HW6

311512064 鄧書桓

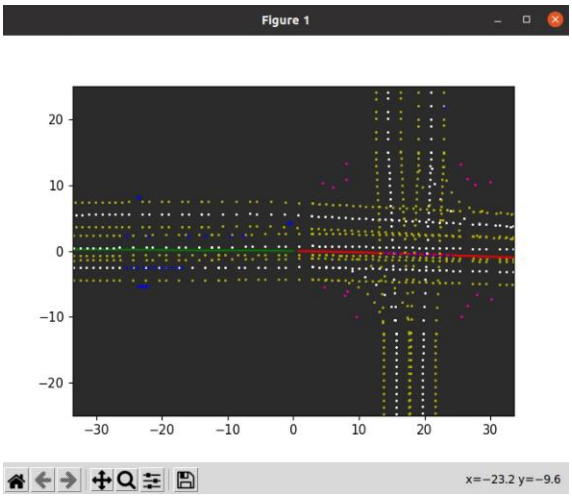
Part1:

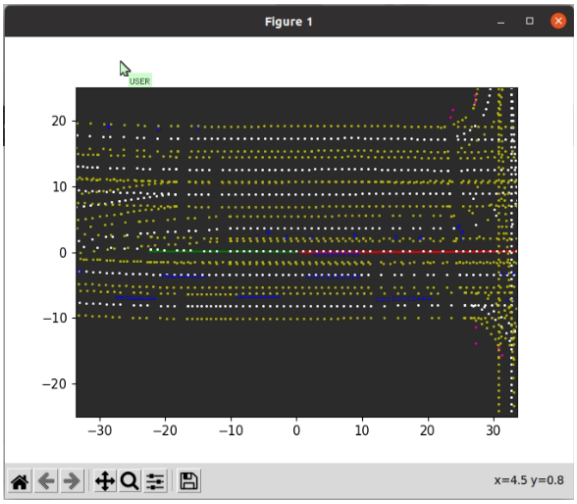
dataset 視覺化的截圖:

下面這段 code 為視覺化道路邊線、斑馬線與周圍物體軌跡(bonus)的部分，其中的那些維度大小是根據該變數.shape()去求得的。

```
#####Bonus!!!!!!!!!!!!!!!######################
# # orig , rot 的值不在範圍內, print不出來
''' neighbor_graph  '''
#shape(N, 11, 6)
neighbor_graph_temp = sample['neighbor_graph']
for i in range(11):
    plt.plot(neighbor_graph_temp[:, i, 0],
             neighbor_graph_temp[:, i, 1], "bo", markersize=1)

''' crosswalk_graph '''
#shape(N, 2, 2)
crosswalk_graph_temp = sample['crosswalk_graph']
for i in range(2):
    plt.plot(crosswalk_graph_temp[:, i, 0],
             crosswalk_graph_temp[:, i, 1], "rD", markersize=1)

#crosswalk_polygon與crosswalk_grap的x, y位置一樣
''' crosswalk_polygon   '''
#shape(N, 5 ,2)
crosswalk_polygon_temp = sample['crosswalk_polygon']
for i in range(5):
    plt.plot(crosswalk_polygon_temp[:, i, 0],
             crosswalk_polygon_temp[:, i, 1], "mo", markersize=1)

''' lane_polygon    '''
#shape(N, 21, 2)
lane_polygon_temp = sample['lane_polygon']
for i in range(21):
    plt.plot(lane_polygon_temp[:, i, 0],
             lane_polygon_temp[:, i, 1], "yo", markersize=1)
```

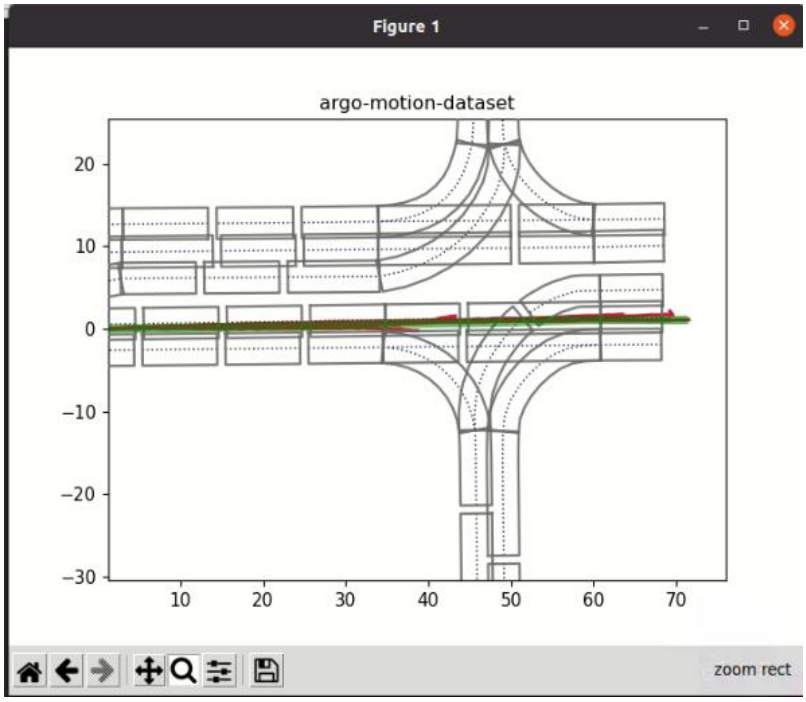
下方的 figure1 為可視化歷史軌跡、未來軌跡、道路中線、道路邊線、斑馬線與周圍物體軌跡的部分。另外，我本來想將檔案中的 orig 與 rot 一併顯示，但他們得值好像介於給定的 x, y 範圍外，所以無法顯示於 matplotlib 上。





Part2:

Argo 軌跡預測截圖:

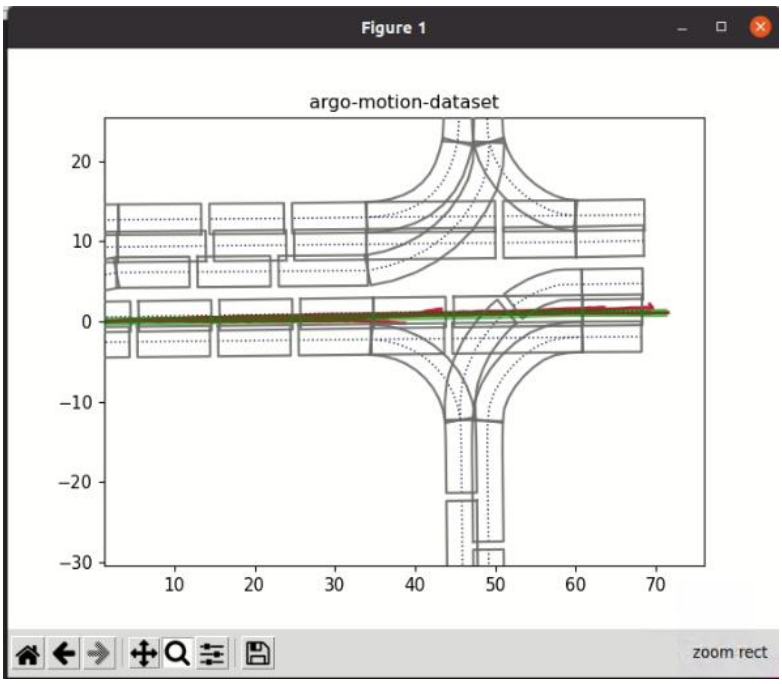


Model 設計方法:

由於本來的 network train 出來的效果就不錯，因此對於整體架構並沒有做過多更改，這邊僅在 MLP 隱藏層作些許調整，如更改其中幾個神經網路的大小，來使特徵擷取的效果變得更好。另外，我還有在 multithread 加入 nn 中的 normalization 以此來避免 over fitting，雖然本來的 network 中已經有使用 droupout 來改善了，但是特殊的情境(如轉彎)時的 overfitting 還是較嚴重。最後，我有將 epoch 改成 150，以此來提高在轉彎處的預測成功率，經過多次測試可知，當 epoch 調整到 85 以上時在過彎處才會有一組預測是成功的。

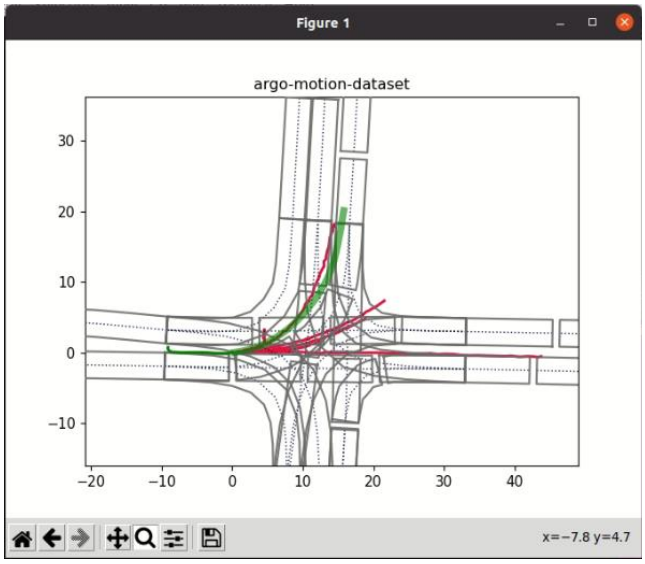
Case study:

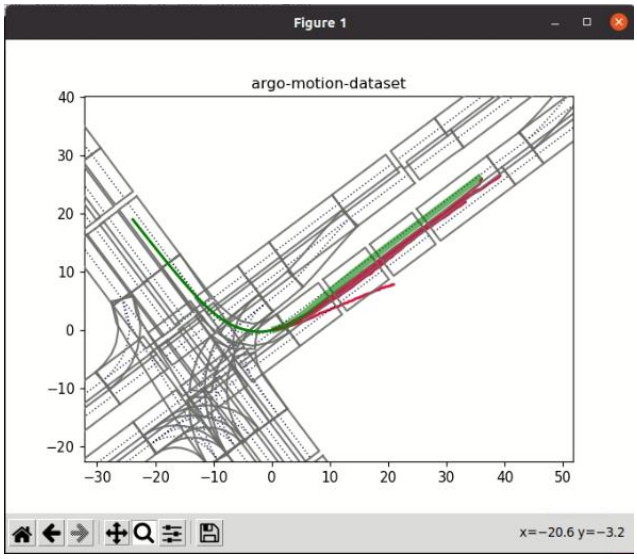
1. **直線:**



在直線的預測中，原本的 model 就預測得非常好，6 個預測軌跡幾乎吻合原先的 ground truth，而其中有一些會插出去的原因我認為是 overfitting 造成的(這邊的 epoch 設為 150)，會被環境中些許的極值資料影響，但若並非這樣設 epoch 的話，於後續彎道的預測將會慘不忍睹，因此並沒有再做調整。

2. **彎道:**





由左上截圖可知，在彎道時有 5 個預測的軌跡是跑偏掉的，僅有一個預測是成功的，我認為是因為彎道本身就比較難估計，而且我拿去 train 的資料與預測的資料僅有地圖與觀察 5 秒的資訊，對於彎道的資訊較少，無法藉由周遭的動態障礙物去協助判斷，因此只會有一個成功。另外，需要

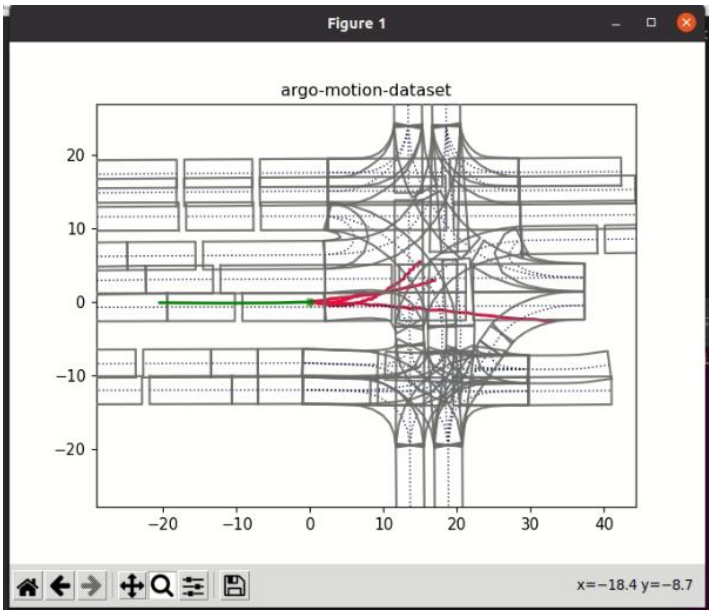
將 epoch 調整至 85 以上才會有一個預測的軌跡能夠符合 ground truth，雖然這樣有達成本次的目標，但是因為訓練集與輸入的資料是在相同環境下，因此 over fitting 的負面影響較小，但這樣的泛化性就極差，在光復路或是其他場景時的預測效果就會很不好。而右上方的截圖顯示預測的效果非常好，我認為是這邊的地圖資訊較充足，因此並不會因為是彎道而預測失敗。

另外，對於上述 overfitting 的現象，我有在 multithread 加入 nn 中的 normalization，以此將極值去除，使用的方式為下方第一個截圖。另外，我也有嘗試改變 MLP 中的通道數量，希望藉此能夠改善於彎道預測結果較差的問題，但效果沒有很好，改變的方式如下方第二章截圖。

```
self.norm = nn.LayerNorm(d_model)
```

```
class MLP(nn.Module):
    def __init__(self, in_dim, hidden_dim, out_dim):
        super(MLP, self).__init__()
        self.mlp = nn.Sequential(
            nn.Linear(in_dim, hidden_dim//2),
            nn.ReLU(),
            nn.Linear(hidden_dim//2, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, hidden_dim//2),
            nn.ReLU(),
            nn.Linear(hidden_dim//2, out_dim),
        )
```

3. 急停



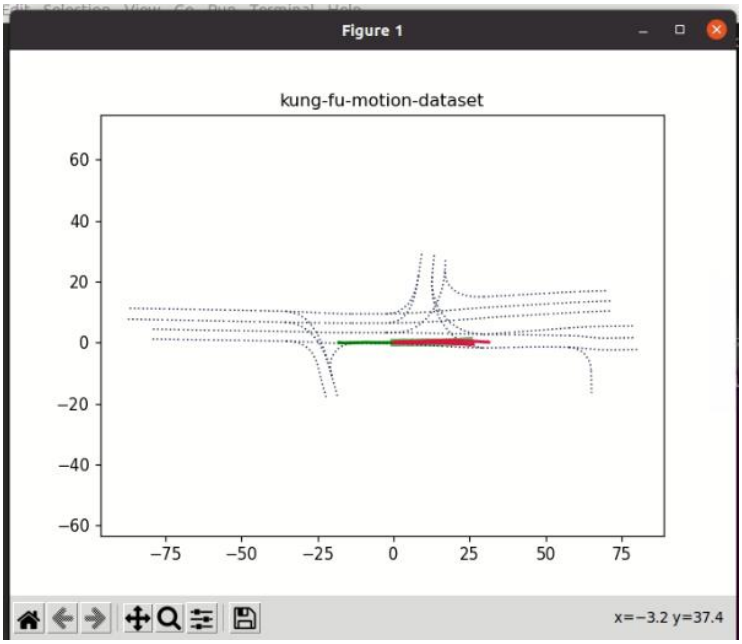
由上方的截圖可知，對於急煞的預測完全失敗，我認為是因為並沒有考量到周遭的動態物體，因此對於急停並沒有辦法預測出來。於此部分，我有設法將周遭動態資訊加進去做 train，但是一直失敗。我認為只要將周

遭動態物體的資訊也考慮進去，就能夠很好的完成急停的預測。

Part3:

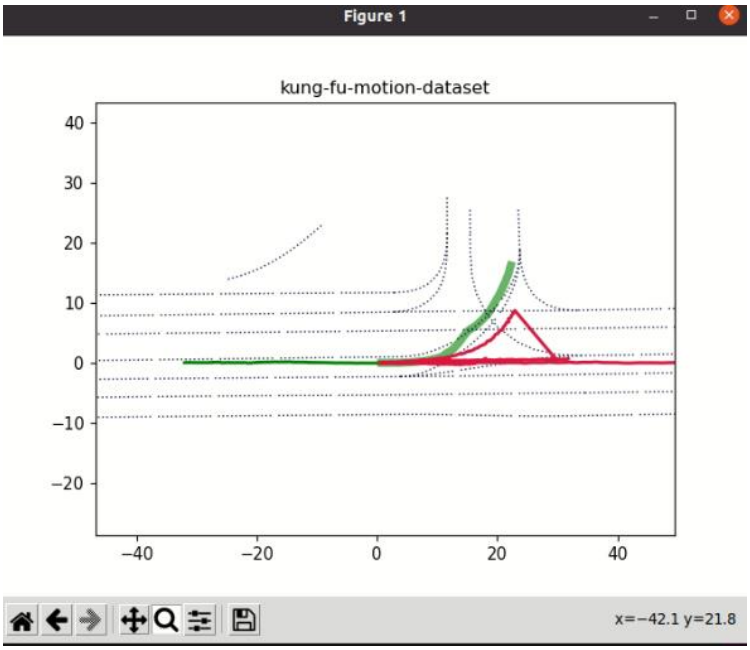
光復路軌跡預測截圖

1.直線:



由上方的圖可知，直線的預測非常完美，預測出來的軌跡與 ground truth 十分吻合，因此對於此部分沒有做過多的調整。

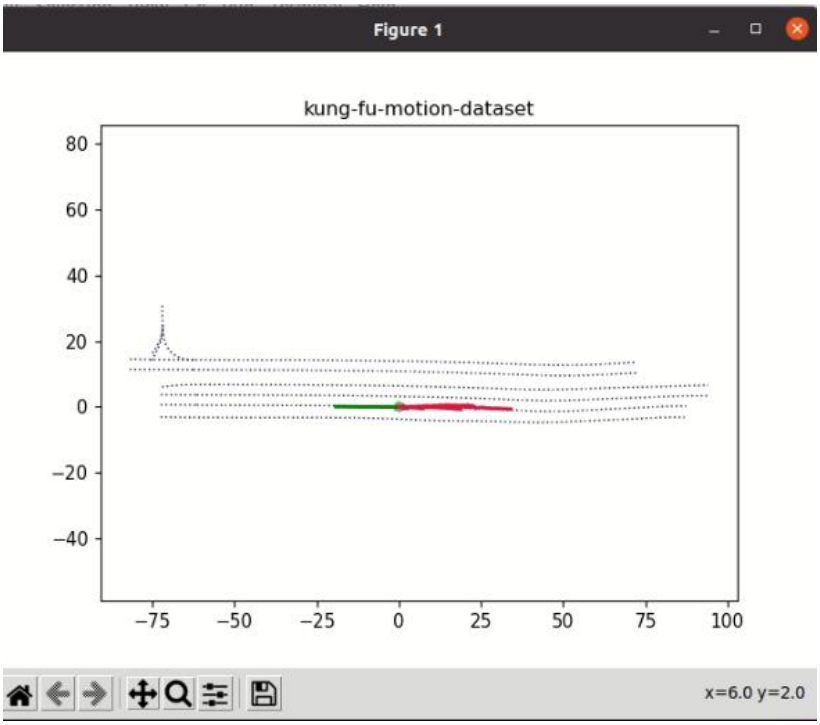
2.彎道:



由上方的截圖可知，於彎道時的預測完全失敗，我認為是因為訓練出來的模型已經 over fitting 的關係，此 model 僅能作用在 Argo 環境中。我有設法加入 normalization 與 新增 dropout 層來避免 overfitting，但是效果微乎其微(所用的 Dropout 與 normalization 在下方截圖所示)。我認為若要預測成功，應該要使用更能避免 overfitting 的方法，這樣可能要讀一下其他 paper 才有辦法成功。

```
self.norm = nn.LayerNorm(d_model)
self.dropout = nn.Dropout(dropout)
```

3.急煞



這邊的裝況和在 part2 的狀況一模一樣，由於沒有將周遭動態資訊給考慮進去，才會造成此問題，我認為只要將周遭動態物體的資訊也考慮進去，就能夠很好的完成急停的預測。