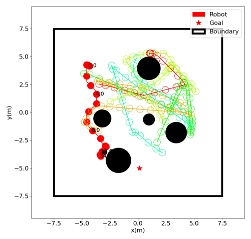
Final Report: Social Navigation with RL

Yun-Jin Li
Technical University Munich
MSc. Robotics, Cognition & Intelligence
Munich, Germany
yunjin.li@tum.de

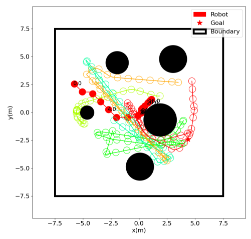
Kagan Kücükaytekin
Technical University Munich
MSc. Robotics, Cognition & Intelligence
Munich, Germany
kagan.kuecuekaytekin@tum.de

I. Introduction

We envision a future with safe, interactive robots which can co-exist with people. For this reason we chose the topic ”Social Navigation”. Social navigation is the type of navigation, during which the agent aims to avoid conflicts with pedestrians in the environment while navigating towards its goal. SARL, the state of the art method proposed by Chen et. al. [1], explores this problem in a simple environment without any obstacles. In our work, we investigate this problem further under more challenging conditions, explore the challenges, and share our insights in overcoming them.



(a) Freezing robot (1)



(b) Freezing robot (2)

Fig. 1: Examples of freezing robot problem

II. Challenges

We refer the reader to our milestone report for an investigation of the baseline method and our preliminary efforts on adding complexity to the simulation environment. To summarize, we extend the baseline environment with:

- Static obstacles with variable-radius,
- Randomly generated start and goal position for the robot
- A new goal position for the pedestrians once they reach their goal

After these extensions, we observed that our trained agent often got stuck in intermediate configurations, resulting in the so-called ”freezing robot problem” as shown in figure 2. Trautman and Krause explain this problem as follows: ”Once the environment surpasses a certain level of complexity, the planner decides that all forward paths are unsafe, and the robot freezes in place (or performs unnecessary maneuvers) to avoid collisions [2].” For the purpose of dealing with this problem, we propose two approaches.

1) Changing the training strategy: Curriculum learning
2) New architecture: Graph attention network for social navigation (GAT4SN)

III. Curriculum Learning

During our experiments with the baseline method, we discovered that setting the maximum radius of the static obstacles close to the radius of the pedestrians results in successful training, while setting the maximum radius higher leads to an increasing presence of the freezing robot problem. If the obstacle radius is large enough, the model fails to learn the task entirely. In [3] and [4] it was shown that curriculum learning can allow models to learn complex tasks which are otherwise failed in a reinforcement learning setting. This inspired us to apply curriculum learning for our task as well.

Curriculum learning is a training strategy, in which the training data is presented to the model from easier to harder examples. A curriculum learning setting consists of a difficulty measurer, which serves to evaluate the difficulty of the training samples, and a training scheduler, which selects the samples used for the training [5]. In our project, both of these components are hand-crafted, categorizing our setting as ”Predefined CL”.

In our setting, we measure the difficulty by the radii of the obstacles in the environment. Training is scheduled by increasing the obstacle radii by a predefined amount once a particular success rate is achieved. Our training scheduler is inspired by the discrete scheduler called ”Baby Step Scheduler” proposed by Bengio et. al. [6]. In this scheduler, the training dataset is divided into buckets of increasing difficulty. As the model reaches a certain performance, more difficult buckets are included in the training dataset. To avoid catastrophic forgetting, the easier buckets are always kept in the dataset. The alternative approach called ”One-Pass Scheduler”, in which the easiest samples are thrown away, yields worse results according to empirical studies [6].

In our deep value learning setting, we use a memory buffer with a fixed capacity as our data source. After the buffer reaches its full capacity, the oldest samples are thrown away to make space for new samples. This prevents us from keeping the easiest samples in the dataset. To remedy this problem, we incorporate the easier aspects of the old data within the new data. Particularly in our case, we set only a portion

Algorithm 1 Curriculum Learning for Social Navigation

```
**Input:** Model $\mathcal{M}$, Memory buffer $\mathcal{R}$, Simulator $S$, Number of pedestrians and obstacles $N_p$, $N_o$, Radius of pedestrians $r_p$, Maximum obstacle radius $r_{obst.max}$, Radius increment for obstacles $r_\Delta$, Hardest obstacle ratio $\gamma_{hard}$ Target success rate $SR_{target}$
**Output:** Trained Model $\mathcal{M}^*$
$r_{min} \leftarrow r_p$, $r_{max} \leftarrow r_p$
**repeat**
  $S.addPedestrians(N_p)$
  **for** $i = 1$ **to** $N_o$ **do**
    **if** $i \leq \gamma_{hard} \times N_o$ **then**
      $S.addObstacle(r_{max})$
    **else**
      $S.addObstacle(sampleUniform(r_{min}, r_{max}))$
    **end if**
  **end for**
  $\{(s, a, r, s')\}_\tau = S.simulate()$
  $\mathcal{R}.addSamples((s, a, r, s')\}_\tau)$
  $\mathcal{M}.train(\mathcal{R})$
  **if** $\mathcal{M}.successRate() \geq SR_{target}$ **and** $r_{max} < r_{obst.max}$ **then**
    $r_{max} \leftarrow r_{max} + r_\Delta$
  **end if**
  $S.reset()$
**until** convergence
**return** $\mathcal{M}$
```

of the obstacles with the maximum radius of the current level, while sampling the radii of the other obstacles from the uniform distribution between minimum and maximum radii. This approach allows to increase both the diversity and the complexity of the training samples. In contrast to baby step scheduler, where chunks of harder data is included in the training dataset at once, we add one new harder example to our memory buffer at every training iteration. This makes the transition between difficulties smoother and in effect turns our training scheduler into a continuous version of the baby step scheduler.

IV. GAT4SN

A. Motivation

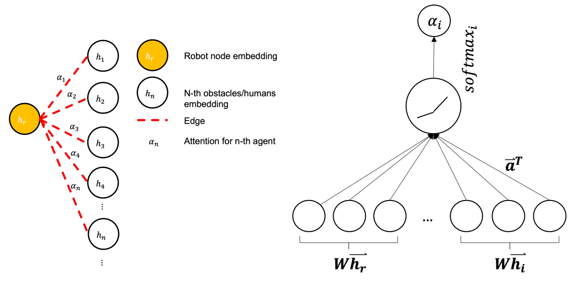
Taking a different perspective to our problem, if we consider the interaction between the robot and all the other agents as edges, we can build up a graph in the environment. In other words, the social navigation problem can be reformulated as a graph problem. With that being said, we can apply state-of-the-art graph neural network architectures to solve this problem.

B. Architecture

1) Graph attention network: We refer to the architecture, Graph Attention Network from Veličković [7] et. al. Basically, our problem can be represented as a single node (robot) updating problem where α_n serves as the edge embedding and it tells us how important this agent is to the robot. The attention mechanism $att(*)$ is defined as follow:

$$\boldsymbol{H} = att(\begin{bmatrix} \vec{\boldsymbol{h}_r} \\ \vec{\boldsymbol{h}_1} \\ \vdots \\ \vec{\boldsymbol{h}_N} \end{bmatrix}) = \begin{bmatrix} \boldsymbol{W}\vec{\boldsymbol{h}_r} \\ \alpha_1 \boldsymbol{W}\vec{\boldsymbol{h}_1} \\ \vdots \\ \alpha_N \boldsymbol{W}\vec{\boldsymbol{h}_N} \end{bmatrix} \tag{1}$$

$$\begin{aligned} \alpha_i &= softmax_i(LeakyReLU(\vec{\boldsymbol{a}}^T[\boldsymbol{W}\vec{\boldsymbol{h}_r}||\boldsymbol{W}\vec{\boldsymbol{h}_i}])) \\ &= \frac{exp(LeakyReLU(\vec{\boldsymbol{a}}^T[\boldsymbol{W}\vec{\boldsymbol{h}_r}||\boldsymbol{W}\vec{\boldsymbol{h}_i}]))}{\sum_{n=1}^N exp(LeakyReLU(\vec{\boldsymbol{a}}^T[\boldsymbol{W}\vec{\boldsymbol{h}_r}||\boldsymbol{W}\vec{\boldsymbol{h}_n}]))} \end{aligned} \tag{2}$$



(a) Graph structure (b) Attention mechanism

Fig. 2: **Left**: Graph structure of our use case. **Right**: Attention mechanism for each robot-agent pair.

Moreover, instead of using only single attention mechanism, we decide to use multi-head graph attentional layer (see figure 3) as suggested in [7]. Basically, we compute the attention mechanism for multiple times and then concatenate all the output column-wise together as shown in equation 3.

$$\begin{aligned} \boldsymbol{H^K} &= ||_{k=1}^K[\boldsymbol{H^k}] \\ &= \begin{bmatrix} \boldsymbol{H^1} & \boldsymbol{H^2} & \boldsymbol{H^{K-1}} & \boldsymbol{H^K} \end{bmatrix} \end{aligned} \tag{3}$$

Furthermore, we put the concatenated matrix into output attention mechanism which serves as an averaging function in the whole process. Last but not least, we extract only the meaningful features from the output of the multi-head graph attentional layer to get the vector $\vec{\boldsymbol{h}^{\boldsymbol{G}}}$.

$$\begin{gathered} \boldsymbol{H^{out}} = att_{out}(\boldsymbol{H^K}) = \begin{bmatrix} \vec{\boldsymbol{h}_r^{out}} \\ \vec{\boldsymbol{h}_1^{out}} \\ \vdots \\ \vec{\boldsymbol{h}_N^{out}} \end{bmatrix} \\ \vec{\boldsymbol{h}^{\boldsymbol{G}}} = [\vec{\boldsymbol{h}_{\boldsymbol{r}}^{\boldsymbol{out}}}||\sum_{i=1}^N \vec{\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{out}}}] \end{gathered} \tag{4}$$

2) Deep Value Network: We combine our graph attention network with deep value network proposed by Chen et. al. [1] as shown in figure 3. The calculated value represents how good an input state is (s_r and all the other $s_i^{h/o}$). This information is later utilized to determine the robot's next action in order to maximize the cumulative rewards.

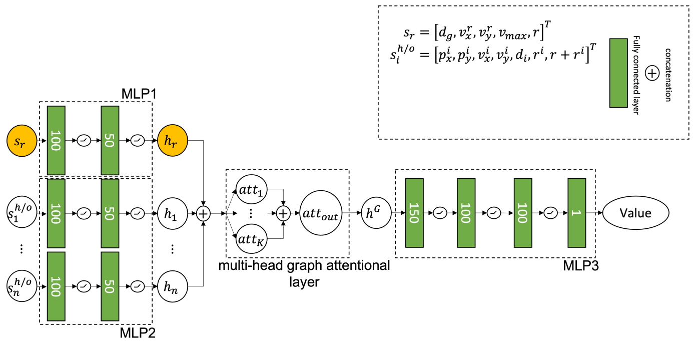

$s_r = [d_g, v_x^r, v_y^r, v_{max}, r]^T$
$s_i^{h/o} = [p_x^i, p_y^i, v_x^i, v_y^i, d_i, r^i, r + r^i]^T$
Fully connected layer
concatenation
MLP1
s_r
100
50
h_r
$s_1^{h/o}$
$s_n^{h/o}$
h_1
h_n
att_1
att_K
att_{out}
h^G
150
100
100
1
Value
multi-head graph attentional layer
MLP3
MLP2

Fig. 3: Deep Value Network (GAT4SN)

V. Result

A. Curriculum Learning

We run our curriculum learning training algorithm 1 on SARL with $N_p = 5$, $N_o = 10$, $r_p = 0.3$, $r_{obst.max} = 2.5$, $r_\triangle = 0.2$, $\gamma_{hard} = 0.2$, $SR_{target} = 0.7$ for 25000 episodes and provide the test results in 500 unseen environments in table I. It can be seen that, curriculum learning allows to learn to navigate in complex environments which are otherwise impossible. Additionally, the robot does not forget the easier levels and perform even better in these cases.

Model	Max. Obstacle Radius	Success Rate
SARL with CL	1.0	0.95
SARL with CL	1.5	0.83
SARL with CL	2.0	0.73
SARL with CL	2.5	0.64
SARL without CL	2.5	0.05

TABLE I: Performance of the robot trained with curriculum learning with $r_{obst.max} = 2.5$ on different difficulties vs. robot trained without curriculum learning

B. GAT4SN

1) Quantitative evaluation: We train the agent with 5 static obstacle and 5 pedestrians (both have fixed radius 0.3). Firstly, we use imitation learning with 3000 episodes of demonstrations generated by ORCA. Then, we train both GAT4SN with different number of attention heads and SARL for 10000 episodes to compare their performances.

As we can see from table II, even though SARL performs slightly better among the others in simple environment, our model can perform better than SARL in the hard environment setting. This implies that our model can learn a more generalized policy than SARL (as also shown in figure 4).

2) Safer policy: In order to demonstrate how well GAT4SN can learn a more generalized policy compared with SARL, we train both models again with 5 pedestrians and 10 obstacles (Both fixed radius 0.3). Then, we evaluate their performance in the environment containing 20 pedestrians. According to table III, our model tends to sacrifice the navigation time by following a safer policy, which leads to lower collision rate and higher success rate. However, SARL, on the contrary, saves time by following a more dangerous policy, driving directly into the crowd towards the goal position. This again shows that our model is more likely to perform better in real-life usage compared to SARL. Please refer to our repository https://github.com/kkaytekin/tum-adlr-ss22-05 for video examples.

	GAT4SN				SARL
Number of attention heads	1	2	3	4	–
Number of parameters	**66051**	72401	78751	85101	96502
Evaluation (simple environment / 500 cases)					
Success	0.994	0.992	0.978	0.996	**0.998**
Collision	0.002	**0.000**	0.002	0.002	**0.000**
Nav. time	11.96	12.20	12.39	11.61	**11.46**
Evaluation (hard environment / 500 cases)					
Success	0.982	**0.992**	0.954	0.972	0.958
Collision	0.014	**0.004**	0.010	0.014	0.018
Nav. time	**12.74**	13.01	15.00	12.75	12.88

TABLE II: Quantitative evaluation: **Simple**: Same environment setting as training. **Hard**: Static obstacle with variable radius $\in [0.5, 1.5]$.

Method	Success	Collision	Time
GAT4SN (Ours)	**0.972**	**0.022**	16.989
SARL (Baseline)	0.960	0.040	**13.645**

TABLE III: 20 humans environment result

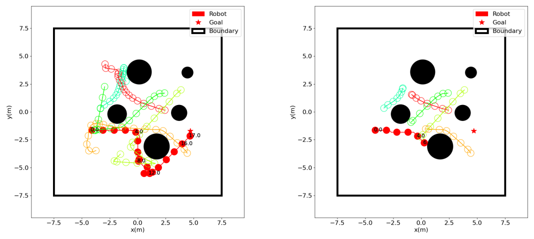


Fig. 4: Comparison of GAT4SN and SARL acting in the same environment. **Left**: GAT4SN tends to pass through the obstacle instead of running into it. **Right**: SARL tend to run into the obstacle and never move again

VI. Conclusion

Social navigation is an important milestone towards interactive mobile robotics. In our work we aimed to enable social navigation in more realistic environments with higher complexity. We demonstrated that a more sophisticated architecture or a more sophisticated training strategy can allow to obtain much better results in such complex settings.

This project can be further developed by making a hyperparameter optimization for our architecture GAT4SN. With better choice of hyperparameters, the model might be able to perform better than the baseline in all settings.

References

[1] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[2] P. Trautman, "Robot navigation in dense crowds: Statistical models and experimental studies of human robot cooperation," Ph.D. dissertation, California Institute of Technology, 2012.

[3] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," 2017. [Online]. Available: https://arxiv.org/abs/1705.06366

[4] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1707.05300

[5] X. Wang, Y. Chen, and W. Zhu, "A comprehensive survey on curriculum learning," *CoRR*, vol. abs/2010.13166, 2020. [Online]. Available: https://arxiv.org/abs/2010.13166

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 41–48. [Online]. Available: https://doi.org/10.1145/1553374.1553380

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.