Deep Learning-Based Pedestrian Simulation with Limited Real-World Training Data: An Evaluation Framework

Vahid Mahzoon*, Abbey Liu*, Slobodan Vucetic
Temple University
{vahid.mahzoon, abigail.liu, vucetic}@temple.edu,

A Code and Data Availability

The code and data is available at https://github.com/vmahzoon76/DL-Crowd-Sim.

B Knowledge-based Simulators

B.1 ORCA

Optimal Reciprocal Collision Avoidance (ORCA) guarantees collision-free trajectories for multiple agents. It was originally created with mobile robots in mind, where multiple agents need to avoid collisions with each other and static obstacles in the environment. Each agent a_i has a preferred velocity $v_{a_i}^{pref}$, which is used to guide an agent towards its goal location. In an environment without static obstacles, an agent's preferred velocity would point directly towards its goal. In scenarios with obstacles, the preferred velocity may point towards some sub-goal to avoid the obstacle during its trajectory path. ORCA first finds a set of velocities such that the agent a_i will not collide with any other agents during a set time horizon τ. This set of velocities is denoted $ORCA_{a_i}^{\tau}$. From within this set, the algorithm then selects the velocity that is closest to the preferred velocity $v_{a_i}^{pref}$. We mathematically define the collision-avoidance velocity v_{a-i}^{+} as

$$v_{a-i}^{+} = \underset{v \in ORCA_{a_i}^{\tau}}{\operatorname{argmin}} \|v - v_{a_i}^{pref}\|. \tag{1}$$

All agents in ORCA are modeled as a disk with some specified radius. There are several parameters that can be set to tune this mathematical model for different agents or scenarios. The parameters that we focus on are listed in Table 1 along with a brief description for each. We implement ORCA with the RVO2 simulator[1].

B.2 SFM

Social Force Model (SFM) is a mathematical model that considers each agent as a particle that interacts with other agents through social forces. It leverages Newton's second law to control the behavior of n agents in the following manner:

$$m\frac{d^2\vec{p}_i}{dt^2} = \vec{f}_i^{d} + \sum_{j \neq i} \vec{f}_{ij}^{\text{social}}, \quad \text{for } i = 1, \ldots, n \tag{2}$$

where f_i^d is the desired force which pushes an agent towards a predefined goal location. f_{ij} is the social force between each pair of agents i and j, taking into account collision avoidance or clustering behavior in crowds. Then, using a numerical method, we can solve the model (2) and obtain the position of each pedestrian at the next time step. Similar to the ORCA, agent is modeled as a disk with some specified radius. The implementation of SFM in our study is based on a Github repository[2]. There are several parameters that can be set to tune this mathematical model for different agents or scenarios. The parameters that we focus on are listed in Table 2.

B.3 Parameter Calibration Details

For ORCA, we test parameters NeighborDist, TimeHorizon, and TimeHorizonObst with integer values 1 to 5. Rather than test MaxSpeed directly, we test prefered speed (the magnitude of preferred velocity which points towards an agent's goal location) with values 1.0 to 1.3 in 0.1 value increments, and later set the MaxSpeed to preferred speed + 0.5. We keep MaxNeighbors at 30 because that is the highest number of pedestrians simultaneously present in a time step for all real world datasets (see section A.4 for more information). We also keep Radius set to 0.2, since the average shoulder width of an average adult is 0.4m. The time step is set to 0.4s to match the time step of real world datasets. For the SFM, we evaluated several parameters by varying their values: Max Speed and Max Speed Multiplier were tested across a range from 1.1 to 1.6 in 0.1 value increments. Relaxation Time values ranged from 0.1 to 0.7, with increments of 0.2. Coefficients of Desired Force and Social Force were varied from 1 to 30 in 5 value increments. After performing parameter calibration for each of the four real world datasets, we compared the resulting values and found that the best parameter values were quite similar.

C Model Implementation details

The model was optimized with the given loss function using the Adam optimizer with a learning rate of 5×10^{-5}. Regarding the transformer model, hidden dimensions were set to $nhid = 128$, and we used 4 layers and 8 attention heads in each layer. The ReLU activation function was used

*Equal Contribution

[1]https://github.com/sybrenstuvel/Python-RVO2

[2]https://github.com/yuxiang-gao/PySocialForce

Parameter	Description
Max Speed	The maximum speed of an agent
Max Neighbors	The maximum number of other agents that an agent takes into account in navigation
Neighbor Dist	The maximum distance to other agents that an agent takes into account in navigation
Radius	The radius of an agent
Time Horizon	The minimal amount of time for which an agent's velocities that are computed by ORCA are safe and collision-free from other agents
Time Horizon Obs	The minimal amount of time for which an agent's velocities that are computed by ORCA are safe and collision-free from static obstacles

Table 1: ORCA parameters

Parameter	Description
Max Speed	The maximum speed of an agent
Relaxation Time	The time it takes for an agent to adapt their velocity towards their desired velocity
Radius	The radius of an agent
Coefficient of desired Force	Importance of desired force
Coefficient of social Force	Importance of social force
Max Speed Multiplier	The maximum speed does not exceed the multiplier of initial speed

Table 2: SFM parameters

Parameter	Value
Max Speed	1.8m/s (preferred speed 1.3m/s)
Max Neighbors	30 neighbors
Neighbor Dist	5.0m
Radius	0.2m
Time Horizon	5.0s
Time Horizon Obs	1.0s

Table 3: ORCA parameter values chosen for data generation via grid search

Parameter	Value
Relaxation Time	0.5 s
Coefficient of Desired Force	1
Coefficient of Social Force	20
Max Speed	1.4 m/s
Maximum Speed Multiplier	1.1

Table 4: SFM parameter values chosen for data generation via grid search

throughout the model, and the model was trained in batches of $1,024$ samples. To prevent overfitting, early stopping was employed. For pretraining on simulation data, training was stopped after 400 epochs if the validation error failed to improve, while for fine-tuning on real world data, training was stopped after 50 epochs.

Regarding the CNN model, Lidar data are fed into 5 convolutional layers, each containing 32 or 64 filters with dimensions 5×2, sliding over the angles with a stride of 2.

Regarding the creation of lidar data, the scanner possesses an angular resolution of 0.5 degrees, resulting in 720 distance values per lidar scan. For Lidar representation, we included static obstacles in the form of polygons, while for Trajectory representation, we ignored obstacles as done in most of related work of simulation models and trajectory prediction.

D Real World Data

The features of the datasets are summarized in Table 5. Images of all datasets, along with their corresponding approximated obstacles, are presented in Figure 1. For ETH, the pedestrians avoid walking on the snow throughout the entire period. Since the snow appears to influence pedestrian trajectories, we consider it as obstacles in the scene. For Hotel, the trees and train station benches are denoted as obstacles. For Zara1 and Zara2, the storefront is in both datasets, but the car parked on the street is only in Zara1. This is because the car is driven away before the start of Zara2's video recording. We map the goal location to the last recorded location of each pedestrian.

The raw videos were manually converted into trajectory data and obstacles were not encoded in the data and that is probably why most of the papers on these datasets ignore obstacles.

Dataset	Video	# Samples	# Agents	NL Agents	Velocity Avg (m/sec)
ETH	9min	8259	360	56	1.3
Zara1	6min	4641	146	48	1.13
Zara2	7min	5686	203	55	1.2
Hotel	13min	6154	390	4	1.03

Table 5: Summary information of real world datasets and their properties. The columns from the left: The columns from the left are: video length in minutes, number of data samples, number of unique agents, number of nonlinear agents, and agent average velocity in meters per second

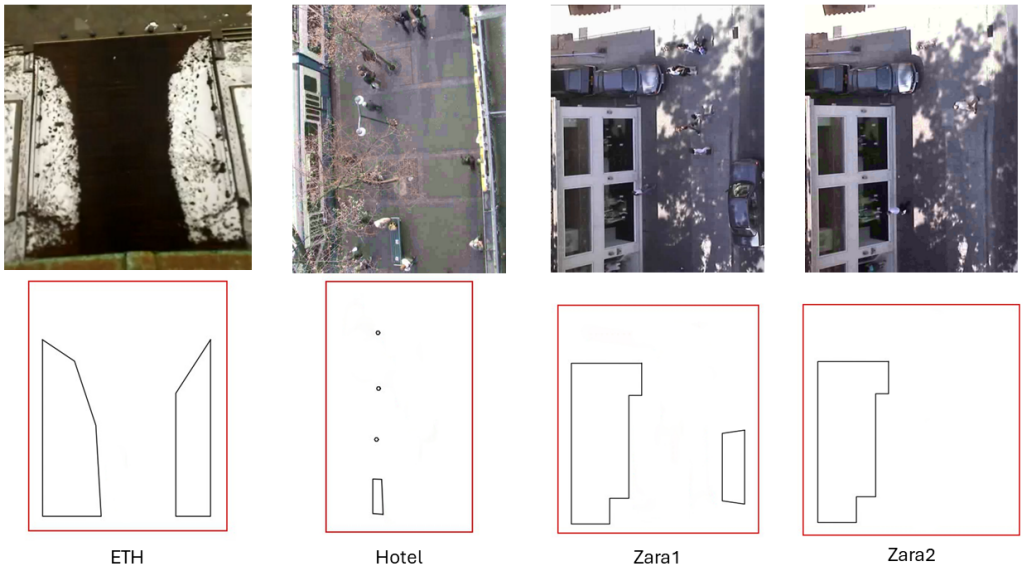


Figure 1: Datasets scene environment for simulator

E Additional Results

E.1 Fine-tuning Results

We compare our all the baselines on four datasets, ETH, Zara1, Zara2 and Hotel. Table 6 compares models under the **within** distribution while Table 7 compares the same models under the **outside** distribution. The time horizon was set to infinity, which means the rollout always goes from t_s to t_f. In all cases, fine-tuned models performed better than models trained from scratch on limited real world data.

The improvement from fine-tuning is relatively small for ADE(h), but is more pronounced for nADE(h). A low MDE(h) indicates that agents are able to reach their goal locations during simulations. While all baselines models have relatively low MDE(h) values, Fine-tune models have the lowest.

The Hard Collision metric reports how often collisions happen, while the Soft Collision metric reports the severity of collisions that occur. Note that these metrics only report collisions between multiple agents, and not between agents and obstacles. For these metrics, ORCA and SFM perform the best with the lowest Hard Collision values. This is because of the nature of these knowledge-based simulators, where ORCA guarantees no collisions and SFM has hyperparameters that can be tuned to make agents collision avoidant. While there are some collisions, Fine-tune models retain relatively low values for those metrics.

We found the above trends of these metrics to be present when using both the **within** and **outside** distribution. For the **outside** distribution, we see the benefits of using the Lidar data representation over the Trajectory representation.

Overall, fine-tuning on generated data from knowledge-based simulators improves performance. This generated data is incredibly valuable in the case of pedestrian trajectory prediction, where there is little real world data.

E.2 Lineplots for within and outside distribution

Line plots in Figures 2, 3, 4, 5, 6 illustrate the performance of different methodologies for each metric across varying values of h, highlighting the evolution of these metrics as h increases. For readability, only a subset of representative models is shown. As h grows, predicting the trajectory of an agent becomes increasingly challenging. This trend is evident as the ADE(h), nADE(h), and Soft Collision metrics generally increase across all methods with increasing h.

Fine-tuned models consistently outperform other methods across all values of h, with the performance gap widening notably in **within** distribution as h increases. In the ADE(h) and nADE(h) plots, we observe that for small values of h,

	Method	ADE(h)	nADE(h)	MDE(h)	hACS(%)	sACS(%)
Hotel	CVM	0.47	1.86	**0.05**	1.34	0.56
	ORCA Vis Obst	0.84	1.04	0.11	**0.00**	**0.00**
	ORCA Invis Obst	0.84	1.04	0.11	**0.00**	**0.00**
	SFM Invis Obst	0.62	0.74	0.08	**0.00**	**0.00**
	S-LSTM	1.32	0.92	0.08	1.47	0.39
	Scratch Transf	0.72	1.04	0.08	4.58	0.98
	Transf Fine ORCA	0.59	0.93	**0.05**	1.89	0.56
	Transf Fine SFM	**0.36**	**0.59**	**0.03**	0.98	0.14
	Scratch CNN	0.86	1.45	0.09	1.47	0.37
	CNN Fine ORCA	0.47	0.76	**0.03**	0.92	0.09
	CNN Fine SFM	0.43	1.05	**0.03**	**0.81**	**0.08**
Zara2	CVM	1.34	2.20	0.12	4.54	1.23
	ORCA Vis Obst	1.53	1.96	0.14	2.21	0.21
	ORCA Invis Obst	1.57	2.07	0.14	1.66	0.21
	SFM Invis Obst	1.61	2.06	0.10	**0.21**	**0.04**
	S-LSTM	1.41	1.38	0.11	4.84	1.27
	Scratch Transf	1.73	2.50	0.16	5.65	1.87
	Transf Fine ORCA	1.18	**1.25**	**0.04**	2.84	0.95
	Transf Fine SFM	**1.14**	1.47	**0.03**	2.50	0.37
	CNN Scratch	1.64	1.84	0.11	2.55	0.70
	CNN Fine ORCA	1.29	1.36	**0.03**	4.12	1.08
	CNN Fine SFM	**1.14**	1.56	**0.03**	3.06	0.63

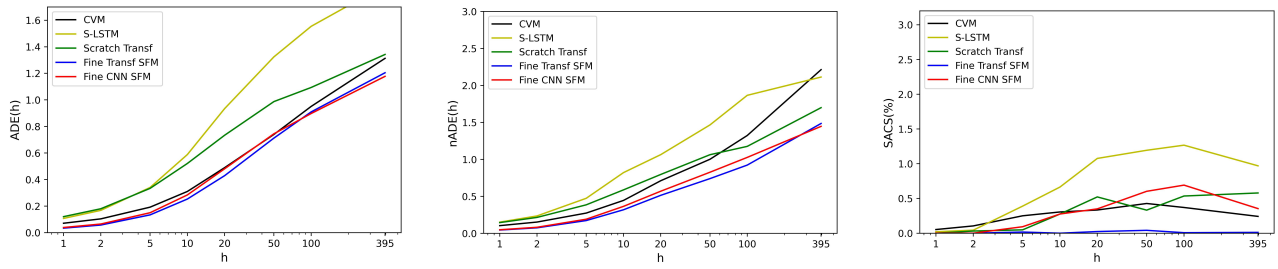
Table 6: Test metrics results for Hotel and Zara2 datasets with **within** distribution at $h = \infty$.

	Method	ADE(h)	nADE(h)	MDE(h)	hACS(%)	sACS(%)
Zara1	CVM	1.47	2.69	0.09	1.45	0.46
	ORCA Vis Obst	1.38	2.26	0.12	**0.04**	**0.00**
	ORCA Invis Obst	1.49	2.60	0.13	**0.04**	**0.00**
	SFM Invis Obst	1.32	1.92	0.10	**0.00**	**0.00**
	S-LSTM	1.23	1.97	**0.05**	2.27	0.76
	Transf Scratch	1.29	1.90	**0.04**	1.45	0.35
	Transf Fine ORCA	1.18	1.62	**0.04**	2.42	0.40
	Transf Fine SFM	1.19	1.50	**0.04**	1.05	0.10
	CNN Scratch	1.26	1.67	**0.04**	2.01	0.61
	CNN Fine ORCA	1.22	**1.36**	**0.03**	2.90	0.49
	CNN Fine SFM	**1.14**	**1.37**	**0.03**	**0.24**	**0.08**
Hotel	CVM	0.43	1.81	0.04	2.54	0.67
	ORCA Vis Obst	0.70	1.00	0.11	**0.03**	**0.00**
	ORCA Invis Obst	0.70	1.00	0.11	**0.03**	**0.00**
	SFM Invis Obst	0.55	0.70	0.07	**0.02**	**0.00**
	S-LSTM	0.50	0.63	0.04	2.91	0.58
	Transf Scratch	0.52	0.66	0.04	2.87	0.53
	Transf Fine ORCA	0.52	**0.42**	0.04	**0.73**	**0.09**
	Transf Fine SFM	**0.38**	**0.42**	**0.03**	1.08	0.16
	CNN Scratch	0.97	0.81	0.04	2.93	0.76
	CNN Fine ORCA	0.52	0.89	**0.03**	2.38	0.51
	CNN Fine SFM	0.62	0.80	**0.03**	2.43	0.55
Zara2	CVM	1.53	2.84	0.12	4.06	1.10
	ORCA Vis Obst	1.56	2.06	0.15	**1.58**	**0.16**
	ORCA Invis Obst	1.69	2.44	0.15	**1.30**	**0.16**
	SFM Invis Obst	1.77	2.32	0.12	**0.23**	**0.02**
	S-LSTM	1.53	1.74	0.10	5.79	1.53
	Transf Scratch	1.27	1.47	0.09	5.39	1.59
	Transf Fine ORCA	1.48	1.61	0.06	2.76	0.64
	Transf Fine SFM	1.17	**1.38**	**0.04**	2.67	**0.49**
	CNN Scratch	1.28	1.56	**0.04**	3.10	0.95
	CNN Fine ORCA	1.38	1.62	**0.04**	**2.63**	**0.52**
	CNN Fine SFM	**1.11**	**1.39**	**0.03**	**2.65**	**0.51**

Table 7: Test metrics results for Hotel and Zara2 datasets with **outside** distribution at $h = \infty$.

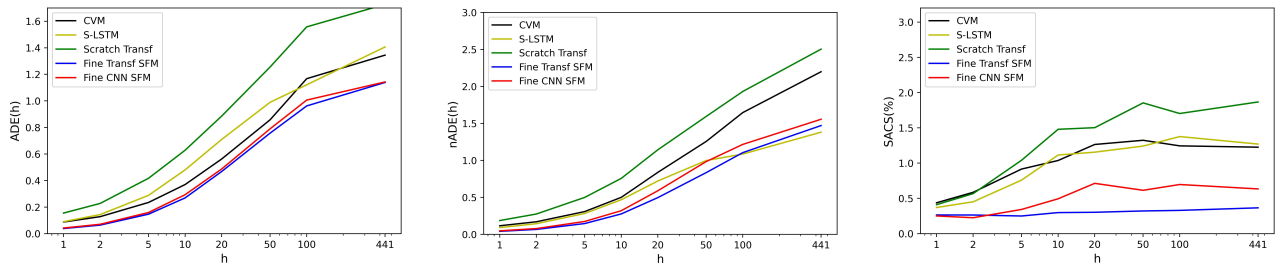
most methods perform similarly, but as h increases particularly after $h = 20$, the Scratch Transformer and CVM methods exhibit a much steeper deterioration compared to others. This rapid increase in error metrics for these methods indicates their declining performance in handling long-term trajectory predictions.

Furthermore, the Soft Collision metric indicates that as the horizon h grows, the risk of collisions increases, especially for models not fine-tuned on more complex data representations like Social LSTM and Scratch Transformer. Fine-tuned CNN and Transformer models on the SFM dataset manage to keep the Soft Collision metric relatively low, demonstrating their effectiveness in maintaining safe and realistic simulations even as prediction horizon increases.



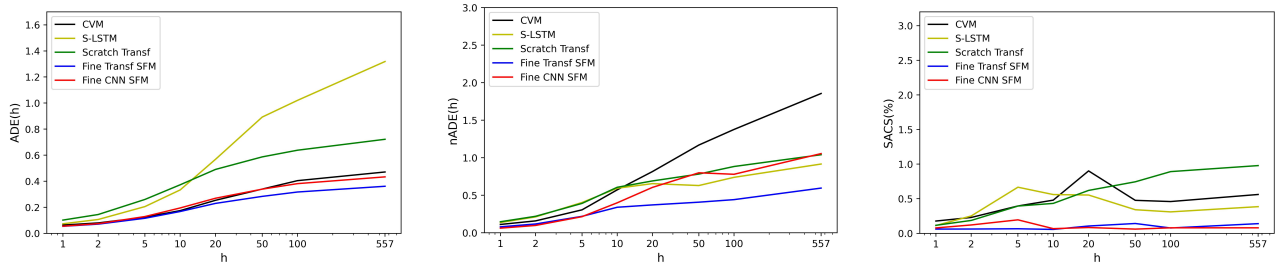
(a) ADE(h) (b) nADE(h) (c) Soft Collision

Figure 2: Test metrics for Zara1 dataset with varying values of h in **within** distribution.



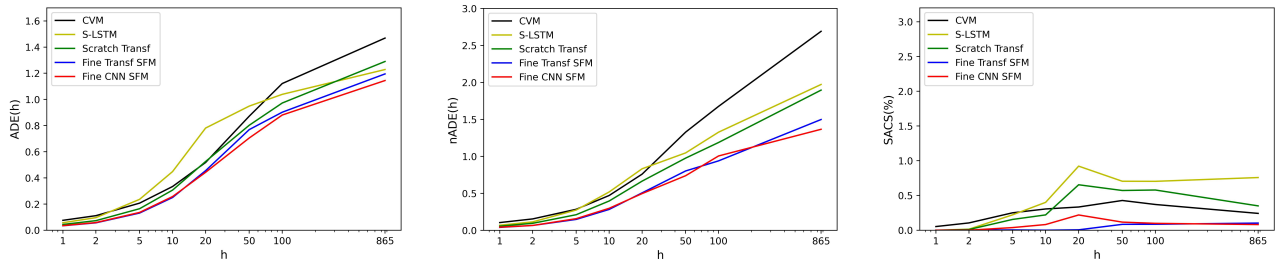
(a) ADE(h) (b) nADE(h) (c) Soft Collision

Figure 3: Test metrics for Zara2 dataset with varying values of h in **within** distribution.



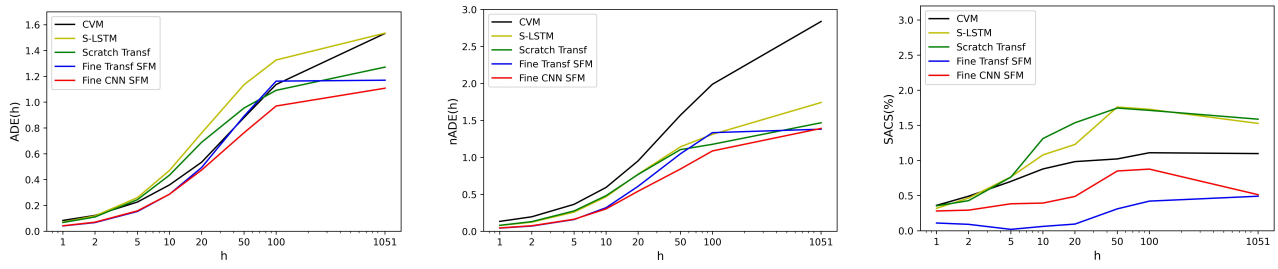
(a) ADE(h) (b) nADE(h) (c) Soft Collision

Figure 4: Test metrics for Hotel dataset with varying values of h in **within** distribution.



(a) ADE(h) (b) nADE(h) (c) Soft Collision

Figure 5: Test metrics for Zara1 dataset with varying values of h in **outside** distribution.



(a) ADE(h) (b) nADE(h) (c) Soft Collision

Figure 6: Test metrics for Zara2 dataset with varying values of h in **outside** distribution.