HW4 Report

B08901165 電機四 南策昇

Problem 1

1. (a)NeRF 將一個景象各個角度照片的每個點轉換成空間中那個點的五維向量，其中前三維是該點 xyz，後兩維是視角 theta、phi。之所以要多上視角的資訊是因為視角不同也會影響物體的遮蔽情形和顏色。之後，作者要求 model 能夠預測出空間中點的 density 和 r、g、b 共四個數值。因為 rgb 數值 groundtruth 可以從照片上取得，因此這個 model 在邏輯上是可以取得不錯效果的。
   除了這樣直觀的設計外，作者還加入幾個改進。首先，這個 model 的 inputs 並不是同時進入的，outputs 也不是同時出來，xyz 首先會經過一堆 MLP，產生一個 feature vector 和 density，feature vector 再和 theta phi 做連接，過完後面的 model 產生 rgb。之所以這樣做是因為 density 理論上的確與看物體的視角無關，因此這樣可以幫助 model 學到更有用的資訊。再來，作者使用 position encoding 將位置資訊投影到更高維度的空間，幫助 model 學習。此外，如果要重建模型，原本空間中的每一個點都需要去判定他的透明度，為了加快 model 速度，所以作者先用一個 coarse model 快速判定哪些點密度較高，有機會影響輸出，接著再用 fine model 依據 coarse model 的結果，將較有機會影響輸出的點的採樣機率調高，這樣就省去採樣那些不重要的點的時間了。

1. (b)我認為最重要的是將 theta 和 phi 列為 inputs，這也是 NeRF 的核心之一。作者做的其他改進大多是讓 model 比較容易 train，但是增加角度卻是直接提供給 model 更多資訊，並且僅用在預測 rgb 上面也能幫助到 model 從 xyz 找出 density。可以看到原始 paper 中有做 ablation study，少掉 view dependency 的版本也是各項 metrics 差距最大的。

1. (c)NeRF 的優點是畫質很好，而且因為有得出詳細的 density 等資訊的關係，可以應用在其他相關的任務上。但是他最大的缺點就是運算實在太慢了。相比 DVGO 這篇 paper 只需要用到十幾分鐘，NeRF 卻要用到十幾二十小時。

2. DVGO 的架構大致上和 NeRF 差不多，但是作者做了一些改進。首先，作者使用 voxel grid 來表示 density，這樣的話就可以很快地用插值法取得任意 xyz 的 density。但此方法的缺點是最佳化不易，因此作者使用 Mip-NeRF 作為 activation function，直接優化 voxel grid，並提出 post-activation 的概念，

在插值後才通過 activation function，如此一來可以產生出更 sharp 的面，比較好用 voxel grid 來表示。這裡作者再讓 voxel grid 初始在接近 0 的數值，而且用 sample 到某一點的次數來動態調整 learning rate，越常 sample 到 learning rate 越大，以上這些改進都是為了要更好的去 optimize 以 voxel grid 表示的方法。最後作者也用了 coarse、fine 兩個 model 分層產生結果，先用 coarse model 勾勒出大概的模樣，再用 fine model 去將整個畫面產生出來，這樣的設定同樣也加快了整體速度。DVGO 每訓練一段時間就會增加 voxel grid 的數量，提高解析度，另外在訓練過程中，能省略的 render 步驟就省略，比方說這個視角下，某些光線並不會打到任何物體，或者光線通過空的區域，那其實都是可以不用 sample 的，就直接省略就好。也因此，DVGO 才可以在基於 NeRF 的情況下，做出那麼大的進步。

3.

	PSNR	SSIM	LPIPS
Setting 1	35.1823	0.9744	0.0412
Setting 2	35.1897	0.9745	0.0413

Setting 1 is set with default hyperparamers in [sunset1995/DirectVoxGO: Direct voxel grid optimization for fast radiance field reconstruction. (github.com)]
Setting 2 is set the same as setting 1 except the based learning rate is 5e-3.

PSNR：將圖片可能的最大像素值除以兩張圖片的 MSE，然後以 dB 表示，所以可以看成一個常數減掉一個取對數的 MSE 值，也就是說 PSNR 越大越好。
SSIM：將 luminance、contrast、structure 這三個圖片要素透過各自的公式計算出兩張圖片的差異，通常都是取一個小 patch 計算，然後滑動 patch 在計算一次，滑完整張圖片取平均。兩張圖片的 std 和 mean 越接近，SSIM 越大，因此 SSIM 也是越大越好。
LPIPS：將圖片通過一個 model（我是使用 VGG），並將兩張圖片在各層的 output 彼此拿來比較計算距離，然後再乘上該層的權重，經過簡單的 normalize 之後得出。由於是計算距離，所以越小越好。

可以發覺 setting 1 和 setting 2 結果是差不多的，我猜測可能和 DVGO 有動態調整各個 voxel 的 learning rate 有關，所以些微的 based learning rate 差異並不會太影響到最終結果。不過還是可以觀察到 setting 2 稍微高一點，其實在 setting 1 的時候，train 完 20000 個 epochs 時並沒有收斂，可以看到 PSNR 仍在上升，所以如果把 learning rate 調高一點，應該是有機會讓他收斂更快的。雖然效果不明顯，但是仍然可以看出一點點端倪。

Problem 2

1. 我使用 BYOL 作為 SSL 的方法，在 backbone 的部分使用 torchvision.models 提供的 resnet50，並且只 pretrain 到 avg_pool layer，pretrain 了 100 個 epochs，optimizer 使用 Adam，learning rate 為 3e-4，beta 使用 pytorch 預設數值，batchsize 64，並且使用助教提供的 preprocess 方式做圖片處理，先 resize 到 128，再 center crop 成 128*128，然後用 ImageNet 的標準做 Normalization。

2.

Setting	Pre-training	Fine-tuning	Validation accuracy
A	-	Full model	0.2980
B	TA's backbone	Full model	0.3448
C	SSL pretrained	Full model	0.4064
D	TA's backbone	Fix backbone	0.1970
E	SSL pretrained	Fix backbone	0.2512

Mean of A~E: 0.2995

觀察可以發現，以上結果大概可以分成三個區段(B, C)、(A)、(D, E)。可以看到有使用 Full model 做 training 的效果明顯較好，我想這也是因為我只使用了一層 Linear layer 而且沒有加任何的 activation function，導致只有更新最後一層，model 並不足以學到足夠的東西。另外，(B, C)較(A)好，顯示 pretrain 還是很重要的，不管有沒有使用 label，至少都有幫助 model 去擷取到圖片中的信息。接著，C 比 B 好，E 比 D 好，顯示在這次作業中，不使用 label pretrain 才能達到比較好的效果，推測原因是因為 office 和 home 兩個資料集的分類問題並不相同，如果使用 label，那 model 就會為了要學會 home 的分類而擷取那些只有在 home 才會出現的圖片信息，然而擷取這些信息卻在 office 上面毫無用武之地，甚至可誤導 model。如果不使用 label，那 model 學會的只是如何分析任一張圖片，並沒有學會要做什麼特定的任務，就像小孩一樣，在還沒有特定的生活模式的時候，要怎麼培養或者教育都很容易，但是如果已經長大，有了自己的生活習慣，要強迫他學習什麼就變得比較困難。