影像處理概論

作業三：視訊處理與顯著性偵測實習

選擇之論文介紹

我選擇的論文是助教提供的第四篇： (CVPR19) See More, Know More: Unsupervised Video Object Segmentation with Co-Attention Siamese Networks ，這份論文的基礎是由非監督式的影片顯著物分析，其主要著重在影片幀之間固有的相關性，並採用了全域注視機制（ global co-attention mechanism ）來改進基於深度學習的解決方案，這項機制主要專注於在短時間內透過外觀和運動來學習具有區別性的前景表示（ foreground representations ）。 COSNet 中的 Co-attention layer 藉由合併計算並且將 Co-attention response 加入特徵空間中，為全域相關性和場景前後文的理解提供了有效輔助。 COSNet 使用成對的影片幀來訓練，使得訓練數據和學習能力得已增加。在 Segmentation 階段， Co-attention model 透過同時處理多個參考幀來獲得有用的資訊，如此一來可以更容易找出出現頻率較高且較為突出的前方物體。在使用豐富背景的影片的實驗中， COSNet 大大優於單一且端到端的可訓練框架模型。

COSNet 將時間視為相關的影響，透過一對幀的網絡訓練策略，學會從每個幀的背景中區分出主要對象，並追蹤幀之間的時間相關性。在很多的實驗結果都證明， COSNet 可以有效地抑制相似性很高的目標所產生的干擾。 COSNet 是用於處理有序幀的學習通用模型，可以很容易的使用在各種影像分析項目，例如影片顯著物偵測和光流估計（ optical flow estimation ）。

實作方法

labelme

1. 在終端機輸入 labelme 開啟 Labelme 圖形介面。
2. 打開圖片。
3. 點擊 Create Polygons。
4. 用滑鼠標記 Salient objects。
5. 儲存標籤。
6. 執行提供的程式產生所需檔案。

論文所提供之網路架構

1. Frame pair ：一組兩幅幀的影像，會被輸入至特徵模組。
2. Feature embedding ：透過特徵模組得到特徵表示。

3. Co-attention ： Co-attention module 對於兩張圖像當中產生關聯的摘要進行計算。
4. Segmentation ：將 Co-attention module 得到的結果合併並產生分割結果。
5. Loss ：透過將分割結果和 ground truth 進行比對得到誤差。

模型訓練與評估結果

Training

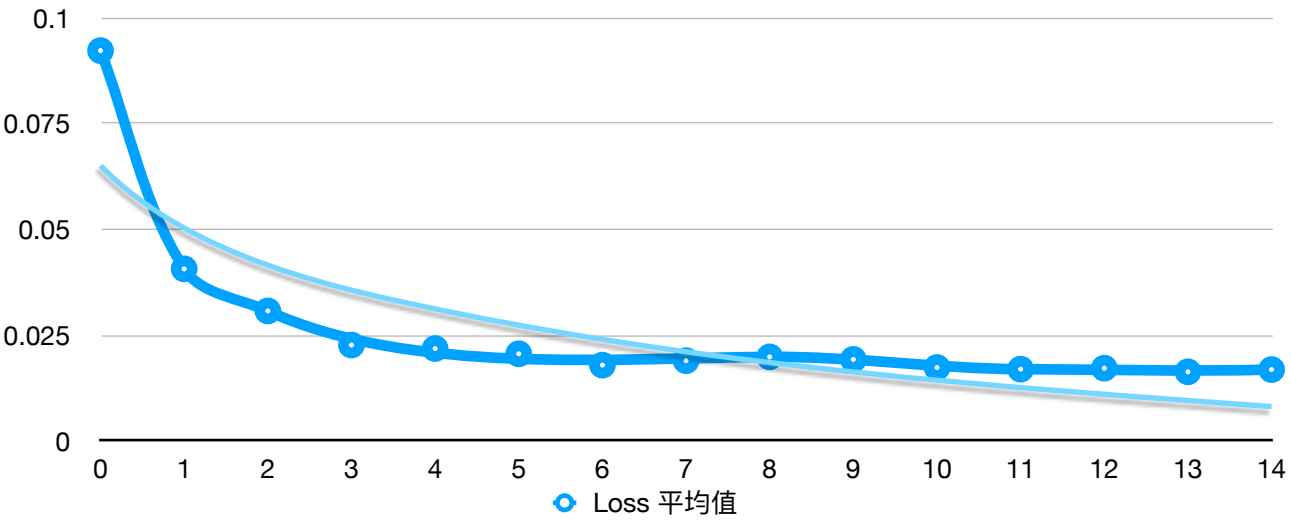
改良

我在 Module 中增加了數層不同的 layer 包含 bn1 、 layer3 、 main_classifier 等等，希望由藉由 layer 的增加，使得結果更為準確。此外，也將已經不再由Scipy支援的imresize改為使用pillow，並更改為uint8型態。

由於訓練時間不夠長的關係，原先的 learning rate 似乎太小，因此調整初始 learning rate 為 0.025 。

Loss 表現結果

將全部 Training 結果的 Loss 做平均，產生 0 至 14 個 Epochs 的 Loss 表現，以其平均 Loss 做圖，其結果如下圖。圖中的淺藍色曲線為對數趨勢線，可以看出在 4 個 Epochs 之前，訓練出來的 Loss 有明顯的在減少，之後的減少幅度開始慢慢趨緩



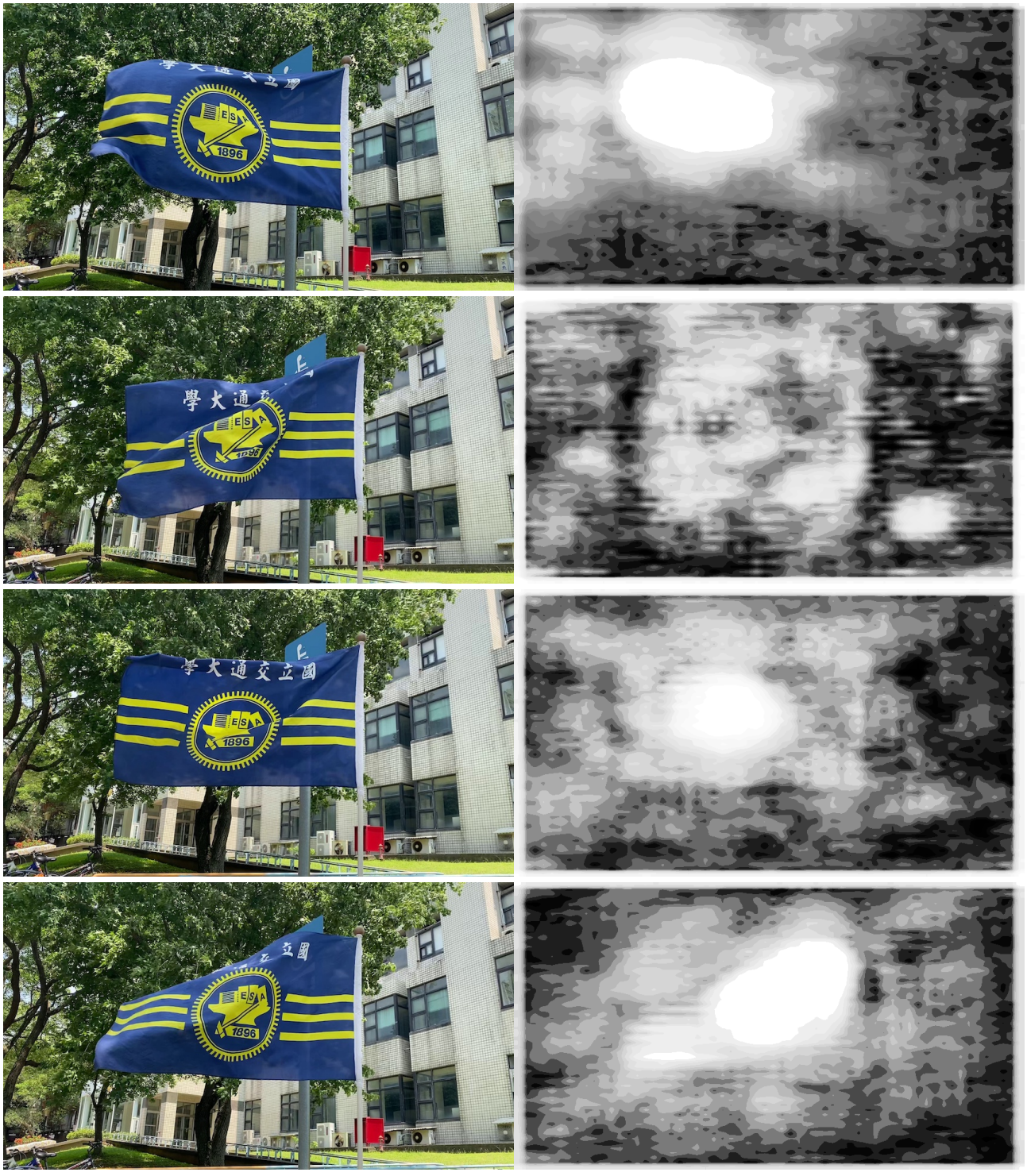
Testing

分析

儘管在訓練時 Loss 看似已經很低，但是實際跑測試時，效果卻非常不如預期，推測原因是因為在只有單個GPU的機器上做訓練，並非此篇 paper 所提供的 code 的預期。

結果

以下為我 Part 1 時的影片部分幀的測試結果。可以看到測試的結果並不相當顯著，推測是由於訓練不夠的關係所致。值得一提的是，許多 Mask 都有白邊的產生，也就是圖片周圍有一圈白色，而我產生此結果的原因並不太清楚。



分析與討論

環境

在這個作業當中，訓練之前，需要確保 torch.cuda.is_available() 為 True 才可進行訓練，但如此一來需要先更新 Cuda 版本，不過更新完之後，即會出現 Nvidia driver 和 Cuda 版本不相容的問題，同時不同版本的驅動也產生相依性問題，無法使用 apt install 解決，而是需要 reboot 才可以，但是容器無法 reboot ，會出現無法與最初 daemon 溝通的問題。此外，由於我所選擇的論文是需要 2 張顯卡來做訓練，如果以 1 張顯卡來訓練，結果將會大大不如預期，因此後來只好改用其他機器。

另外，從 paper 所提供的 code 看來，其預設將 batch size 設為 10 ，max epoch 設為 15 ，所需的記憶體和時間都將遠遠不夠，再加上需要進行修改並再度測試，且多人共用下，時間相當緊迫，需要修改 batch size 和 max epoch，使其降低後才得以進行訓練。

除此之外，論文所附的 code 使用了 from scipy.misc import imresize ，但在官方文件中，imresize 已不再由新版本的 scipy 支援，須改用 pillow ，需要將所有 imresize(img, self.inputRes) 改為 np.array(Image.fromarray(((img + 1)*127.5).astype(np.uint8)).resize(self.inputRes)) ，其中的 astype(np.uint8) 是需要確保最後的數值型態須為 uint8 否則會報錯，這些都是在真正開始訓練之前，即會遇上的問題，整個環境安裝的時間大約耗時一週。

結果討論

至於結果的討論，由於這份 paper 所提供的 code 有顯得將single GPU和 multiple GPUs 的訓練方式作出區別，因此，一開始我使用但一顯卡做訓練時，其結果非常差，所產生的 results 都是全部灰色，後來才想辦法移到雙顯卡的機器進行訓練，並觀察結果是否有改善。

我認為，如果使用預設到 batch size、max epoch以及input size，整個結果應該會更加準確，因為paper當中提到，Co-attention 不只著重在當前幀，也會觀察前後關聯，應該要優於傳統的圖片顯著物辨識（我認為至少應該優於上一個作業），但是礙於時間和記憶體空間的限制，可能暫時無法做出那樣的訓練。在未來有時間的時候，我希望可重新詳細的再做一次訓練，屆時再來觀察其整體結果是否有更加優越。

字數統計

2477 個字元（不含空格）
2692 個字元（含空格）
1543 個字