Self-Driving Cars

309611087 洪得瑜

2022 10/20

1 Part 1 Motion Dataset

Matplotlib 視覺化場景:

- 歷史軌跡: 綠色
- 未來軌跡: 紅色
- 道路中線: ”- -” 白色
- 道路邊線: 灰色
- 斑馬線: 紫色
- 周圍動態軌跡: 藍色

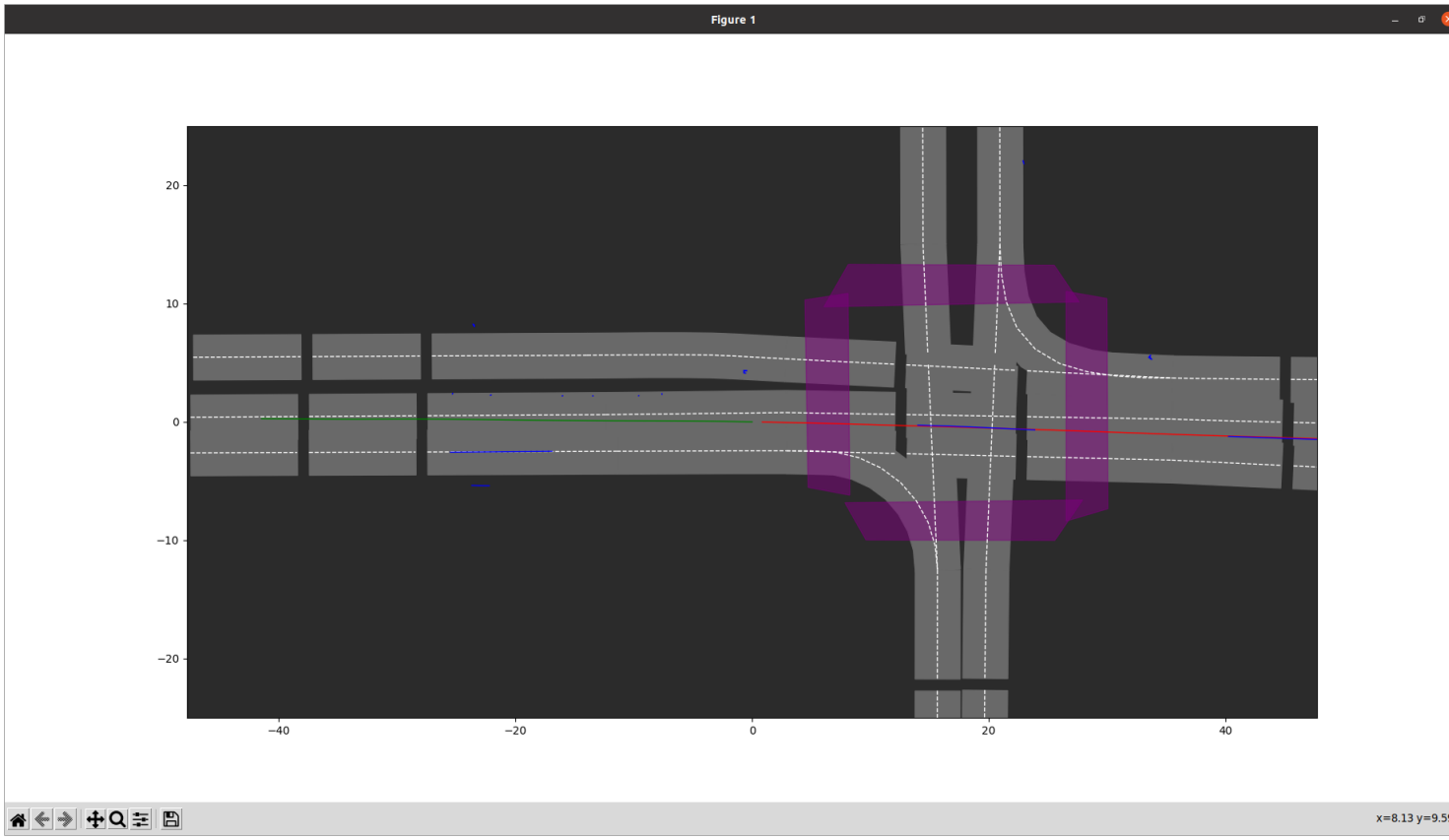


Figure 1: Dataset 場景圖

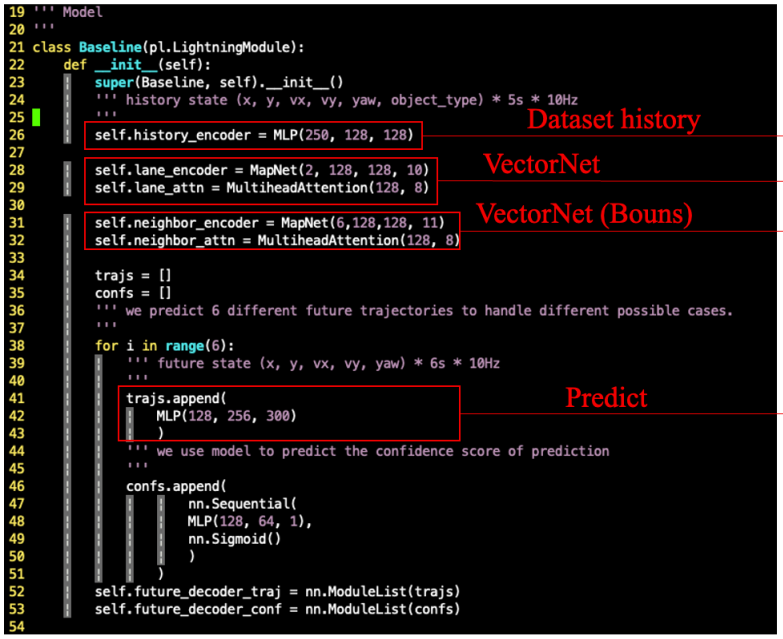
1.1 Code

```
556 def plot_argo_scenario(sample):
557     fig = plt.figure()
558     ax = fig.add_subplot(111)
559     ax.set_facecolor('#2b2b2b')
560 
561     ''' History Trajectory '''
562     x = sample['x'].reshape(_OBS_STEPS, 6)
563     ax.plot(x[:,0],
564             x[:,1],
565             "-",
566             color='g',
567             alpha=1,
568             linewidth=1
569             )
570     ''' Future Trajectory '''
571     y = sample['y'].reshape(_PRED_STEPS, 5)
572     ax.plot(y[:,0],
573         y[:,1],
574         "-",
575         color='r',
576         alpha=1,
577         linewidth=1
578         )
579 
580     ''' Lane Centerline '''
581     lane = sample['lane_graph'].reshape(-1, 10, 2)
582     for i in range(len(lane)):
583         ax.plot(lane[i][:,0],
584             lane[i][:,1],
585             "--",
586             color='w',
587             alpha=1,
588             linewidth=1
589             )
590     ''' neighbor_graph '''
591     neighbor_lane = sample['neighbor_graph'].reshape(-1,11,6)
592     for i in range(len(neighbor_lane)):
593         ax.plot(neighbor_lane[i][:,0],
594                 neighbor_lane[i][:,1],
595                 "-",
596                 color='b',
597                 alpha=1,
598                 linewidth=1,
599                 )
```

```
600     ''' lane '''
601     lane_polygon = sample['lane_polygon'].reshape(-1, 21, 2)
602     for i in range(len(lane_polygon)):
603         polygon = Polygon(lane_polygon[i], closed=True)
604         collection = PatchCollection([polygon],color='dimgray')
605         ax.add_collection(collection)
606 
607     ''' cross walk '''
608     crosswalk_polygon = sample['crosswalk_polygon'].reshape(-1,5,2)
609     for i in range(len(crosswalk_polygon)):
610         polygon = Polygon(crosswalk_polygon[i], closed=True)
611         collection = PatchCollection([polygon], color = 'purple', alpha=0.5)
612         ax.add_collection(collection)
```

2 Part 2

Model Design: Model 中 Baseline 參考文獻 [1][2] 設計，建立高精地圖特徵 MapNet[1] 使用向量化地圖建構車道圖，而文獻 [1] 中以 LaneGCN 方法輸出地圖特徵，結合周遭動態特徵及車道特徵進行動態預測，而在本次車道與周圍關係以文獻 [2]VectorNet 方法將高經地圖特徵與車道及周圍動態融合，VectorNet 將同一元素以向量組成行成一條 poyline 將多個 polylines 行成 Subgraph，將所有軌跡和地圖特徵全局分析。

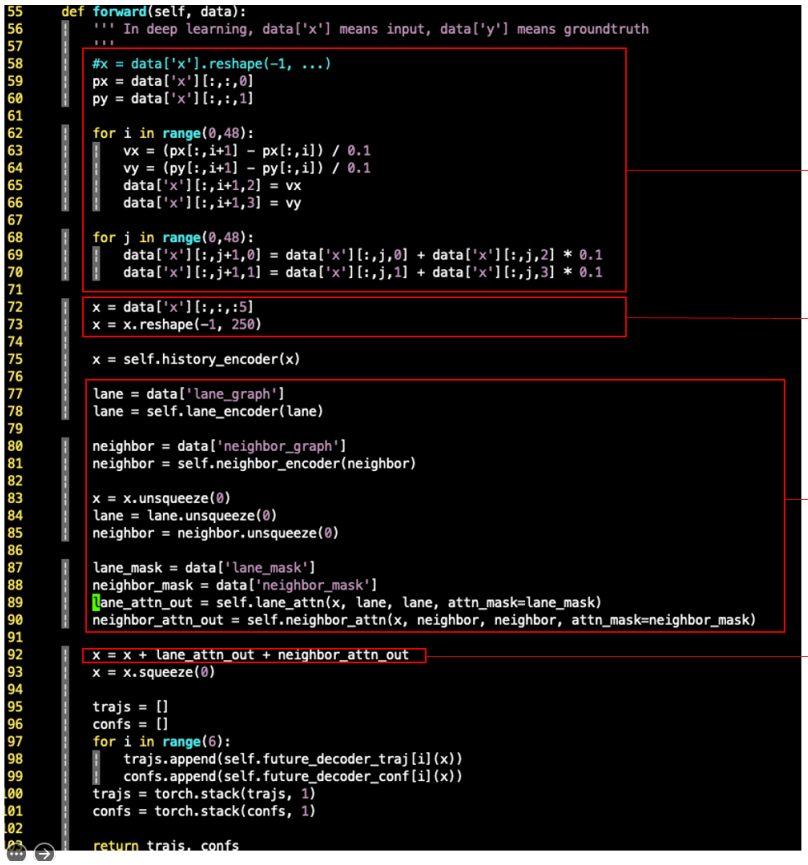



```python
''' Model
'''
class Baseline(pl.LightningModule):
    def __init__(self):
        super(Baseline, self).__init__()
        ''' history state (x, y, vx, vy, yaw, object_type) * 5s * 10Hz
        '''
        self.history_encoder = MLP(250, 128, 128)

        self.lane_encoder = MapNet(2, 128, 128, 10)
        self.lane_attn = MultiheadAttention(128, 8)

        self.neighbor_encoder = MapNet(6,128,128, 11)
        self.neighbor_attn = MultiheadAttention(128, 8)

        trajs = []
        confs = []
        ''' we predict 6 different future trajectories to handle different possible cases.
        '''
        for i in range(6):
            ''' future state (x, y, vx, vy, yaw) * 6s * 10Hz
            '''
            trajs.append(
                MLP(128, 256, 300)
                )
            ''' we use model to predict the confidence score of prediction
            '''
            confs.append(
                    nn.Sequential(
                    MLP(128, 64, 1),
                    nn.Sigmoid()
                    )
                )
        self.future_decoder_traj = nn.ModuleList(trajs)
        self.future_decoder_conf = nn.ModuleList(confs)
```




```python
    def forward(self, data):
        ''' In deep learning, data['x'] means input, data['y'] means groundtruth
        '''
        #x = data['x'].reshape(-1, ...)
        px = data['x'][:,:,0]
        py = data['x'][:,:,1]

        for i in range(0,48):
            vx = (px[:,i+1] - px[:,i]) / 0.1
            vy = (py[:,i+1] - py[:,i]) / 0.1
            data['x'][:,i+1,2] = vx
            data['x'][:,i+1,3] = vy

        for j in range(0,48):
            data['x'][:,j+1,0] = data['x'][:,j,0] + data['x'][:,j,2] * 0.1
            data['x'][:,j+1,1] = data['x'][:,j,1] + data['x'][:,j,3] * 0.1

        x = data['x'][:,:,:5]
        x = x.reshape(-1, 250)

        x = self.history_encoder(x)

        lane = data['lane_graph']
        lane = self.lane_encoder(lane)

        neighbor = data['neighbor_graph']
        neighbor = self.neighbor_encoder(neighbor)

        x = x.unsqueeze(0)
        lane = lane.unsqueeze(0)
        neighbor = neighbor.unsqueeze(0)

        lane_mask = data['lane_mask']
        neighbor_mask = data['neighbor_mask']
        lane_attn_out = self.lane_attn(x, lane, lane, attn_mask=lane_mask)
        neighbor_attn_out = self.neighbor_attn(x, neighbor, neighbor, attn_mask=neighbor_mask)

        x = x + lane_attn_out + neighbor_attn_out
        x = x.squeeze(0)

        trajs = []
        confs = []
        for i in range(6):
            trajs.append(self.future_decoder_traj[i](x))
            confs.append(self.future_decoder_conf[i](x))
        trajs = torch.stack(trajs, 1)
        confs = torch.stack(confs, 1)

        return trajs, confs
```

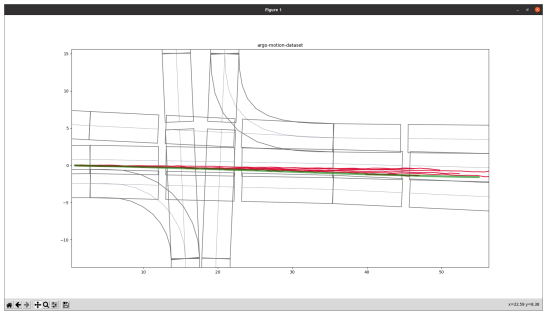
2.1 Case study

Argo Dataset: 以兩種方法調整解決路徑預測不佳問題

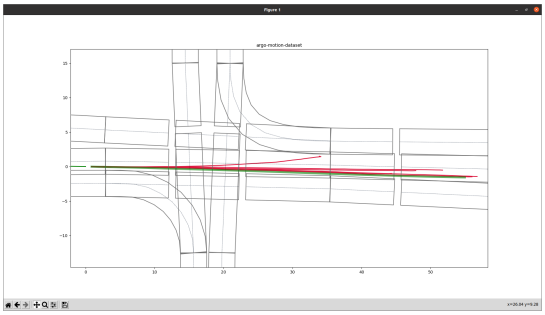
- 可增加 Epoch 次數可降低 train loss 而在 ADE(Average Displacement Error) 及 FDE(Final Displacement Error) 也可逐步降低。
- 而在加 Neighbor 的動態軌跡時有提高訓練收斂速度，可在較少 Epoch 達到最好的路徑。

2.1.1 直線預測

可成功預測出直線路徑，但 Epoch100 預測出直線有波浪現象因此多增加訓練次數效果較佳



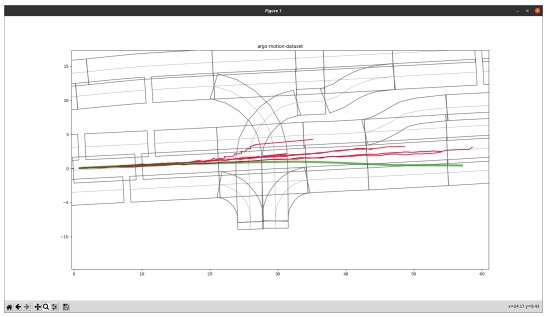
(a) epoch 100



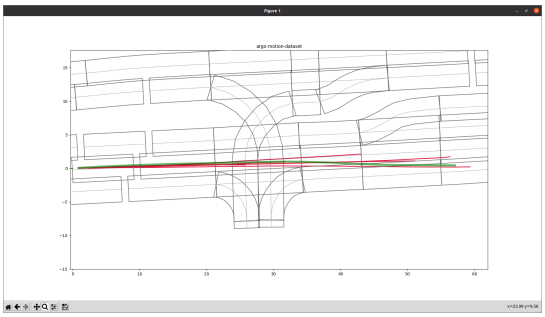
(b) epoch 1000

2.1.2 變換車道

100Epoch 無法正確預測變換車道結果，但經由增加 Epoch 可成功預測出結果。



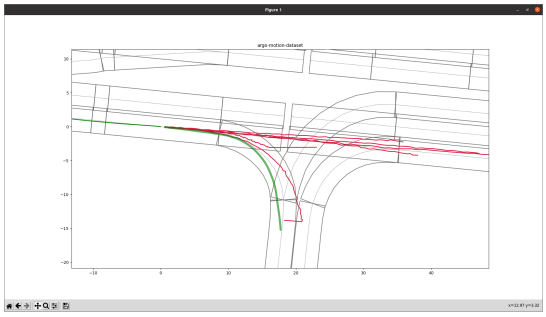
(c) epoch 100



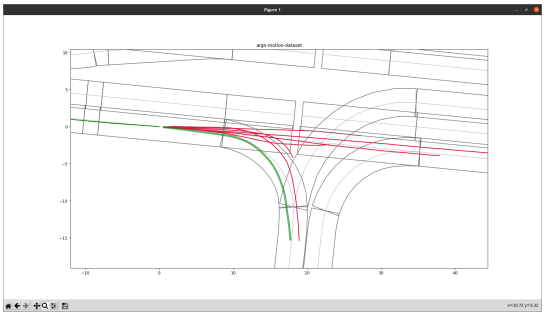
(d) epoch 1000

2.1.3 車輛轉彎預測

在 100epoch 及 1000epoch 都能預測出轉彎路徑, 但 1000epoch 效果較好



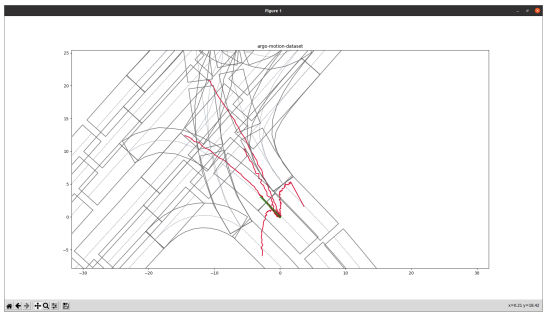
(e) epoch 100



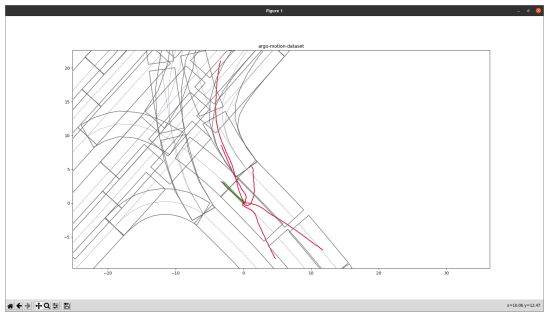
(f) epoch 1000

2.1.4 停止短路徑預測

由於歷史資料少或是停止所造成路徑預測效果不佳, 提高 Epoch 能增加預測結果。



(g) epoch 100



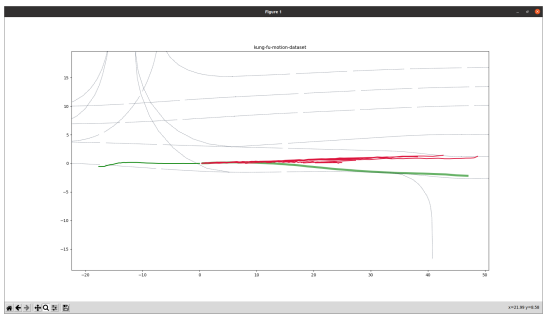
(h) epoch 1000

3 Kung Fu

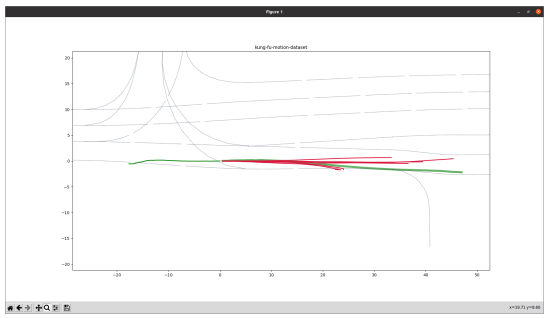
經由 Argo 訓練好的模型套用到光復路資料集做測試，經過實驗測試結果顯示同樣的模型套用的效果跟 Argo Dataset 相比略有差距，對於直線預測可以正常預測，但變換車道可能預測並不是相當好，而在轉彎處及停止可正確預測出。

3.1 變換車道

Epoch100 無法預測出變換車道，而在 1000 epoch 已預測出切換車道但不完整可能需要在增加 Epoch 效果應該能更好



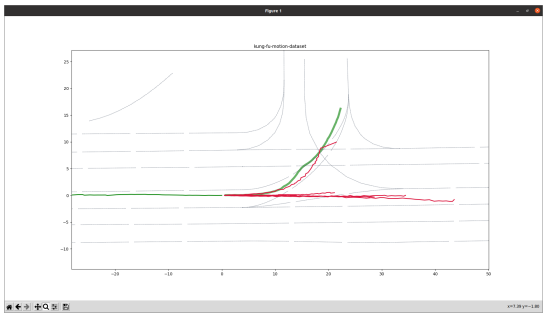
(i) epoch 100



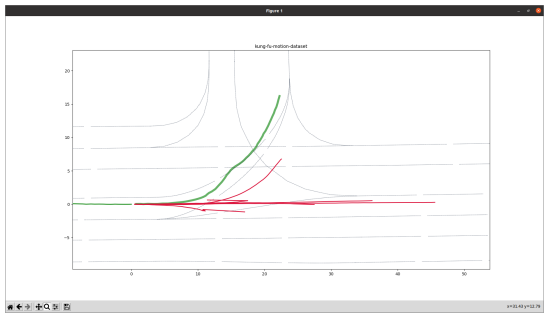
(j) epoch 1000

3.2 車輛轉彎

相比之下 100 epoch 卻比起 1000 epoch 預測的更好，可得到並非訓練多次效果會越好。



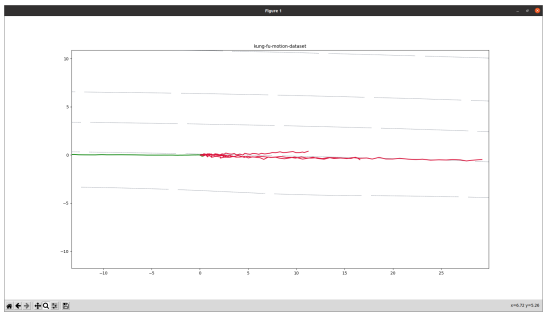
(k) epoch 100



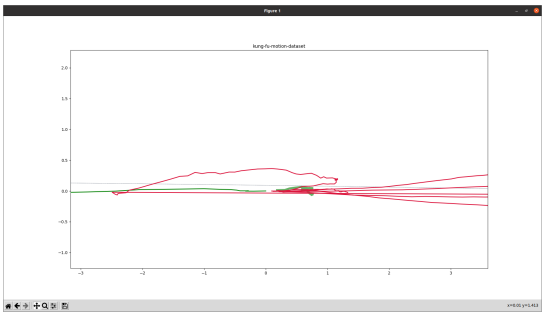
(l) epoch 1000

3.3 車輛停止

車輛停止在 1000 epoch 有成功預測出，但 100epoch 未能預測出車輛停止



(m) epoch 100



(n) epoch 1000

References

[1] Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R. (2020, August). Learning lane graph representations for motion forecasting. In European Conference on Computer Vision (pp. 541-556). Springer, Cham.

[2] Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C. (2020). Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11525-11533).