[Re] Contrastive Learning of Socially-aware Motion Representations

Roopsa Sen, Sidharth Sinha, Parv Maheshwari, Animesh Jha
Indian Institute of Technology, Kharagpur

Original Paper: major claims

Current datasets for trajectory prediction do not have negative scenarios (like collisions) for the models to train on. Therefore even though the models achieve high accuracy in terms of Final Displacement Error, the predicted trajectories have a significantly high collision rate.

The paper suggests the use of data augmentation to generate negative samples from a dataset and add a contrastive loss to incorporate that.

Original Paper: major claims

Robustness is quantified through:

1. Final displacement error (FDE): the Euclidean distance between the predicted output and the ground truth at the last time step.
2. Collision rate (COL): the percentage of test cases where the predicted trajectories of agents run into collisions

A lower FDE is preferred, however the current reproduction mainly aims to see the decrease in collision rate.

Aim of the reproducibility report

Our report verifies the following claims:

1. Addition of the Social-NCE loss in human trajectory forecasting models significantly decreases collision rate while maintaining similar final displacement error.

2. Addition of the Social-NCE loss in imitation learning models for robot navigation in crowded environments significantly decreases the collision rate.

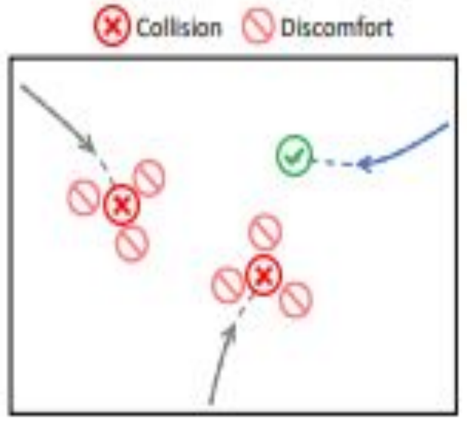
3. Addition of the Social-NCE loss in reinforcement learning models increases sample efficiency, and they obtain a collision-free policy quickly.

Social Contrastive Loss: Augmentation

Negative sample: The future state of all other agents are treated as a negative key for a particular agent

8 points are taken uniformly from a circle with radius of minimum distance of comfort(ρ) around the agent j (all agents other than i) as negative keys for agent i

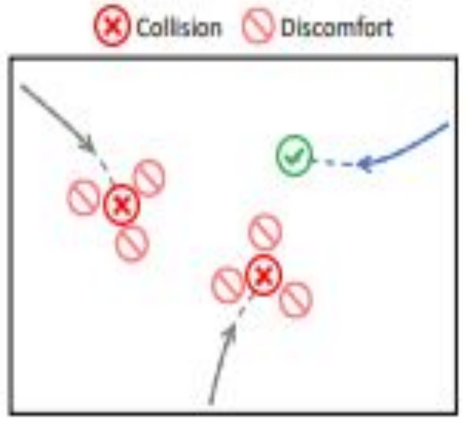
A normally distributed noise is also added to introduce randomness



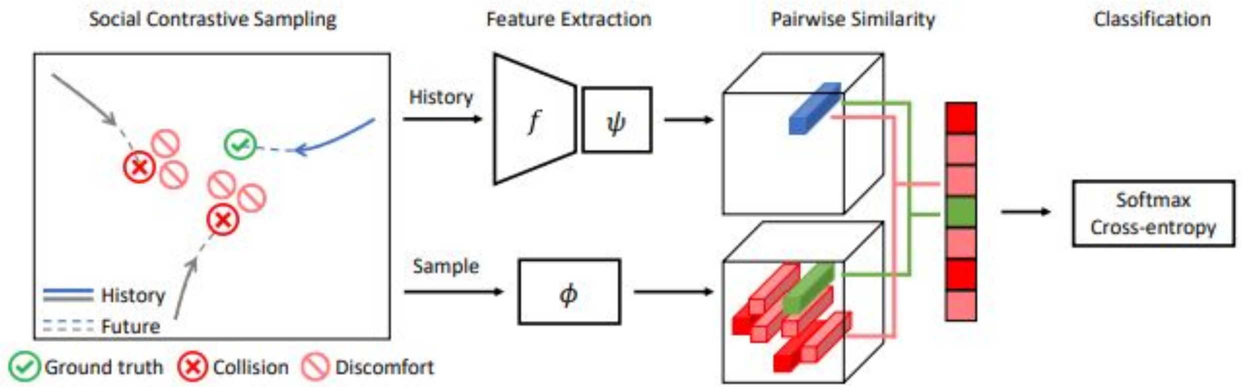
Social Contrastive Loss: Augmentation

Positive key: The future state of the agent is taken as a single positive key for a particular agent

A normally distributed noise is added



Social Contrastive Loss: Pipeline



f Gives vector encoding of agent i at time t given the state of all agents till time t

ψ Gives query vector from vector encoding of agent i given by f

ϕ Gives key vectors for each possible future augmented samples

Social Contrastive Loss: Loss Function

The InfoNCE contrastive loss is given by:

$$L_{NCE} = -log\frac{exp(sim(q,k^+)/\tau)}{\sum_{n=0}^{N} exp(sim(q,k_n)/\tau)}$$

Here sim(q,k) is the similarity function between the query and the key which in contrastive loss is the cosine similarity between the two vectors

Social Contrastive Loss: Loss Function

The SocialNCE addition to the contrastive loss is given by:

$$L_{Social-NCE} = -log\frac{exp((\psi({h^i}_t)\cdot\phi(s^{i,+}_{t+\delta t}, \delta t)/\tau)}{\sum_{\delta t\in\Lambda}\sum_{n=0}^{N} exp((\psi({h^i}_t)\cdot\phi(s^{i,n}_{t+\delta t}, \delta t)/\tau)}$$

The cosine similarity function is modified to the dot product of the 2 embedded vectors ψ(·) and ϕ(·)
The three encoders f (·), ψ(·) and ϕ(·) are jointly trained such that the query is encoded closer to the positive key and further from the negative keys.

Methodology

We have done the following work for our report:

1. Reproduced the findings of the paper based on the public repositories for the implementation of Social-NCE on Trajectron++, Social-STGCNN and Rainbow DQN.
2. Attempted ablations on Social-NCE hyper-parameters in the Social-STGCNN model to improve its performance.
3. Ported the codebase for Trajectron++, Social-STGCNN and the imitation learning model to PyTorch Lightning

Datasets

The Social-STGCNN and Trajectron++ models were run on a processed version of **ETH and UCY datasets**.

The original dataset is a collection of **5 video segments** of pedestrian trajectories from which the states of each agent per frame id had been stored and the dataset had been pre-divided into train, test and validation sets to maintain uniformity in accuracy comparison

The imitation and reinforcement learning models used pedestrian data from an open-source simulator based on OpenAI gym library The dataset consisted of **5000 simulated situations** in which the position of **5 random agents** are stored for each time step. A **validation split of 0.3** was taken

.

Code Implementation

The training code for Trajectron++ and Social-STGCNN was run on Kaggle with GPU (Tesla P100-PCIE-16GB) and CPU (13GB RAM + 2-core of Intel Xeon)

The code was also integrated with WandB to conduct further experiments. This was useful in logging data while carrying out sweeps during hyperparameter tuning.

Hyperparameter Choice

The Social-NCE loss includes three hyperparameters specific to it.

Details on the loss hyperparameters:

1. **Temperature(τ):** Part of the Social-NCE loss which controls the weight of the penalty and reward for negative and positive samples respectively.
2. **Sampling Horizon(δt):** The future time step till which the negative samples are considered for data augmentation
3. **Contrastive Weight(λ):** The weight between the main loss of the model and the Social-NCE Loss

Hyperparameter Choice

The data augmentation for generating positive and negative keys also has three hyperparameters.

Details on the augmentation hyperparameters:

1. **Minimum Separation:** Minimum admissible value of ρ in negative augmentation which is the minimum comfortable distance between two agents
2. **Maximum Separation:** Maximum admissible value of ρ in negative augmentation which is the maximum distance after which agents can pass each other with collision.
3. **Weight between maximum separation and noise:** The weight between the added normal noise and the position of the augmented sample.

Results: Social-STGCNN

A comparison of the FDE(Final Displacement Error) and COL(Collision Rate) for the addition of Social-NCE in Social-STGCNN model in original paper, reproduced and ported code.

Addition of Social-NCE to Social-STGCNN showed a **35.7%** decrease in collision rate on average

The Final Displacement Error(FDE) showed deviation of **less than 1%**

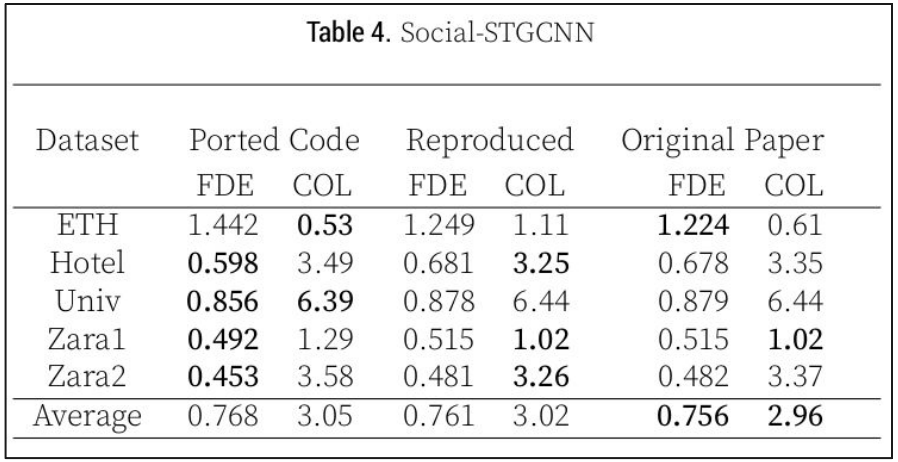


Table 4. Social-STGCNN

Dataset	Ported Code		Reproduced		Original Paper	
	FDE	COL	FDE	COL	FDE	COL
ETH	1.442	**0.53**	1.249	1.11	**1.224**	0.61
Hotel	**0.598**	3.49	0.681	**3.25**	0.678	3.35
Univ	**0.856**	**6.39**	0.878	6.44	0.879	6.44
Zara1	**0.492**	1.29	0.515	**1.02**	0.515	**1.02**
Zara2	**0.453**	3.58	0.481	**3.26**	0.482	3.37
Average	0.768	3.05	0.761	3.02	**0.756**	**2.96**

Results: Trajectron++

A comparison of the FDE(Final Displacement Error) and COL(Collision Rate) for the addition of Social-NCE in Trajectron++ model in original paper, reproduced and ported code.

Addition of Social-NCE to Trajectron++ showed a **34.8%** decrease in collision rate on average

The Final Displacement Error(FDE) showed deviation of **less than 1%**

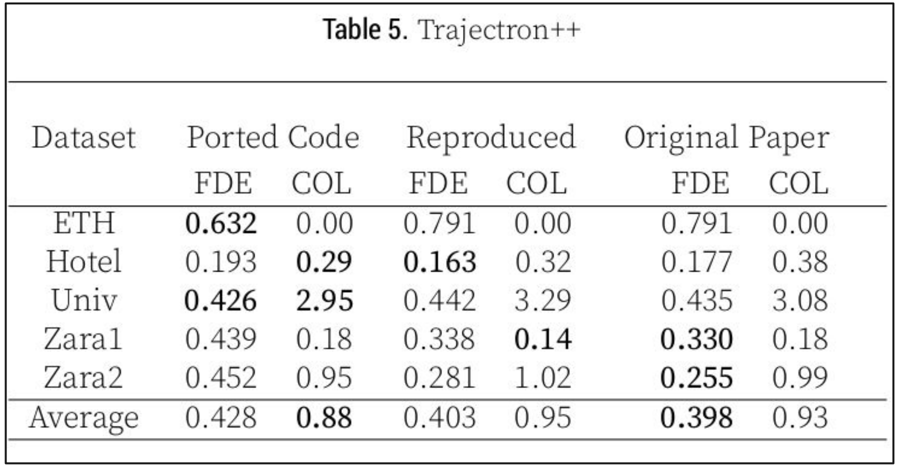


Table 5. Trajectron++

Dataset	Ported Code		Reproduced		Original Paper	
	FDE	COL	FDE	COL	FDE	COL
ETH	**0.632**	0.00	0.791	0.00	0.791	0.00
Hotel	0.193	**0.29**	**0.163**	0.32	0.177	0.38
Univ	**0.426**	**2.95**	0.442	3.29	0.435	3.08
Zara1	0.439	0.18	0.338	**0.14**	**0.330**	0.18
Zara2	0.452	0.95	0.281	1.02	**0.255**	0.99
Average	0.428	**0.88**	0.403	0.95	**0.398**	0.93

Results: Imitation Learning

A comparison of the collision rate and time taken for the robot to reach destination for the addition of Social-NCE in the imitation learning model in original paper, reproduced and ported code.

The collision rate decreased by **68.9%** on average in comparison to the original model without Social-NCE loss

The time taken shows little deviation

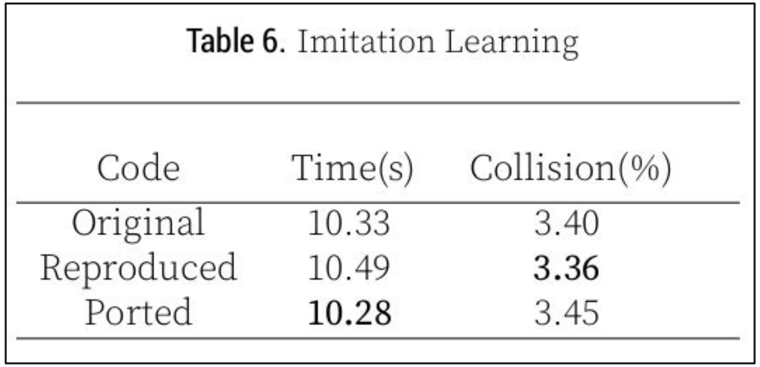


Table 6. Imitation Learning

Code	Time(s)	Collision(%)
Original	10.33	3.40
Reproduced	10.49	**3.36**
Ported	**10.28**	3.45

Results: Rainbow-DQN

A table of reward vs number of episodes trained for the implementation of Social-NCE on the Rainbow-DQN based reinforcement learning model.

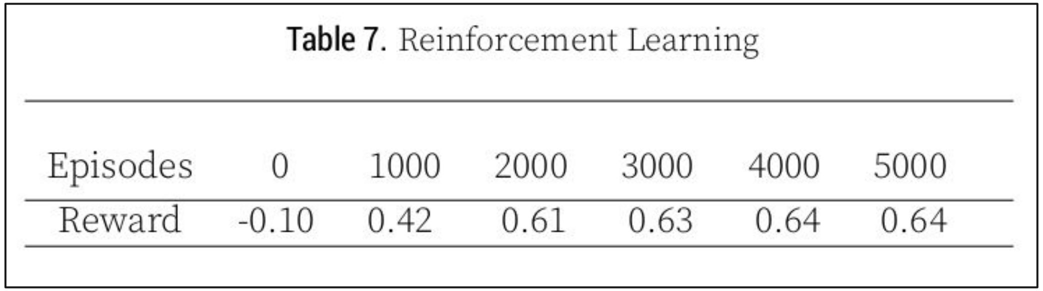


Table 7. Reinforcement Learning

Episodes	0	1000	2000	3000	4000	5000
Reward	-0.10	0.42	0.61	0.63	0.64	0.64

The Social-NCE addition to the model achieves a **reward of 0.6 in 2000 episodes** in comparison to 4000 episodes of the original model

Results: Hyperparameter Tuning

The FDE and collision rate after training the Social-STGCNN model for 400 epochs on the original (Original Parameters) and tuned hyperparameters (Tuned Parameters) are:

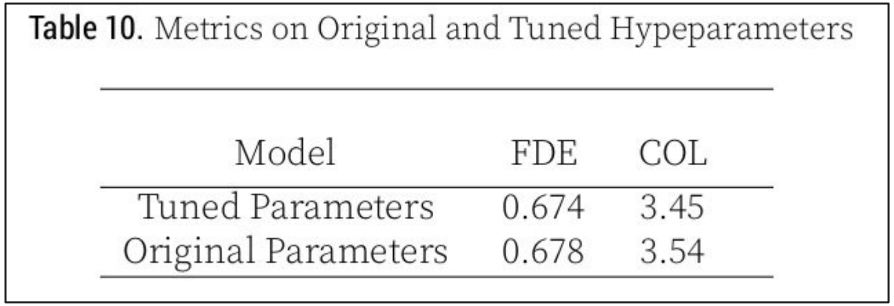


Table 10. Metrics on Original and Tuned Hypeparameters

Model	FDE	COL
Tuned Parameters	0.674	3.45
Original Parameters	0.678	3.54

The hyperparameter tuning conducted led to a decrease in collision rate by 1%

Conclusion and Future Work

We have reproduced the results of the paper to a maximum deviation of 2%

Due to lack of time we could not perform hyperparameter studies on the other models used in the paper and that can be done in the future

Further, the Social-NCE loss can be added into state of the art models on other benchmarks such as on the PGP model for the nuScences dataset

Thank You!