**Deep-Learning-for-Autonomous-Driving**

Lab3 : Semantic Segmentation

ID : 310605007 Member : 鄭晴立 Department : 機器人學程

# Introduction

我針對助教提供的建議，盡量使用原本提供的code去實現U-Net，並在原始UNet上加上以下改進，包含Loss function、Skip connection及Dilated convolution，並由以下幾點說明其改內容。

## U-Net

本次Lab為U-Net實作，以下為實作的code。U-Net分為Encoder及Decoder。Encoder為五次的雙層Convolution，我將其定義在DownConv裡面，中間會進行maxpooling。Decoder部分則進行五次的Upsampling及雙層Convolution，我將其定義在UpConv。

|  |  |
| --- | --- |
| 一張含有 文字 的圖片  自動產生的描述  圖 一 U-Net code | 一張含有 文字 的圖片  自動產生的描述  圖 二 Upsampling of U-Net |
| 圖 三 Convolution of U-Net |

## Dilated convolution

由於時間關係，沒有單獨測試Dilated convolution，僅隨意挑選一個尺度進行實驗。

一張含有 文字 的圖片

自動產生的描述

圖 四 Dilated convolution code

## Loss Function

由於本次的所要解決的是語意分割的問題，因此有幾個較為經典的Loss Function，包含dice soft loss、soft iou loss及Lovasz softmax loss，由於時間關係，僅嘗試dice soft loss、Lovasz softmax loss及傳統的Cross Entropy Loss。圖 五 為Lovasz softmax loss節錄。

一張含有 文字, 螢幕擷取畫面, 黑色, 螢幕 的圖片

自動產生的描述

圖 五 Lovasz softmax loss

## U-Net++

### U-Net++ Introduction

我找了幾篇U-Net相關文獻後，發現U-Net++為基於U-Net做改進的代表性文章，而我將採用其方法嘗試。U-Net++ 為 2019發表的一篇文獻，作者認為，傳統U-Net的Skip connection是有改進空間的，UNet僅將同一層級的特徵串接在一起，因此U-Net++將許多不同scale的UNet模塊串接在一起，讓不同的特徵之間能都流通。圖 六 為UNet++的code。

|  |  |
| --- | --- |
| 一張含有 文字 的圖片  自動產生的描述 | 一張含有 文字 的圖片  自動產生的描述 |

圖 六 UNet++ code

### U-Net++ Architecture

U-Net++ 的架構可分解成多（L1~L4）層U-Net組合而成，可對應到圖 六 和 圖 七 的L1到L4，可視為將模型加廣，並加強各層之間特徵的連結。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

圖 七 UNet++ Architecture

### Deep supervision

另一個重點於U-Net++ 是Deep supervision，因為這樣的模型會有兩個問題：

* 如果用最後一層的單一輸出，會導致模型train不起來。
* 由於我們將模型加廣，表現變好也是理所當然的，相對模型大小也會變大很多。

有關第一點，我已經實際嘗試過，確實會train不起來；而有關第二點，由於我們將模型加廣，導致表現變好也是理所當然的，因此透過Deep supervision的機制可以去觀察深度的影響力，藉此刪除不必要的層數，其實際操作方式為，在validation階段時，去觀察L3及L4的差異，若差異太小，則代表L4層可被刪除，因此在最後輸出model將最後一層刪掉，但因為其在backward也會有幫助，因此不會一開始就刪掉。而我在實作上，並沒有額外做刪除的動作，僅將其各層Loss加起來，使其model能夠train起來。

一張含有 文字 的圖片

自動產生的描述

圖 八 Deep supervision下的loss運作方式

## Learning rate schedule

由於這次的圖片所要train的時間比較久，因此需要透過Learning rate schedule，能夠增加training效率，我僅做簡單的schedule，能夠稍微改善其效率。

一張含有 文字 的圖片

自動產生的描述

圖 九 Learning rate schedule

# Result

## Implement

我將前章節的修正做一個結論，mIOU為 〝 - 〞 代表train不起來，或是前幾個epoch表現就非常差，因為時間不夠，所以沒時間一一挑出問題，不一定代表其改進是無效的。

我僅做出U-Net++ 及傳統U-Net，而且還沒有達到預期的效果，而其他的Loss Function及Dilation，都因為某些問題，無法完全確定其效果如何。

表 一 個修正方式之比較

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Loss | Dilation | mIOU |
| U-Net | Cross Entropy | No | 0.838307 |
| U-Net | Lovasz softmax | No | - |
| U-Net | dice soft loss | No | - |
| U-Net | Cross Entropy | Yes | - |
| U-Net++ | Cross Entropy | Yes | - |
| U-Net++ | Cross Entropy | No | 0.827237 |

表 二 預測結果

|  |  |
| --- | --- |
| Model | Image |
| U-Net |  |
| U-Net++ |  |

## Conclusion

從結果可以看出，我的各項調整皆無法使表現優於原始U-Net，不管是mIOU、還是肉眼，包含loss function、及model的改善，大概就像我的人生一樣，庸庸碌碌、徒勞無功、超級可悲。

# Discussion

## Log file problem

整體的code都盡量依照助教原先所提供去修正，因此在log file 裡面產生了蠻多看不太懂得檔案，希望未來助教能提供code的說明。

## UNet++

一開始使用傳統U-Net時validation的mIOU已經可以到0.8左右了，後續使用其他修正皆很難往上提升，或是需要花很多時間。

## Testing時會CUDA out of memory

因為要求testing時要用原始大小，

我用家中12G的3060顯卡仍然無法testing，希望未來不要用原始大小的圖片去做testing。

## mIOU的提升的影響

由於傳統傳統U-Net方法已經可以達到不錯的效果，目前嘗試的幾種方法，以及文獻上所看到的方法，大多僅能提升一些效能，mIOU在0.7~0.85之間，對於肉眼來說，幾乎是看不出差別的，因此可能需要捨棄傳統U-Net架構，使用更突破性的方法才有辦法更加提升表現性。

## Upsample的方法對於mIOU的影響

似乎不同的Upsample方式會對mIOU有些許的影響，嘗試了transform的resize、nn的Usample以及助教提供的方式，test出來的mIOU會有些許不同，但沒時間實際去比較。

## Code的差異

我和同學分別在不同的時間下載助教提供的code，

裡面code內容似乎有些許的不同，以至於我們在testing的時候會遇到一些相同，但結果不同的奇怪的狀況。