變化較小。在進入第二階段前，我們利用 Kaggle 風景圖片資料庫來微調最優的模型。而在第二部分，我們發現生成出的影片會有色彩不連續性的問題，於是我們提出了三套方案來提高影片整體品質來抵銷模型在前饋過程中所產生的不確定性以及隨機性。分別為 H.264 編碼技術，ORB 預測個別幀，以及 HSV 提高色相穩定度。

壹、研究動機

此主題是我們閱讀 pix2pix 論文時所發想到的，相對於其他影像轉換問題，黑白圖片色彩化的難度更高，因此我們想針對此圖像轉換問題進行研究。首先，此問題不僅只是風格轉換問題(如黑夜轉為白天)，更包含圖像識別問題，須「合理」並依照圖片中的物體進行上色。其次，由於特定圖像之顏色並不固定，一種物體可能有兩種以上「合理」的顏色，因此，模型對生成之「穩定度」也極為重要，意思就是針對兩張類似圖片，應該要預測出類似顏色；但顧及穩定度之同時，卻也要顧及生成色彩的「豐富度」，即生成圖片之彩度。

而本研究受到近年來發展迅速的「DeepFake」領域啟發，想利用深度學習技術來建構圖像轉換模型，我希望不僅藉著本研究能探討模型中各種變因對生成結果的影響，也希望能將本研究延伸並推廣至更通用的模型，不只適用於色彩化問題，更能應用到影像轉換的其他領域。而本研究另一部份則是改善影片不連續性部分，我們希望建置幾種改善方法。最終我們希望能改善整體色彩化影片的真實性。

貳、研究目的

一、 影像色彩化。

二、 改善生成影片不連續性問題。

三、 分析各實作結果及成效差異。

參、研究設備及器材

一、 軟體環境：Python 3.7、Jupyter Notebook、PyTorch 1.4、Pytorch-Ignite 0.4.2、OpenCV、Tensorboard 2.4

二、 硬體規格：

(一) 中央處理器(CPU)：Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz

(二) 記憶體：128.0 GB

(三) 圖形處理器(GPU)：2 個 Quadro RTX 5000 (GPU 記憶體 16 GB)

(四) 作業系統：Linux (Ubuntu 18.04)

三、 資料來源：coco(2017)、Kaggle Landscape Pictures

肆、研究過程或方法

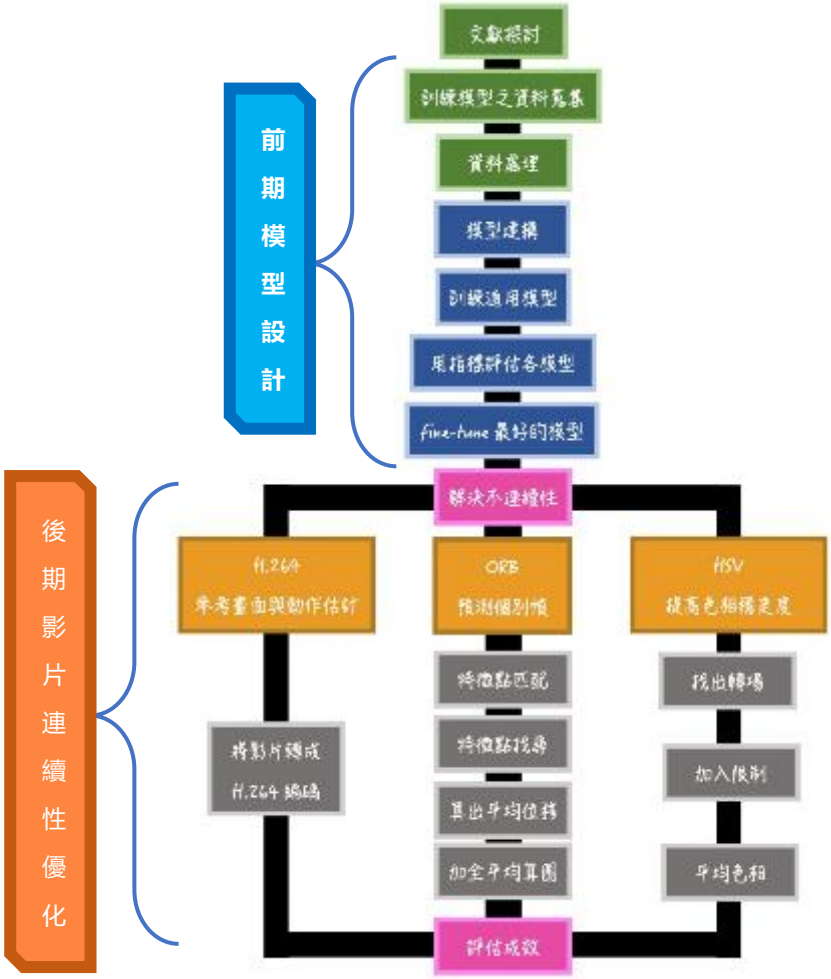


圖 1、實驗流程圖

上圖為我們研究之大致流程圖，由文獻方面的探討和瞭解開始，事先了解過去論文中提出的各個模型架構，使我們對於此領域能有深厚的知識基礎，接下來就是訓練資料部分的收集了，我們選定了 COCO 資料集，將資料處理後，開始設計建構模型，訓練完各個模型後，我們選出最好的模型進行 fine-tune。隨後，就是解決不連續性的部分。我們提出了三個方法：一、H.264 參考畫面與動作估計 二、ORB 預測個別幀 三、HSV 提高色相穩定度。最後我們對各方法進行評估及比較。

一、 文獻探討

(一) 模型設計部分：生成對抗網路(GAN, Generative Adversarial Network)

GAN 是無監督式學習的一種方法，原始目標是學習從潛在空間取隨機變量 z 經由類神經網路生成 y。其主要包含兩個部分，分別是生成器與判別器，利用梯度下降，同步訓練兩類神經網路。而隨機變量輸入生成器後得到圖片，然後再輸入判別器；判別器是二元分類器，圖片經過判別器後會輸出數值以進行區別輸入為真實抑或生成。判別器盡量從真實圖片與生成圖片中判別出差異處，而生成器的目標則盡可能使生成圖片與真實圖片一致，以騙過判別器。 兩者持續優化自身參數，理想情況下，最後生成圖片會接近真實圖片。利用兩者對抗性進行訓練，類神經網路最終能學習將隨機變量映射至真實資料集之分布。

而下**圖 2** 即為 GAN 的原始 objective function(損失函數相反)。以此訓練，判別器盡可能將真實圖片判別為 1，而將生成圖片判別為 0，以區別其中差異。相反生成器目標則會盡可能生成可以被判別器判別為 1 的圖片。簡言之，即是控制生成器內部參數使生成圖片分布向真實圖片分布靠近。

$$min_{\theta_G} max_{\theta_D} V(G, D) = E_{x \sim P_{data}(x)}\left[\log(D(x))\right] + E_{z \sim P_z(z)}\left[\log(1 - D(G(z)))\right]$$

圖 2、GAN 的 objective function 數學式

(二) 模型設計部分：conditional GAN(c-GAN)與 pix2pix

c-GAN 是針對原始 GAN 增加條件限制，使得生成之結果不僅需要逼真，還需達成設立之條件。其訓練方式與一般的 GAN 雷同，主要差別之處有兩部分。首先為生成器之輸入的改動，是所謂的「condition」，可以是類別抑或圖片，而非僅是隨機變量。其次則為判別器之更動，判別器的輸入改為生成器之輸出加上條件，訓練判別器從條件及圖片判別真偽。

而 pix2pix 也是一種 c-GAN，但其不僅限制生成器與判別器輸入，亦限制生成器之輸出。它利用 L1 Loss 或 Perceptual Loss 以限制生成器之輸出，使得生成圖片中的「內容(content)」與目標圖片一致。

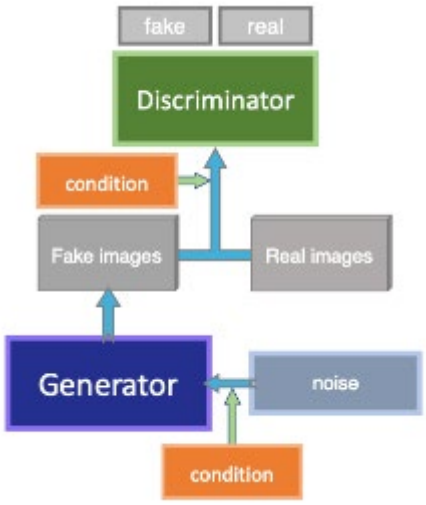


圖 3、c-GAN

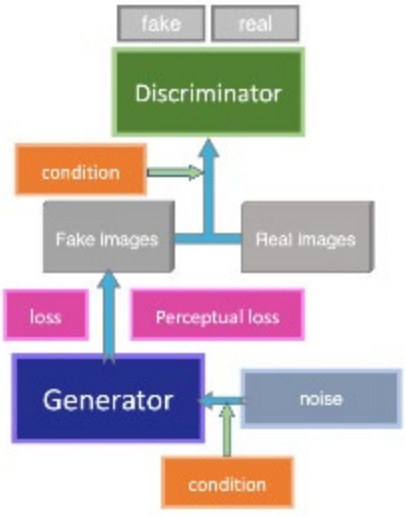


圖 4、pix2pix

(三) 模型設計部分：深度殘差網路(ResNet, Deep residual network)與 U-Net

ResNet 和 U-Net 此二者皆運用到跳躍連接(skip-connection)。跳躍連接即是將前端模組的輸出保留至特定模組輸出後，兩者相加抑或拼接。如此一來，當圖片在模型中進行計算時，後方模組也能經由跳躍連接獲取到前端的訊息。而跳躍連接的公式為：f（x)=f(x)+x，將模組之輸出與輸入合併。

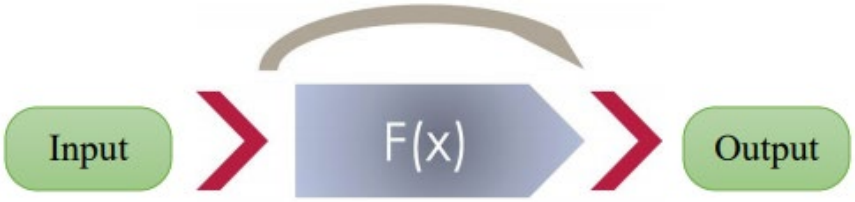


圖 5、跳躍連接圖

而 ResNet 與 U-Net 的差異，基本上就是長短與連接方式的差異，對於 ResNet 而言，其本體是「短」跳躍連接架構；而對於 U-Net，其模型分為下採樣層、瓶頸層、上採樣層，利用「長」跳躍連接將下採樣層的輸出保留，與上採樣層的輸入合併。

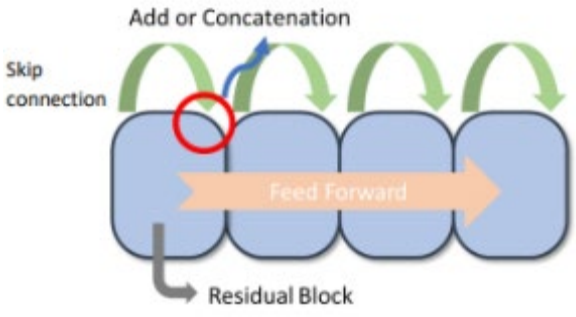


圖 6、ResNet

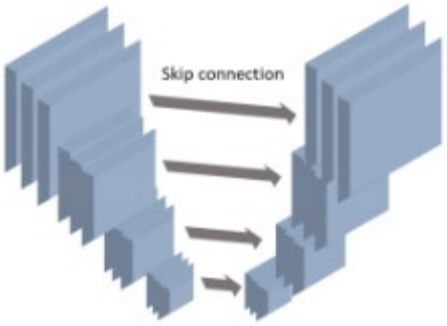


圖 7、U-Net

(四) 影片連續性優化部分：H.264 編碼技術

H.264 可大致分為對四種冗餘方面的改良，分別為時間、空間、熵編碼以及感官冗餘。而本研究著重於時間冗餘改良的部分。此改良方法分為兩部分，分別為動作估計以及多重參考畫面。

1. **時間冗餘**：在一個高幀率的影片中，每一幀與其相鄰幀之間的影像是非常相似的，往往只有細微差距。如果儲存了所有幀的資料，這就造成了時間冗餘問題。
2. **動作估計**：H.264 的可變動區塊有七種(16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)，可以依據影片的動作或材質的複雜程度，彈性選擇碼率，也就是失真最佳的區塊類型。此外，H.264 提供 1/4 像素的精確度搜尋，以提高動作補償的準確度。
3. **多重參考畫面**：畫面共可分為三種：I、P、B 畫面。

(1) I-畫面(Intra-pictures)

採用與 JPEG 極為相似的畫面內編碼，壓縮時不參考其他畫面，它可以做為隨機讀取(如快轉、快回、段落選取)的依據，以及 P-畫面與 B-畫面的參考畫面。

(2) P-畫面(Predicted pictures)

利用過去最近的多張 I-畫面或 P-畫面做為向前預測(forward prediction)的參考畫面，再結合動作估計與轉換編碼進行壓縮，它可以做為其他畫面的參考畫面，但不能做為隨機讀取的依據。

(3) B-畫面(Bidirectional predictive)

編碼時可以參考前後最近的 I-畫面或 P-畫面，再結合動作估計與轉換編碼進行壓縮，不能做為隨機讀取的依據。

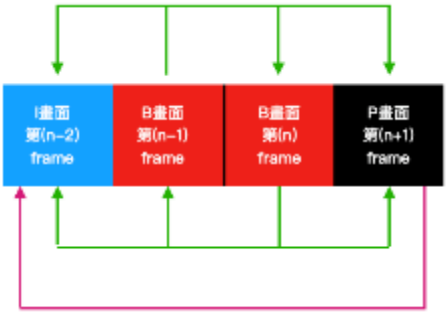


圖 8、多重參考畫面示意圖

(五) 影片連續性優化部分：HSV 色彩空間

HSV 色彩空間是將 RGB 色彩在圓柱座標系中的一種表示法，由色相、飽和度、明度組成。

1. **色相(H)**：控制我們平常所稱的顏色，例如藍色、綠色等。
2. **飽和度(S)**：色彩純度，越高表示顏色越飽和，越低表示越淡。範圍 0%~100%。

3. **明度(V)**：色彩明亮程度。範圍 0%~100%。

(六) ORB(Oriented FAST and Rotated BRIEF)

ORB 是一種找尋特徵點的演算法，具有尺度和旋轉不變性，對於噪聲及其透視變換也具有不變性，使 ORB 在進行特徵描述時的應用場景十分廣泛。此特徵檢測演算法是在 FAST 特徵檢測和 BRIEF 特徵描述子的基礎上提出來的。FAST 關鍵點檢測能較快地找出特徵點，利用隨機取點並以該點為中心畫一個圓過十六個像素點，如果此點的灰度值比其周圍領域內足夠多的灰度值大或者小，則該點可能為特徵點。不過利用此演算法所的到的特徵點沒有方向，不滿足尺度變化，於是效仿 SIFT 演算法，建立尺度影像金字塔，通過在不同尺度下的影像中提取特徵點達到滿足尺度變化的效果。找到關鍵點後，需要確立關鍵點的方向，首先將特徵點的鄰域範圍視為一個區域，並算出其質心位置，將特徵點與其連線，即是此特徵點的方向。

BRIEF 將 FAST 找到的特徵點轉換為二進位特徵向量。二進位特徵向量是僅包含 1 和 0 的特徵向量。簡而言之，每個關鍵點皆由特徵向量描述，此特徵向量是 128~512 位字符串。透過此方法可以較容易的對這些特徵點進行匹配。

二、 訓練模型之資料收集與資料處理

(一) 訓練模型之資料收集：

本研究模型設計的前部分是通用模型之訓練，我們選擇以 COCO 資料集訓練，是因為各模型之「整體」成效，並不侷限於特定種類之生成。而選擇 COCO 的原因在於其資料集的圖片場景較廣，比起其他資料集如 ImageNet，圖片中的單一物體佔據的比例較小，因此較適合色彩化模型。我們先從 COCO 官網下載資料，並從中過濾出黑白圖片。最後分成訓練集 118060 張圖片，驗證集 4990 張圖片，測試集 40640 張圖片。而後面部分則是針對於風景類別的生成，我選擇了 Kaggle 風景資料集， 與前部分一樣過濾出黑白圖片，並將其分為 3337、417、418 三個部分。

(二) 資料處理：

我們使用 PyTorch 函式庫建構資料處理程式，從圖片所在路徑讀入圖片後，進行資料擴增。針對訓練資料，先將圖片轉換成特定大小，然後進行隨機水平翻轉、隨機角度旋轉、 隨機色彩變換 (亮度、對比度)。最後將圖片標準化至-1~1 之間，並回傳黑白圖片與彩色圖片。

```
class Data(Dataset):
    def __init__(self, fileList, mode, size):
        super().__init__()
        ...
        self.__gray = Grayscale(3)
        self.__toTensor = ToTensor()
        self.__norm = Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        if mode == 'train':
            self.transform = Compose([
                Resize(size),
                ColorJitter(brightness=0.15, contrast=0.15),
                RandomHorizontalFlip(),
                RandomRotation(15),
            ])
        else:
            self.transform = Compose([
                Resize(size),
            ])
    def __len__(self):
        return self.length
    def __getitem__(self, idx):
        rgb = Image.open(self.fileList[idx]).convert('RGB')
        rgb = self.transform(rgb)
        bw = self.__gray(rgb)
        return {
            'rgb': self.__norm(self.__toTensor(rgb)),
            'bw': self.__norm(self.__toTensor(bw))
        }
```

圖 9、資料處理程式

三、 模型建構與模型訓練

(一) 模型建構

1. **模型概略**：本研究建立了四種模型當作實驗組，並以最基本的 U-Net 當作 baseline 模型與對照組模型。判別器皆為相同，在本研究中視為控制變因，而改動處只有生成器。

	生成器	判別器
對照組： basic U-Net (baseline)	Encoder：16x 縮小 Decoder：16x 放大	4 層含殘差模組的 PatchGAN
實驗組： Residual Network (RUNet)	將內部**基本模組**改為 **殘差模組**(相對對照組)	同上
實驗組： Feature-based Network (FUNet)	將模型內加入本研究 提出的**特徵提取模組** (相對對照組)	同上
實驗組： Enhanced feature-based	前兩個實驗組合併模 型	同上

Network (EFUNet)		
實驗組： Enhanced feature-based Network(large) (EFUNet-plus)	增加前一模型的深度	同上

2. 模型中的各模組：

(1) 基本模組

下圖 10 為基本模組程式，其架構是數層卷積層、BatchNormalization 層、LeakyReLU activation function、與一層 Dropout 層。

```
class BasicBlock(nn.Module):
    def __init__(
        self,
        ...
    ):
        super(BasicBlock, self).__init__()
        ... # making layers
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        # simple feed forward
        out = self.activ(self.norm1(self.conv1(x)))
        out = self.activ(self.norm2(self.conv2(out)))
        out = self.activ(self.norm3(self.conv3(out)))
        if self.dropout is not None:
            out = self.dropout(out)
            out = self.output_conv(out)
        return out
```

圖 10、基本模組程式圖

(2) 殘差模組

為了與基本模組參數量一致，其架構與前者大致相同。但加入跳躍連接，將輸入與輸出拼接。

```
class ResidualBasicBlock(nn.Module):
    def __init__(
        self,
        ...
    ):
        super(ResidualBasicBlock, self).__init__()
        ... # making layers
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        out = self.activ(self.norm1(self.conv1(x)))
        out = self.norm2(self.conv2(out))
        if self.stride > 1:
            x = self.avg_pool(x)
        # skip connection
        out = self.norm3(self.conv3(torch.cat([x, out], dim=1)))
        if self.dropout is not None:
            out = self.dropout(out)
            out = self.output_conv(out)
            out = self.output_norm(out)
        out = self.activ(out)
        return out
```

圖 11、殘差模組程式圖

(3) 特徵提取模組

在本研究設計的生成器裡，每一層皆包含數個相同輸入大小(圖片大小)的模組(基本模組抑或殘差模組)。普遍作法即是直接將張量依序輸入數個模組後得到輸出，但我們認為如此作法較無法保留學習到的特徵。因此設計了此「特徵提取」模組。本模組的發想是受到跳躍連接的啟發，此模組將各模組輸出的結果先保留以提取出「特徵」，且同時依序向前輸入每個模組，最終得到每個模組的保留的輸出(特徵)之後，對張量通道拼接，後輸入 1x1 的卷積層降維。如此一來，每一模組的輸出皆能藉由此設計的「跳躍連接」而獲得充分利用。利用此模組，我們希望能讓模型學習到複雜細節的同時，也能保有每個模組獨立的輸出特徵。

```
def __merge(self, x: torch.Tensor, layers: nn.ModuleList, bottle: nn.Module):
        outs = []
        for l in layers:
            x = l(x)
            outs.append(x) # obtain the output of each module
        out = torch.cat(outs, dim=1) # concatenate the output
        out = bottle(out) # bottleneck
        return out
```

圖 12、特徵提取模組程式圖

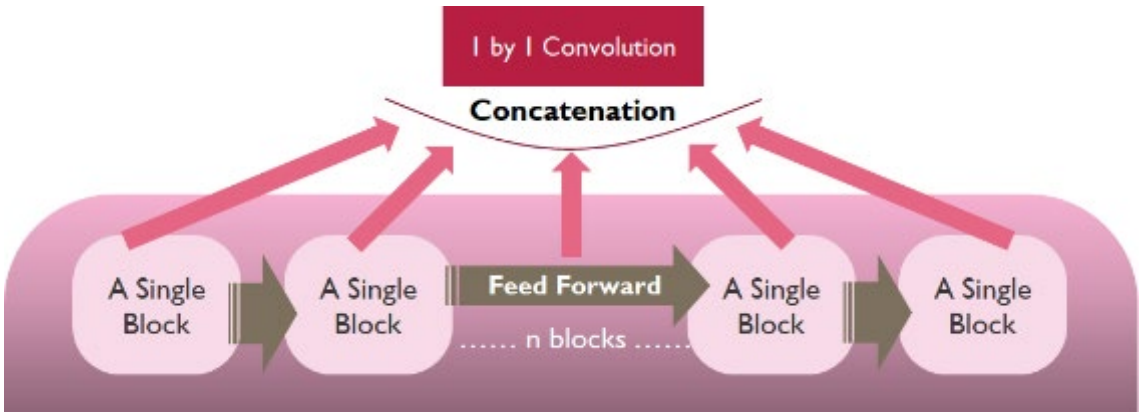


圖 13、特徵提取模組架構圖

3. 各模型之本體設計：

(1) 生成器：baseline(對照組)

本模型為所有模型基礎。可見下**圖 14**，左半部為 encoder 右半部為 decoder。兩者內部又有各 4 層結構，每一層含數個相同 scale(張量大小)的基本模組。當圖片輸入模型時，分成兩個部分，第一部份是維持圖片長寬並輸入含 8 個基本模組的層，第二部分則是輸入 U-Net，從 encoder 輸出長寬小 16 倍的圖片與前幾層之輸出(下圖最上部分)，再輸入 decoder，還原成與輸入圖片大小一樣的張量，最後輸出結果與第一部分(8 個基本模組)合併。經過 Merge block(也是一個基本模組)後，再經過 tanh 函數得到色彩化圖片。而此模型不同層的基本模組數量不同，分別為 encoder：(8, 4, 2, 1)，decoder：(1 ,2, 4, 8)，如此設計的原因是隨著圖片大小減小而通道數增加，則參數量則會大量增加。因此本模型設計成上寬下窄的設計，使越底部的層模組數量越少，以減少運算時記憶體的佔用。

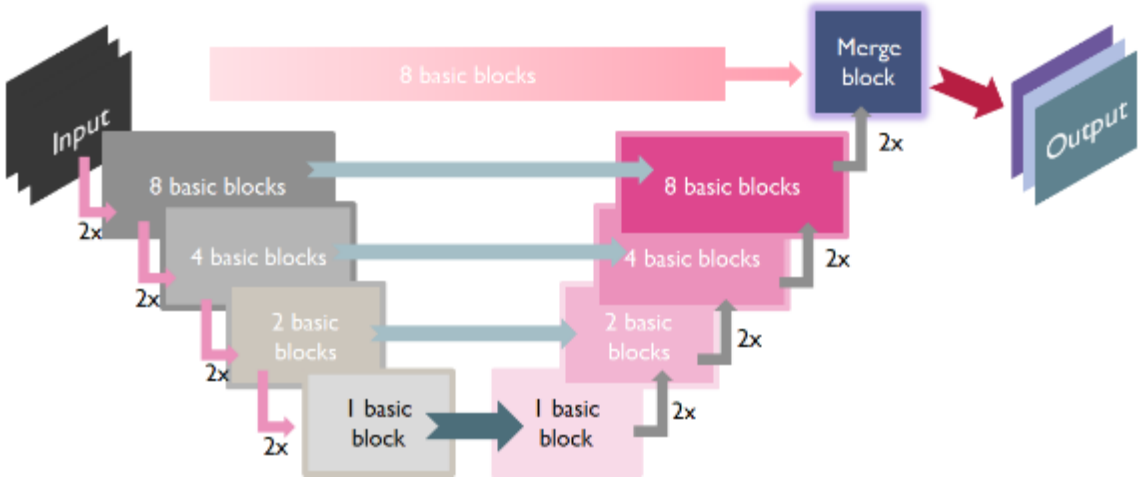


圖 14、baseline 架構圖

(2) 生成器：RUNet(實驗組)

本模型與 baseline 相差在於將所有**基本模組**改為**殘差模組**，其餘相同。

(3) 生成器：FUNet (實驗組)

本模型與 baseline 相差在於將模組與模組之間的傳統 feed forward 運算方式加入**特徵提取模組**，其餘相同。

(4) 生成器：EFUNet(實驗組)與 EFUNet-plus(實驗組)

前者為整合 RUNet 與 FUNet，利用**殘差模組**為模型基本結構，而**特徵提取模組**為每一層中模組與模組的運算方式。

後者則是更改了前者的兩個部分以增大模型：一、增加模型中的卷積層通道數；二、使 U-Net 的 encoder 最底層的輸出再往下縮小二倍，decoder 也是相應改變。如此希望增加模型深度，以期增加生成圖片的真實度。

(5) 判別器：PatchGAN(控制變因)

本模型為包含**四層殘差模組**的模型(判別器)。圖片輸入模型後先經過一層卷積層將通道數從 6(圖片加條件的總通道數)增加到 8，再經過四層殘差模組，而每一層中的卷積層通道數從 8 往上依序乘以 2 倍(8, 16, 32, 64, 128)，而同時將圖片下採樣 2 倍，最終輸入 1x1 的卷積層使張量的通道數變回 1。本模型輸出的並非是一個數值，而是一個類似圖片的矩陣。參考 pix2pix 論文中 PatchGAN 的說明，這樣的設計能以圖片的區域性特徵(如材質、樣式)為判別標準，而非統合成一個數字，使得生成器能依據圖片中的不同區域進行參數優化。這樣也另外有個優點，即為輸入無須為特定大小(長寬)，因為組成模型的卷積神經網路並沒有受到圖片大小的限制。

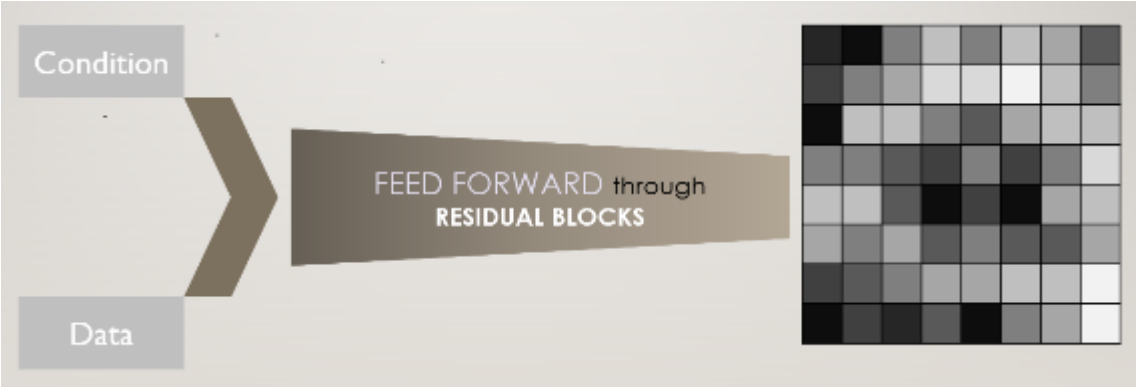


圖 15、PatchGAN 流程圖

```
class PatchCritic(nn.Module):
    def __init__(
        self,
        ...
    ):
        super(PatchCritic, self).__init__()
        ...
        self.block = nn.Sequential(
            nn.Conv2d(6, inplane_base, 3, stride=2, padding=1),
            norm_layer(inplane_base),
            self.activ(),
        )
        self.layer1 = layer_block(inplane_base, layer_depth[0], norm_layer,
activation_function, dropout_rate)
        self.down1 = nn.Sequential(
            nn.Conv2d(inplane_base, inplane_base*2, 5, stride=2, padding=2),
            norm_layer(inplane_base*2),
            self.activ(),
        )
        ... making 3 other layers
        self.squeeze = conv1x1(inplane_base*16, out_inplanes)
        ...
```

圖 16、PatchGAN 程式圖

(三) 模型訓練

1. **訓練流程**：先利用 COCO 資料集訓練好五個模型後，接著使用圖片指標評估成效， 然後選擇最佳的模型，並使用風景資料集訓練。
2. **訓練方式**：使用 c-GAN 的訓練方式同步訓練生成器以及判別器。

(1) 生成器的訓練

a) 將黑白圖片輸入生成器得色彩化圖片 。

b) 將色彩化圖片以及黑白圖片拼接並輸入判別器。

c) 利用判別器的輸出計算生成器的 Adversarial loss(Binary Cross Entropy loss)與計算色彩化圖片與真實圖片的 Content loss(本研究使用 L1 loss)。

d) 將 Content loss 乘以 lambda 加上 Adversarial loss。

e) 梯度下降。

其中計算損失值的部分可見下**圖 17**，分為前半部 Adversarial loss 意旨判別器的對抗損失以及後半部的 Content loss，在此我們使用的是 L1 loss。

$$L(G) = -E_{x,z}\left[\log(D(G(x,z)))\right] + \lambda \cdot E_{x,y,z}\left[\| y - G(x,z) \|_1\right]$$

圖 17、生成器的損失函數

(2) 判別器的訓練

當作二元分類器進行訓練，一樣利用 Binary Cross Entropy loss，但目標與生成器相反。

3. **訓練的超參數以及模型細節**：

(1) 訓練 COCO 資料集

優化器(optimizer)	皆為 Adam 優化器，學習速率 0.0002
圖片大小	128*128
批次大小(batch size)	16
資料集的迭代次數(epoch)	25
Content loss 的 lambda	100

(2) 訓練風景資料集

優化器(optimizer)	皆為 Adam 優化器，學習速率 0.0001
圖片大小	128*128
批次大小(batch size)	16
資料集的迭代次數(epoch)	200
Content loss 的 lambda	50

(3) 各模型參數量(生成器)

模型	參數量
baseline	14720640
RUNet	15695648
FUNet	15007360
EFUNet	15982368
EFUNet-plus	49555232

四、 改善影片不連續性問題

我們共提出了三種方法來改善影片不連續的問題。

(一) H.264 編碼技術算法

由於模型生成影片時對於每一幀之預測並不一定會連貫，導致色彩不連續。基於此，我們決定使用編碼技術，因其有一個流程為參考畫面以及動作估計，這可使相鄰的圖片互相進行微調，而達到解決效果。此外，此方法的最大特色為不會影響太大的整體影片相似度。我們使用 FFmpeg 來實作。

速率控制：指決定多少位元將被用於每個畫格的方法。這將決定影片的檔案大小且品質高低，在此我們使用「Preset」來調控速度。Preset 主要是利用 Reference frames 多寡來進行速度的區分。Reference frames 也就是一張圖片生成前所參考的影片幀數，包括 I、P frames。在本次的實作，我們選擇了九種編碼速度由慢到快分別為 Placebo，Veryslow，Slower，Slow，Medium，Fast，Faster，Veryfast 以及 Ultrafast(上述名字為 FFmpeg 原有的設定)。

速度名稱	Reference frames	B-frames
Placebo	16 幀	16 幀

Veryslow	16 幀	8 幀
Slower	8 幀	3 幀
Slow	5 幀	3 幀
Medium	3 幀	3 幀
Fast	2 幀	3 幀
Faster	2 幀	3 幀
Veryfast	1 幀	3 幀
Ultrafast	1 幀	0 幀

(二) HSV 提高色相穩定度

因為不同場景的色調變化極大，導致不適用於我們提出之改善影片不連續問題的方法。因此必須先設計出演算法找出影片中的轉場處，也是因為我們發現轉場時色調變化程度極大，設計以下三種演算法，以「偵測」出轉場的時機。(以下方法採用的測試影片為 EFUNet-plus 所生成，取前 10000、30000 幀作為範例，共有 27、71 處轉場)

1. **轉場偵測方法**

(1) 絕對值方法步驟

a) 將影片中前後幀的 RGB 像素值進行相減取絕對值，再除以長寬，得到 Rn、Gn、Bn 三組分別代表 RGB 三層第 n 幀與第 n+1 幀的像素差異值。(以 R 示範)

$$R_n = \frac{\left| r_{n+1} - r_n \right|}{h \cdot d}$$

b) 將前後兩像素差異值相減得到得到 Rdn、Gdn、Bdn。(以 R 示範)

$$Rd_n = R_{n+1} - R_n$$

c) 將三者平方相加後除以 3 再開根號，得到 Dn。

$$D_n = \sqrt{\frac{Rd_n^2 + Gd_n^2 + Bd_n^2}{3}}$$

d) 將 Dn 對 n(幀數)作圖。

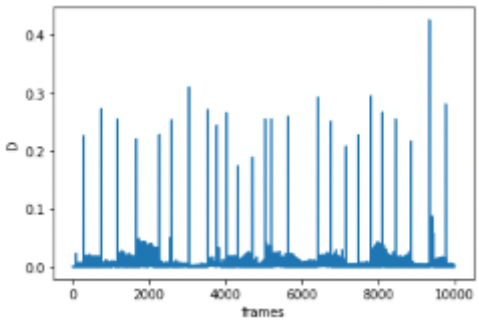


圖 18、10000 幀

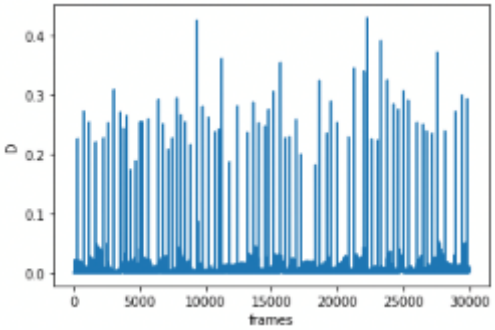


圖 19、30000 幀

圖中可以發現有多處高點，因此我們對每處高點進行勘查，發現中間有數個高點並非轉場，因此我們將大於 0.125 處視為此算法所得到的轉場處，過濾出了 25 處轉場，但少了三個轉場，位於前 200 幀處的黑背景的文字介紹，隨後我們導入片長更長的影片(**圖 19**) 進行測試。發現其偵測出了 68 處轉場，少了三個，也就是前三個，由於此演算法偵測不出前三個轉場，因此我們決定嘗試第二種方法。

(2) Cosine Similarity 方法步驟

a) 將影片中前後幀的 RGB 像素值先進行內積(相乘)，得到 Rn、Gn、Bn 三組分別代表 RGB 三層第 n 幀與第 n+1 幀的像素餘弦值。(以 R 示範)

$$R_n = \frac{\overrightarrow{r_n} \cdot \overrightarrow{r_{n+1}}}{\left|\overrightarrow{r_n}\right| \cdot \left|\overrightarrow{r_{n+1}}\right|}$$

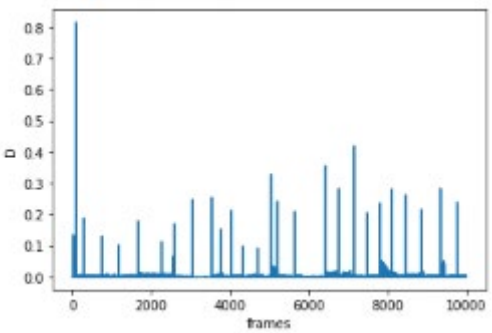
b) 將前後兩像素餘弦值相減得到得到 Rdn、Gdn、Bdn。(以 R 示範)

$$Rd_n = R_{n+1} - R_n$$

c) 將三者平方相加後除以 3 再開根號，得到 Dn。

$$D_n = \sqrt{\frac{Rd_n^2 + Gd_n^2 + Bd_n^2}{3}}$$

d) 將 Dn 對 n(幀數)作圖。



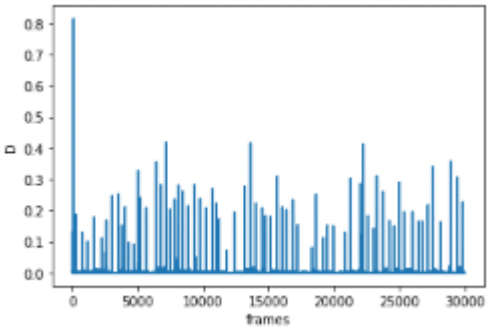


圖 20、10000 幀　　圖 21、30000 幀

(3) 整合前兩套方法

圖中可以發現有多處高點，因此我們對每個高點進行勘查，發現中間有數個高點並非轉場，因此我們將大於 0.085 處視為此算法所得到的轉場處，過濾出了 28 處轉場，且全數正確，我們為了要驗證其準確性，所以導入片長更長的影片(30000 幀，圖 23)進行測試。發現其偵測出了 68 處轉場，少了三個，分別位於 11821、18337、19198 幀，但在這三處轉場，第一個演算法有偵測到兩個，因此我們決定將其二整合，這樣其正確率達到了 98.6%(70/71)(此 30000 幀的影片中)。

2. HSV 平均色相算法

得知影片的轉場後，我們開始解決影片中色調不連續的問題，我們想到的方法是將影片轉換成數幀圖片，且將圖片由 RGB 轉換為 HSV 表示，因此就可以直接對圖片的色調也就是 H 值也就是顏色來進行調整。下一步就是要將 H 值平均，但若直接將整段的場景進行 H 值平均，會因為影片中有許多位置的變化量十分的巨大，例如說岸邊的海水因為海浪而直接由藍色變成白色，造成平均時顏色變得不正常。如下圖 22。

圖 22

為了解決此問題，因此我們決定限制平均的幀數，和對 H 值的平均進行限制。我們設計了以下算法。

(1) 選定一幀為中心取前後數幀為一列 window，以下以 window 大小等於 10 為例。

圖 23、window 示意圖

(2) 將 H 值改變量的限制在 n 閾值的區間內，將每一 window 中的其他幀和中心幀的H值相減。當 H 值差距大於 n，就將其剔除，反之則是採用此幀的 H 值進

行平均，進而生成改良後的中間幀。如此一來，可以將變化量過大的點進行消除，取用原圖片的像素代替，因此可以同時解決色彩不連續的問題，又可以解決上述(**圖 22** 中)顏色不正常的問題。

(3) 測試不同 window 大小與 n 值對影片生成的成效。

(三) ORB 預測個別幀

1. 演算法流程

實作 ORB 技術，利用此演算法找出兩張圖片中的特徵點以及算出其距離和方向。在此步驟中，我們將風景類型的影片下載下來進行測試。由於原始程式只用於找出兩張圖間的特徵點，於是我們針對找出的特徵點進行匹配後，以匹配特徵點描述符之漢明距離為標準，過濾掉較高數值的特徵點，以找出更為相似的特徵點。最後再對這些匹配且過濾完的特徵點，以座標相減算出位移的方向與數值，再平均起來。

2. 演算法測試

(1) 每隔一定幀數取出影片中的一幀，去除剩餘的部分。利用上述演算法，算出兩圖距離及方向，再利用加權平均算出中間的圖片 。針對每幀與前後原圖距離，以上述圖片移動算法算出該幀對前後原圖移動量值，得出兩張預測的圖片 ，分別為前圖與後圖所計算出的。再以該幀與前後圖的幀數差值比，對兩張圖片進行加權平均，預測該幀。

圖 24、ORB 預測個別幀演算法

(2) 測試在每幾幀取幾次的時候，影片不會震盪以及有不連續性的問題。在此步驟中，我們逐步減少幀數間隔進行實作，經過測試後，我們發現 7 幀為一個基準時最佳。

(3) 將我們所要轉成彩色的黑白影片依照前步驟設定取出圖片 ，先轉為彩色，利用演算法還原成原有的影片 。在此步驟中，我們結合了實驗組模型並進行最後的實作。

伍、研究結果

一、 利用 COCO 資料集之各模型訓練結果

各模型生成圖片比較(128px，COCO 測試集之圖片)

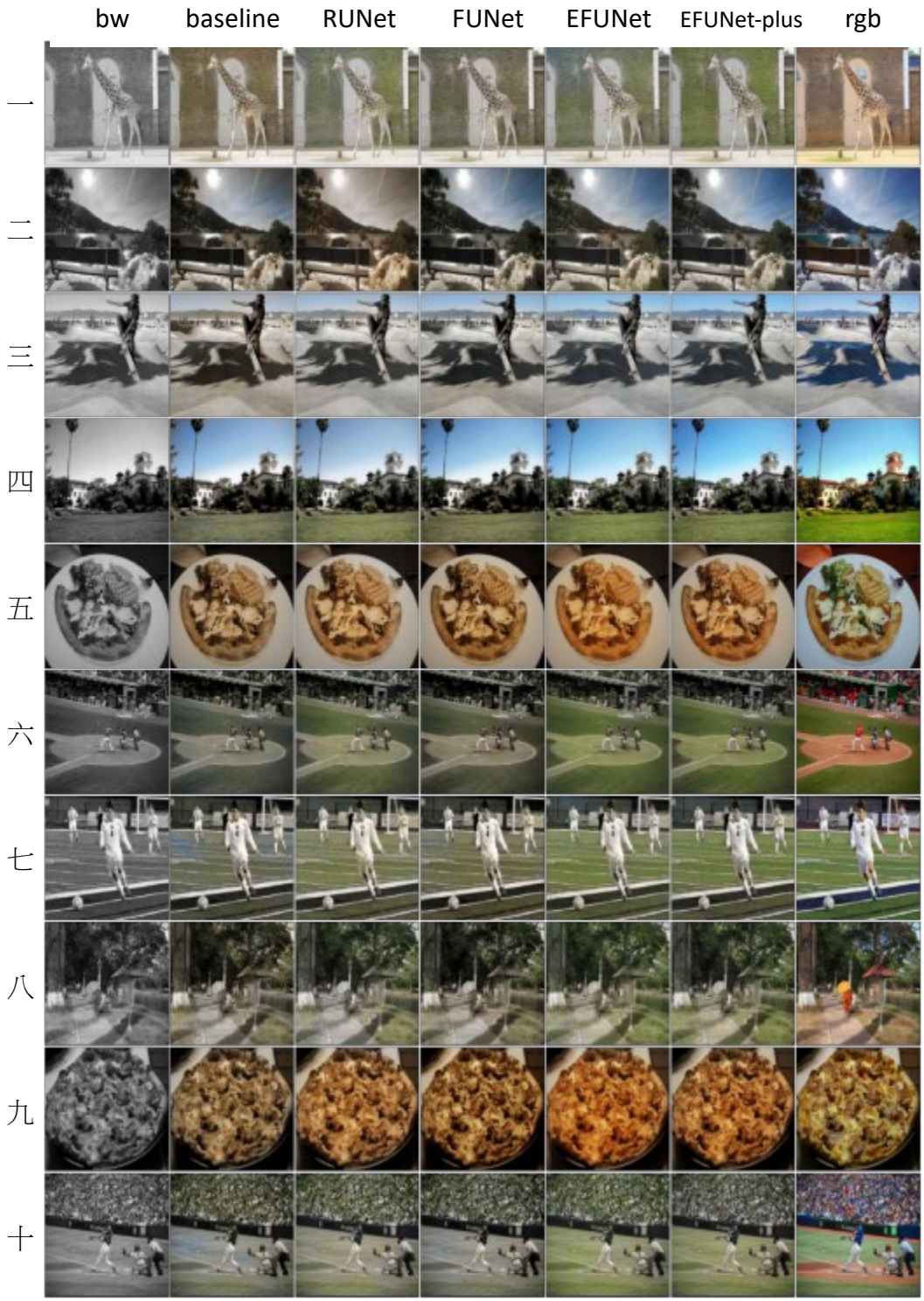


圖 25、各模型生成圖片比較

以上我們選出 COCO 測試集中生成的十種各模型生成之圖片進行比較，由上而下標示一至十；由左而右分別為黑白圖片、baseline、RUNet、FUNet、EFUNet、EFUNet-plus 所生成的圖片，最後是原彩色圖片。

由上圖展示出各模型生成的圖片，可以發現成效差異不大。但從圖片中細微處觀察，還是會有些許差異。首先，我們以全部模型皆有的狀況進行探討，我們發現對於特定場景抑或特定物體，各模型皆無法正確預測出顏色。這可見第五、第六、第八、第十張圖

片，模型無法還原食物、屋頂、以及球場的顏色，我們推論其原因在於因為此種物體(場景)本就有各種顏色之可能，所以模型生成時就很難預測出原本顏色。其次，我們以各模型得生成狀況進行探討。

就生成圖片的鮮豔度而言，我們發現 EFUNet 與 EFUNet-plus 生成的圖片比其他模型還高，可見第三、第四、第五、第九、第十張圖片，先不管生成圖片是否接近真實圖片，此二模型生成的圖片可以看出在「天空」、「草地」、「食物」等物體或場景上色彩更為豐富。而肉眼觀察生成圖片，我們發現 baseline 模型生成的圖片有一個大問題影響其生成圖片真實度，也就「顏色渲染」問題，這可見第二、第七、第十張圖片，其生成的圖片中特定物體或場景中有「渲染」的不正常顏色，比如綠色之草地渲染到藍色。這表示了模型的不穩定，顏色預測的連貫性較差。

以上分析皆是肉眼主觀分析，而討論部分會以圖片評估指標做客觀分析。

二、 利用風景資料集 fine-tune 結果

(一) 模型選擇：我們選擇 EFUNet-plus 為最佳模型進行風景資料集 fine-tune，可見**討論**部分利用圖片指標評估結果的比較。

(二) 模型生成圖片展示(128px，風景測試集之圖片)

黑白圖片

生成圖片

原彩色圖片

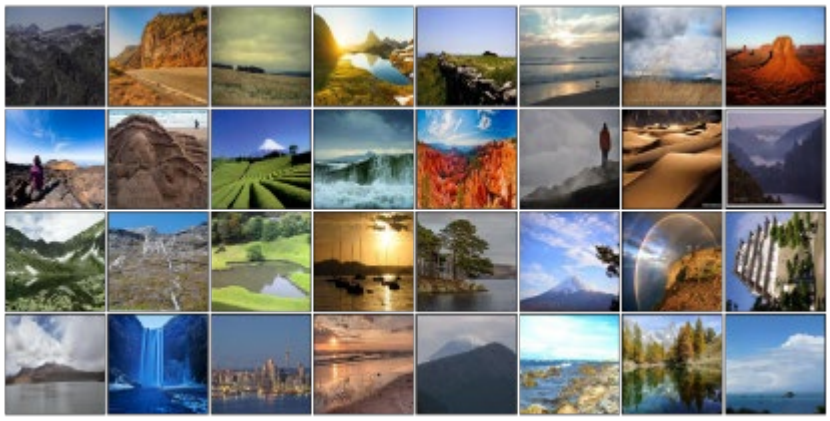


圖 26、模型 fine-tune 後生成的圖片

上圖為 EFUNet-plus 利用風景資料集的測試集生成的圖片展示，可見其適應風景資料集的狀況良好，針對各不同場景生成皆適合。且以肉眼看，生成之圖片色彩豐富度比原本 COCO 資料集訓練的(見前面的結果展示)還高。而雖然與原彩色圖片仍有差距，但我們認為此結果已經足夠良好。模型對於「風景圖片」生成之真實度已然對比原 COCO 資料集訓練的結果有明顯提升。

四、影片色彩化結果(結果於下面連結)

(一) 原風景影片

(二) 對照組(baseline)生成影片

(三) 實驗組(EFUNet-plus)

五、改善影片不連續方法結果(影片結果於下面連結)

(一) H.264 編碼技術結果

(二) HSV 提高色相穩定度結果

(三) ORB 預測個別幀

陸、討論

一、 討論並比較各實作結果

(一) 影像色彩化之評估：比較各模型生成圖片

個別圖片評估指標

1. **FID：**適用於評估 GAN 生成圖片品質的方法，它使用預訓練的模型抓取圖片特徵，再經由計算得出真實圖片分布與生成圖片分布之差異。越小越好。
2. **MSE：**衡量圖片差異的基本算法，將真實圖片與生成圖片直接相減後平方，除以圖片大小。越小越好。
3. **PSNR：**類似 MSE 的指標，只是多了一個隨著輸入值的最大數值的標準，輸出單位為 dB。越大越好。
4. **SSIM：**由亮度、對比度、及結構相似度組成，構成一個綜合指標，數值分布於-

1~1 之間。越大越好。

評分結果(COCO 資料集訓練結果，128px 圖片生成，標示紅色為最佳的)

訓練集	FID	MSE	PSNR	SSIM
baseline	18.716	0.00696	28.265	0.888
RUNet	16.141	0.00677	29.020	0.901
FUNet	16.373	0.00690	27.962	0.889
EFUNet	15.173	0.00573	32.616	0.912
EFUNet-plus	14.844	0.00590	31.135	0.912
驗證集	FID	MSE	PSNR	SSIM
baseline	26.285	0.00696	28.285	0.889
RUNet	23.661	0.00679	29.035	0.902
FUNet	23.959	0.00695	27.965	0.889
EFUNet	22.604	0.00580	32.596	0.911
EFUNet-plus	22.008	0.00593	31.145	0.912
測試集	FID	MSE	PSNR	SSIM
baseline	19.012	0.00698	28.305	0.888
RUNet	16.465	0.00678	29.033	0.901
FUNet	16.663	0.00694	28.002	0.889
EFUNet	15.404	0.00578	32.618	0.912
EFUNet-plus	15.165	0.00593	31.166	0.912

由上表分數進行評估，可以得知「殘差模組」與「特徵提取模組」此二個變因對模型成效的影響，並從中推論成因。

以 RUNet 與 baseline 進行比較，此二模型的差異在於「殘差模組」的有無，從指標中可以看出，加入殘差模組後，各指標皆有提升。以 FID 來看，代表圖片生成後的真實度提升，而以其餘三個指標的分數來看，可以推估圖片在顧及生成之真實度時，原本圖片內容能還原較好。由此可見，加入殘差模組能有效地利用到跳躍連接的優點，以增加模型生成圖片的能力。

而以 FUNet 與 baseline 進行比較，二模型差異處在於「特徵提取模組」的有無，可以看出此模組對於 FID 有改善，其幅度比 RUNet 稍差，但差距不大。惟在其他三指標上，FUNet 表現與 baseline 差不多，顯示在加入此新模組後，對於圖片「還原度」並沒有顯著提升。我們推論其背後原因在於因此模組改變的地方是張量的輸送方式，而非模型架構本身(基本模組)，而模型架構本身才是基礎，因此生成圖片的成效比 RUNet 還差。但其確實對於 FID 有所改善，我們認為此模組應當與殘差模組配合，才能發揮最大功效。而我們 EFUNet 就是如此設計的。

從上表可看出，EFUNet 與 EFUNet-plus 表現相對其他模型都好上許多。以大約相同參數量的 RUNet、FUNet、EFUNet 三種模型進行比較，可以看到單純使用殘差模組(RUNet)時，可以改善各指標分數，但幅度有限；而單純使用特徵提取模組(FUNet)時，則改善幅度更有限，甚至有些分數還比 baseline 模型差。但同時使用兩模組(EFUNet)時，則會相輔相成，即便模型大小沒有增加，但分數卻可以顯著提升。因此，我們推論若想利用特徵提取模組改善模型成效時，則必須搭配殘差模組以為基底。這樣有以下兩點好處。首先利用「殘差模組」以維持運算時的穩定性；其次同時基於此利用「特徵提取模組」進一步做各模組輸出張量整合。兩相搭配使模型在基於「穩定性」下保全圖片「內容」，改善 MSE、PSNR、SSIM，而同時增加模型的複雜度，充分利用各模組的輸出，以改善生成的「真實度」，也就是 FID。如此一來，使 EFUNet 能比前三者模型要佳，且成效顯著提升。

而我們還設計了最後一個模型，比前者模型更深更大。我們從結果發現，以 MSE、PSNR 來看，EFUNet 微幅領先 EFUNet-large，但 FID 卻剛好相反。我們推論其成因在於 EFUNet-large 模型參數量更多，其模型能學習到更複雜的映射函數以生成圖片，因此，對於生成對抗網路而言，生成圖片真實度也就隨之上升。但更複雜的映射函數可能使的模型變的較不穩定，使生成圖片可能過度渲染(可見研究結果的**圖 25**)。導致傳統圖像指標的分數不增反減，象徵圖片「內容」保持度略差。

根據以上分析，我們最終還是選擇了 EFUNet-large 當作 fine-tune 部分的模型，原因是因為我們認為雖然其 MSE、PSNR 比 EFUNet 差，但相差幅度小，且對於生成對抗網路而言，FID 更能表現生成圖片之好壞，而 EFUNet-large 的 FID 比 EFUNet 還低，也就是生成圖片更為真實，故我們最後以 EFUNet-large 進行風景資料集之 fine-tune。

評分結果(風景資料集訓練結果，128px 圖片生成，EFUNet-large)

訓練集	FID	MSE	PSNR	SSIM
EFUNet-plus	11.262	0.00576	30.835	0.898
驗證集	FID	MSE	PSNR	SSIM
EFUNet-plus	37.960	0.00823	30.449	0.890
測試集	FID	MSE	PSNR	SSIM
EFUNet-plus	35.458	0.00820	30.421	0.889

以上是利用 EFUNet-large 所 fine-tune 的結果，從傳統指標(MSE、PSNR、SSIM)分數看，比原本用 COCO 資料集訓練所得出的結果稍微差了一些，這可以從以下**圖 27** 來解釋。

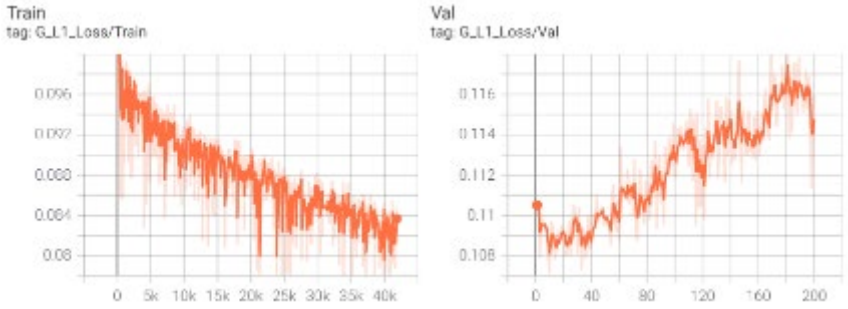


圖 27、訓練集和測試集的 L1 Loss

以上圖看，從 L1 Loss 可以看得出有 overfitting 的問題，這是因為我們使用整個模型 fine-tune，而風景圖片數量與 COCO 資料集相差甚遠。這也導致上表分數變差。我們檢查了訓練時生成的圖片，我們認為其分數是受到改善「真實度」的影響。原本模型雖然在傳統評估分數上表現優異，但其生成之「彩度」不高，但利用特定資料集(風景資料集)fine-tune 後，模型能學習到如何生成高「真實度」的圖片，也就是色彩豐富度高的圖片，也因此而犧牲了「內容損失」。但可見下圖雖然它在 L1 Loss 表現不理想，但影響並不大。

10000 iteration

20000 iteration

30000 iteration

40000 iteration

最終結果

原色彩圖片

由上圖肉眼大致可看出生成之圖片隨著迭代次數增加，彩度(豐富度)增加，但其「內容」差異不大，也就是雖然 L1 Loss 逐步上升，但對於圖片品質沒有過大影響，反而圖片「真實度」提高。

(二) 改善連續性問題方法之比較

1. **連續性評估方法設計：**我們想到利用前面找影片的轉場時所使用的演算法來評估改善影片不連續性的三套方法的成效。以下為具體步驟。

   (1) 使用本研究之轉場偵測方法，將影片的 0~n+1 幀與 1~n+2 幀取出，使用 Cosine Similarity 算出兩圖相似度。

   (2) 將以上得出的(n, c)大小的矩陣對 c 通道平方，平均，再開根號。

   (3) 全部平均得出一個數值，代表影片中圖片的相似程度，數值介於-1~1 之間，越大越好。

   算出各個影片的數值後，將經過不同提升連續性方法後的影片與原影片進行相減，可以視為各個方法對不連續性的改善量，因此我們將其定義成評估不同方法成效的不連續性分數。

2. **整體影片評估 Visual Multimethod Assessment Fusion (VMAF)**

   由於利用每位觀察者所觀察影片品質的好壞過於主觀，於是我們採用利用結合主觀影片品質評估和客觀影片品質評估的程式 Visual Multimethod Assessment Fusion，簡稱 VMAF。VMAF 分數很容易理解，因為它在 0~100 的範圍內執行的。此演算法的基本理念為利用主觀實驗中獲得的意見分數建立機器學習模型並進行訓練和測試。一個影片的特徵可以由很多基本指標來表示，每一種皆有優劣，此模型便是分配各種基本指標一定的權重，這樣最終得到的指標就可以保留每個基本指標的結果，得出更精確的最終分數。其分數不是絕對指標，而是由兩個影片對比而成，分數愈高，理論上相似度愈高，而差異性愈小。(此評估方法我們只用來評估 H.264 方法，因為我們只有 H.264 方法需要知道影片之相似度好壞)

3. **評估結果**

   (1) H.264 參考畫面與動作估計：

   因為此方法經過壓縮，有些特徵會因此損失，於是我們決定利用兩種方法來評估此方法，分別為 VMAF 以及不連續性分數。由左下圖我們可以得知每個種類的 VMAF 分數都不會與原影片的差距太大(皆小於 2.5)，換句話說，此方法並不會巨大影響整體影片相似度。從各個不連續性分數我們可以得知每種速度皆可改善，而其中當壓縮速度為 **I(Veryfast)** 時的成效最為明顯，分數為 0.00070223，而此時的 VMAF 分數為 84.983295。見**圖 28**。我們原先認為 Placebo 會有最良好的成效，因為他的 reference frames 最多，然而，成效最好的竟然是 Veryfast，我們推論是因為 Placebo 參考太多幀了，導致有食物到場景轉換仍然在

進行動做估計，而導致一些場景因為參考上個場景的參數所以顏色由正常變得不正常。還有一個原因，有的場景裡的幀數所生成的圖片原本就是正常的，因為參考範圍過大難免參考到不正常的的幀數。反之，Verfast 的 reference frames 因為只有三幀，所以就算碰到轉場處也是只有少數的幀數受到影響。而至於為什麼 Ultrafast 的分數掉下去，我們也有推論。因為 Ultrafast 的 reference frames 只有一幀。依據定義，這意味著永遠都只能參考前後一幀的畫面，而這並無意義，因為如此一來幾乎整部影片的幀數都是 I-frames，也就是隨機生成的圖片。

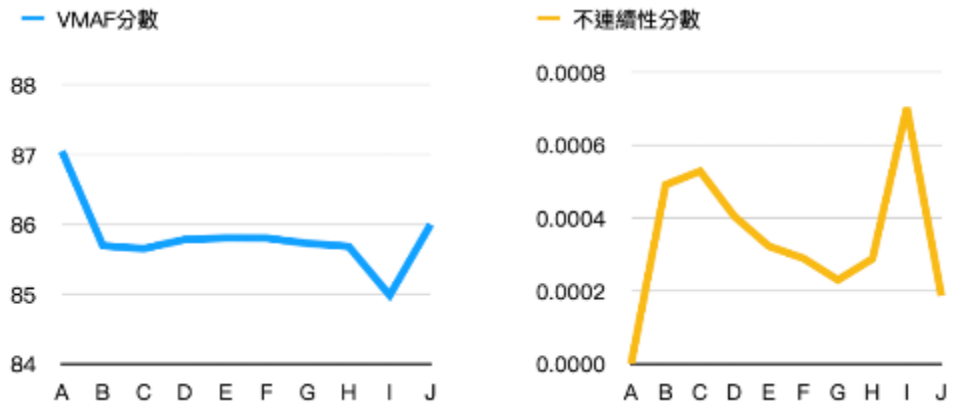


圖 28、A-J 分別為慢到快的生成速度

(2) HSV 提高色相穩定度

(f = window size， n = 閥值)原生成影片：0.984269

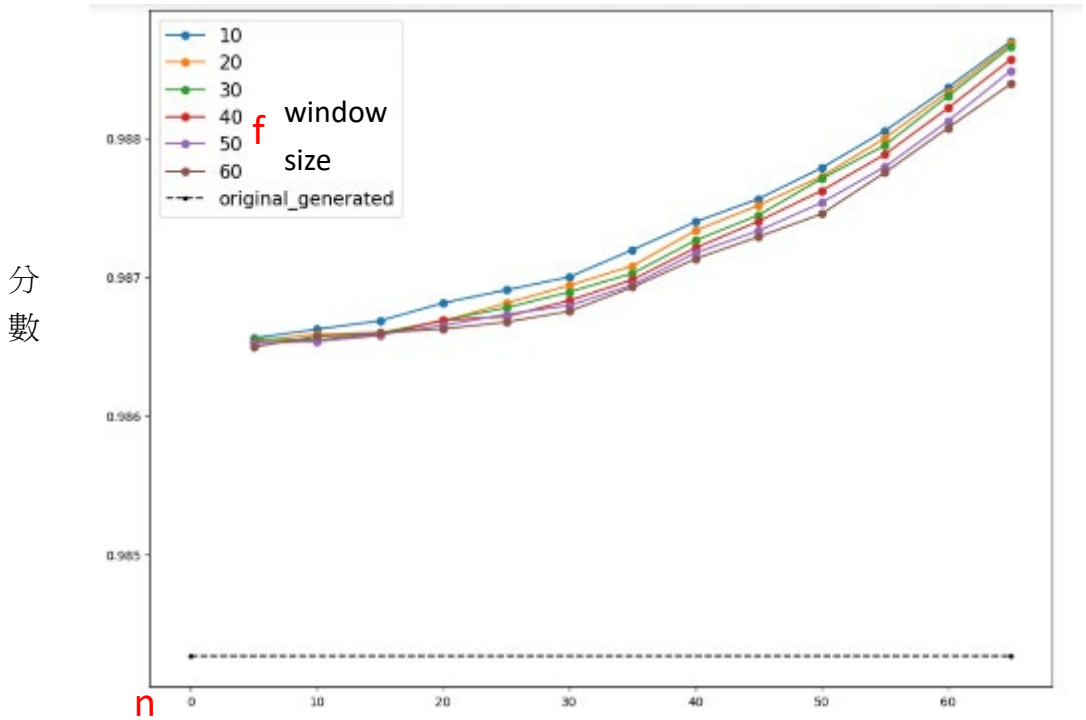


圖 29、HSV 中的 H 當 c 通道的結果(縱軸為分數/橫軸為 n 值)

(以上的影片是由模型 EFUNet-plus 所生成出的 3000 幀影片進行提高色相穩定

度而成。)

分析**圖 29** 我們可以發現，當 n 值增加時，對於不連續性的改善越有效。而在 window size 的部分，各個實驗組沒有太大的差異，由於算力上和生成時間上的限制，我們最後決定使用以 window size=10 , n=65 的生成結果去和其他方法進行比較。(剩餘影片詳細分數如下**圖 30**)

	5	10	15	20	25	30	35	40	45	50	55	60	65
10	0.90658	0.903823	0.906604	0.906813	0.906906	0.907	0.907195	0.907397	0.907564	0.907706	0.908053	0.908365	0.9087
20	0.906536	0.903588	0.906599	0.906681	0.906811	0.906939	0.90708	0.907334	0.907517	0.907725	0.908	0.908332	0.908681
30	0.906523	0.906546	0.906604	0.906687	0.906778	0.906892	0.907026	0.907264	0.907446	0.907709	0.907949	0.908303	0.908661
40	0.906527	0.903537	0.906579	0.906689	0.906716	0.90683	0.90698	0.907209	0.907402	0.907822	0.907893	0.908217	0.908569
50	0.906524	0.903538	0.906587	0.906651	0.906731	0.906794	0.903842	0.907174	0.907333	0.907536	0.907793	0.908123	0.908487
60	0.906496	0.906571	0.906598	0.906626	0.906675	0.906753	0.906826	0.907131	0.90729	0.907456	0.907752	0.908074	0.908393

圖 30、HSV 詳細分數結果

(3)ORB 預測個別幀

以下的分數是由模型 EFUNet-plus 結合 ORB 預測個別幀算法後所生成出的 3000 幀影片。

	ORB 方法
連續性分數	0.0015172

(H.264、ORB 與 HSV 的計算方式運用皆是原影片，但 ORB 算法無法適用於轉場的情況，固我們將影片分段接起，導致評估後的數值不同。) 而 H.264 方法的分數低了其他兩個方法一截，我們認為這是理所當然的，因為他與影片的相似度最高，且保留了最多的特徵。而 ORB 方法又略輸了 HSV 方法，但執行時間略贏了 HSV 方法。我們發現各個方法都有其優缺。於是我們下此結論，如果要保留最多的影片特徵，則選擇 H.264 方法，如果需要最佳的改善色彩不連續性則選擇 HSV 方法，而如果有時間限制，則選擇 ORB 方法

	H.264 方法 (veryfast)	HSV 方法 (window size =10, n=65)	ORB 方法
連續性分數	0.00070223	0.00443085	0.0015172
VMAF 分數	84.983295	無	無

分析上述實驗結果可以得知 HSV 方法相對 ORB 方法以及 H.264 方法對連續性的提高有著更顯著的效果。

二、 未來展望

(一) 圖片色彩化模型

本研究最後以 EFUNet-plus 模型為最佳模型，而我們希望可以基於本模型，未來做以下改進：一、探討不同判別器的影響，而非只改進生成器；二、探討其他方法以改善圖片色彩化問題，比如設計新的損失函數； 三、繼續進行更高解析度圖片的訓練；四、將

本模型中的模組運用到其他生成圖片的研究上，如 Super-Resolution 的研究。

(二) H.264 編碼技術算法

本研究的 H.264 編碼技術算法雖然可以減小影片不連續性，但它還是會減少些許的影片特徵，所以我們希望在未來可以把動作估計以及多重參考畫面單獨提出進行實作。

(三) HSV 平均色相算法

因為我們的演算法模型是以原幀為標準幀，可能會因為原幀本來就有著因為模型生成的破圖，而導致平均後仍然無法改善不連續問題，因此我們希望在未來可以自動偵測破圖的位置，並利用其他幀正確的顏色進行填充。

(四) ORB 預測個別幀算法

本研究的 ORB 生成圖片演算法雖然可以減小影片不連續性，但它的算法只考慮到圖片平均位移，而沒有顧及圖片的縮放、轉動等，或者圖片中個別物體的運動。如果運用在其他動態的影片之中，可能會導致影片模糊生成的現象。

我們想到的未來改進的方法為結合深度學習的技術，使得生成的圖片不只是「硬性」參考前後兩圖的移動進行生成，還能輸出更為逼真的結果。

柒、結論

本研究的前部分為圖片色彩化模型的建構與成效評估，我們建置了兩種新模組，分別是殘差模組與特徵提取模組來對原 baseline 模型進行改善。我們先利用了 COCO 資料集訓練模型以得出一個「通用」的色彩化結果，然後進行比較及評估。我們發現當我們同時使用兩模組(EFUNet)時，模型成效能大幅提升，不管是 MSE 等傳統圖片評估分數抑或用來評估生成圖片真實度的 FID 指數。而經由各模型比較後，我們推論其原因是因為加入殘差模組時能改善整體模型對於特徵張量的保持度，因此對於圖片還原度有所改善，另外加入特徵提取模組後，模型能針對各模組輸出進行整合，藉此利用到各層輸出特徵，改善模型生成之真實度。以此，我們最後選擇了 EFUNet-plus 當作前部分之最佳模組，並以風景資料集訓練以適應色彩化風景之生成。

分析完圖片色彩化模型成效後，我們利用 ORB、HSV 以及 H.264 編碼技術來解決影片不連續性。首先，我們結合絕對值和餘弦演算法，可以進行高準確率的轉場偵測，以將影片切割為片段，利於後續解決不連續性的方法。接著我們分別利用 HSV 以及 ORB 預測個別幀來解決影片不連續性。而 H.264 則不需轉場可直接執行。我們由類似前述轉場評估的方法來進行連續性判斷，我們發現三種方法皆可有效改善影片的不連續性。最後我們可以得出一個結論，當我們所需要的生成影片與原影片的相似度要最高時，我們就選擇 H.264 方法，當我們生成的影片所需要的影片部連續性的成效為最高時，我們就選擇 HSV，當我們所需的時間最少時，我們則選擇 ORB。

本次研究的主要應用為使黑白影片色彩化並增加其流暢度，兩者我們有改進舊有的方法，

並開發新的算法解決，順利達成目標。

捌、參考資料及其他

[1] Jon Bruner and Adit Deshpande (2018*) Generative Adversarial Networks for Beginners Build a neural network that learns to generate handwritten digits*. Retrieved from https://github.com/jonbruner/generative-adversarial-networks/blob/master/gan-notebook.ipynb

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio.(2014) *Generative Adversarial Networks.* Retrieved from https://arxiv.org/abs/1406.2661

[3] Gerasimos (Jerry) Spanakis (2018*) LoGAN: Generating Logos with a Generative Adversarial Neural Network Conditioned on color.* Retrieved from https://www.researchgate.net/publication/330474693_LoGAN_Generating_Logos_with_a_Generative_Adversarial_Neural_Network_Conditioned_on_Color

[4] Tensorflow 官網：https://www.tensorflow.org/learn?hl=zh-tw

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun (2015) *Deep Residual Learning for Image Recognition*. Retrieved from https://arxiv.org/pdf/1512.03385.pdf

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015*) U-Net: Convolutional Networks for Biomedical Image Segmentation*. Retrieved from https://arxiv.org/abs/1505.04597

[7] Han Zhang, Ian Goodfellow, Dimitris Metaxas, Augustus Odena (2019) *Self-Attention Generative Adversarial Networks.* Retrieved from http://proceedings.mlr.press/v97/zhang19d/zhang19d.pdf

[8] 楊士萱、陳柏源：H.264/AVC 技術與應用簡介 Retrieved from https://myweb.ntut.edu.tw/~shyang/Journal%20Papers/H264_AVC.pdf

[9] AcceptedLin.(2018*).SIFT 特徵匹配算法介紹—尋找圖像特徵點的原理* Retrieved from

https://www.twblogs.net/a/5bfe2416bd9eee7aec4e41a8

[10] Ethan Rublee,Vincent Rabaud,Kurt Konolige,Gary Bradski (2011) *ORB: an efficient alternative to SIFT or SURF*. Retrieved from http://www.gwylab.com/download/ORB_2012.pdf

[11] Devin-Gao (2016) *ORB 特徵提取詳解* Retrieved from https://www.itread01.com/content/1542396988.html

[12] Dibya Jyoti Bora, Anil Kumar Gupta, Fayaz Ahmad Khan (2015) *Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation*, International Journal of Emerging Technology and Advanced Engineering(pp. 192-203) Retrieved from https://arxiv.org/ftp/arxiv/papers/1506/1506.01472.pdf

[13]Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros (2016). Image-to-Image Translation with Conditional Adversarial Networks. Retrieved from https://arxiv.org/abs/1611.07004

[14] PyTorch 官網：https://pytorch.org

【評語】052508

此作品利用深度學習技術，將黑白影片彩色化，研究方法嚴謹，分析詳盡，並對於影片不連續問題，加以探討，也有提出改善方法。建議未來能針對所提機制，不同的影片表現的差異上，做更系統性的探討。

作品簡報

Automatic Video Colorization

利用深度學習將黑白影片色彩化

我們想達成的目的

1. 影像色彩化
2. 解決場景轉換後色調不連續問題
3. 分析各個實作的結果及成效差異

研究流程圖

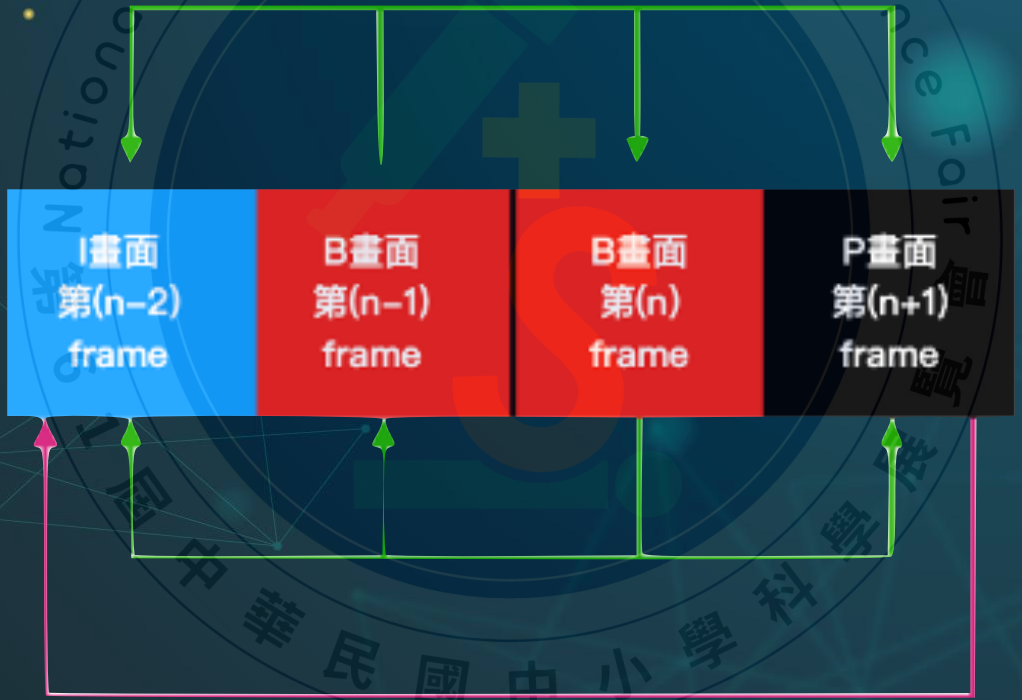

文獻探討
訓練模型之資料蒐集
資料處理
模型建構
訓練通用模型
用指標評估各模型
fine-tune 最好的模型
解決不連續性
H.264
參考畫面與動作估計
將影片轉成
H.264 編碼
ORB
預測個別幀
特徵點匹配
特徵點找尋
算出平均位移
加全平均算圖
HSV
提高色相穩定度
找出轉場
加入限制
平均色相
評估成效

五個模型

	生成器	判別器
對照組：Basic U-Net	Encoder：16x 縮小 Decoder：16x 放大	4層含殘差模組的PatchGAN
實驗組：**RUNet**	將內部基本模型改為殘差模組 （相對對照組）	同上
實驗組：**FUNet**	加入本研究提出的特徵提取模組 （相對對照組）	同上
實驗組：**EFUNet**	前兩個實驗組合併模組	同上
實驗組：**EFUNet-plus**	增加前一模型的深度	同上

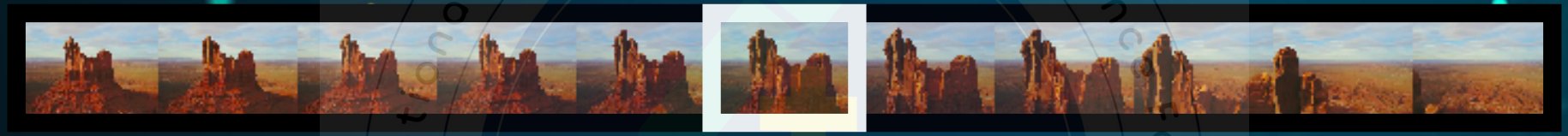
解決色彩不連續

1. H.264參考畫面與動作估計



2. HSV提高色相穩定度

標準幀



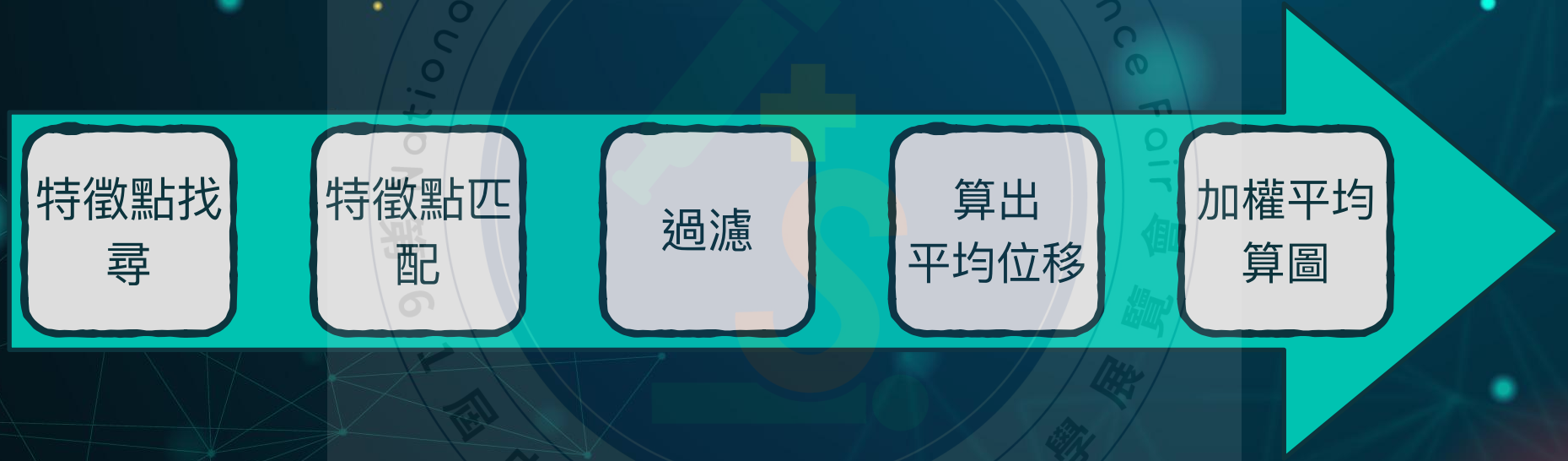
將window中的各幀跟**標準幀**相減

令一個閥值 N

大於 N 就將其剔除

小於 N 就採取的H值進行平均

3. ORB預測個別幀



模型圖片訓練結果

生成影片評估

測試集	FID	MSE	PSNR	SSIM
baseline	19.012	0.00698	28.305	0.888
RUNet	16.465	0.00678	29.033	0.901
FUNet	16.663	0.00694	28.002	0.889
EFUNet	15.404	**0.00578**	**32.618**	**0.912**
EFUNet-plus	**15.165**	0.00593	31.166	**0.912**

三套方案成果比較

	H.264 (veryfast)	HSV (window size=10,n=65)	ORB
不連續分數	0.00070223	0.00443085	0.0015172

結論

最佳色彩化模型

EFUNet-plus

殘差模組

特徵提取模組

解決影片不連續

不連續分數

HSV>ORB>H.264

連續性最高 HSV

相似度最高 H.264

生成時間最快 ORB