Contrastive Learning to Prevent Offroad Vehicle Trajectories

VITA - MT/ROB - Semester Project - 10ECTS

Filip Slezak (286557)

Supervised by Mohammadhossein Bahari

June 17, 2022

Contents

1 Introduction

This semester project follows the work on **vehicle trajectory prediction that works, but not everywhere**. Resulting from that research, where scenes of the **Argoverse** dataset are realistically attacked to challenge the underlying trajectory prediction network, the off road metric was not very pleasing. Indeed, trajectory predictions leaning towards off road areas is quite unlikely in reality and may lead towards catastrophic decision making in the pipeline that relies on that information.

This work attempts, using **Uber's LaneGCN** model to better the off road metric along the usual displacement error. The underlying idea to do so, is to modify the loss function to take into account some expert knowledge, in the form of contrastive samples, similarly to **SocialNCE**. Through a *cross entropy loss*, the model should tend towards *positive samples*, the ground truth, and get away from *negative samples*, close by off road points generated from the map.

2 Better Finetuning

The first step of this project was to verify the results presented in the paper and establishing a baseline for comparison. As shown in Table 1, attacking the scenes already reduces the off road rate on synthetic data, likely thanks to diversification of the training dataset at hand. After some parameter tuning we managed to reduce the number further, and we end up around 30% from the original value. The goal is now to beat that by injecting the contrastive loss mentioned before.

Learning Rate	Attack Power	ADE	FDE	RealOff1	RealOff2	SynthOff1	SynthOff2
3E-6	4000	1.351	2.981	0.13	0.79	21.90	65.00
3E-5	4000	1.372	3.020	0.13	0.83	21.57	64.50
3E-6	4011	1.351	2.981	0.14	0.84	17.18	56.00
3E-5	4011	1.367	3.016	0.14	0.94	12.42	46.00

Table 1: Baseline results following tuning of the learning rate.

3 Contrastive Learning

The first step is to choose an input to consider. Observing Uber's LaneGCN architecture in Figure 1, the choice originally made was to consider the output of the lane to actor attention layer for this was meant to represent the notion of keeping the vehicles in lanes. However, following inconclusive experimentation and code that seemed to work, the input choice shifted to the most likely trajectory output from the header.

A note on project flow: several weeks were spent on understanding the workings and interactions of different modules as well as the data at hand, especially its dimensions and encoding. In hindsight it shouldn't have been that much trouble but it is understandable. To mitigate this, documentation is provided wherever it felt necessary. For future projects, the use of print statements

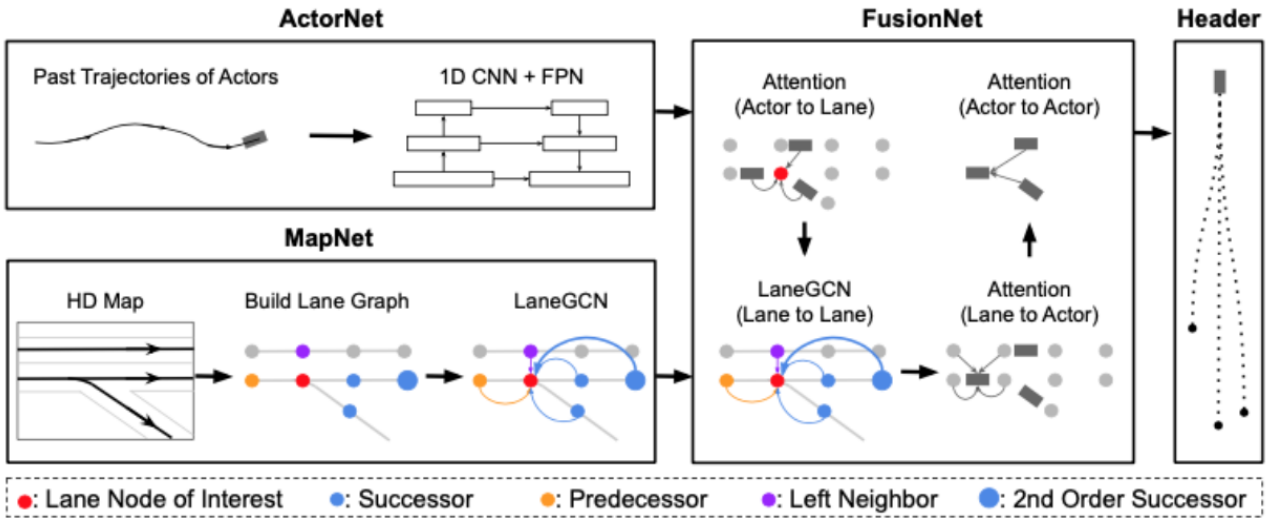

Figure 1: Uber's LaneGCN architecture.

is discouraged in favour of the python debugger. The actual coding parts were usually not the problem. Understanding the data structures is absolutely key.

Eventually, a working implementation outputs the visualisations in Figure 2. Observations were made that several agents had invalid or uninteresting data trajectories. A prefiltering step was therefore introduced. This filter allows the network to consider only trajectories of seemingly valid length (non stationary vehicles) and that veer off road event though their ground truth stays on the road. As it turns out, this is extremely selective but those are the most interesting trajectories.

Furthermore, two sampling strategies are proposed for both positive and negative samples.

- Positive ground truth: simply the ground truth trajectory
- Positive noisy ground truth: the ground truth trajectory with some added noise to account for inexact positioning requirement
- Negative scaling area: a discretized circle is scaled around each trajectory point until the desired number of off road samples is collected (scales indefinitely)
- Negative grid cloud: a grid point cloud is generated and its off road points nearest to the trajectory are collected (defaults to zero if area size is insufficient)

Note that some strange activity regarding these trajectories has been observed and is presented in the ReadMe associated to the code. However, our sampling strategies do generate points in the desired areas, which is where most of the work lies regarding the SocialNCE implementation. Therefore, the network now has the data required to compute the contrastive loss.

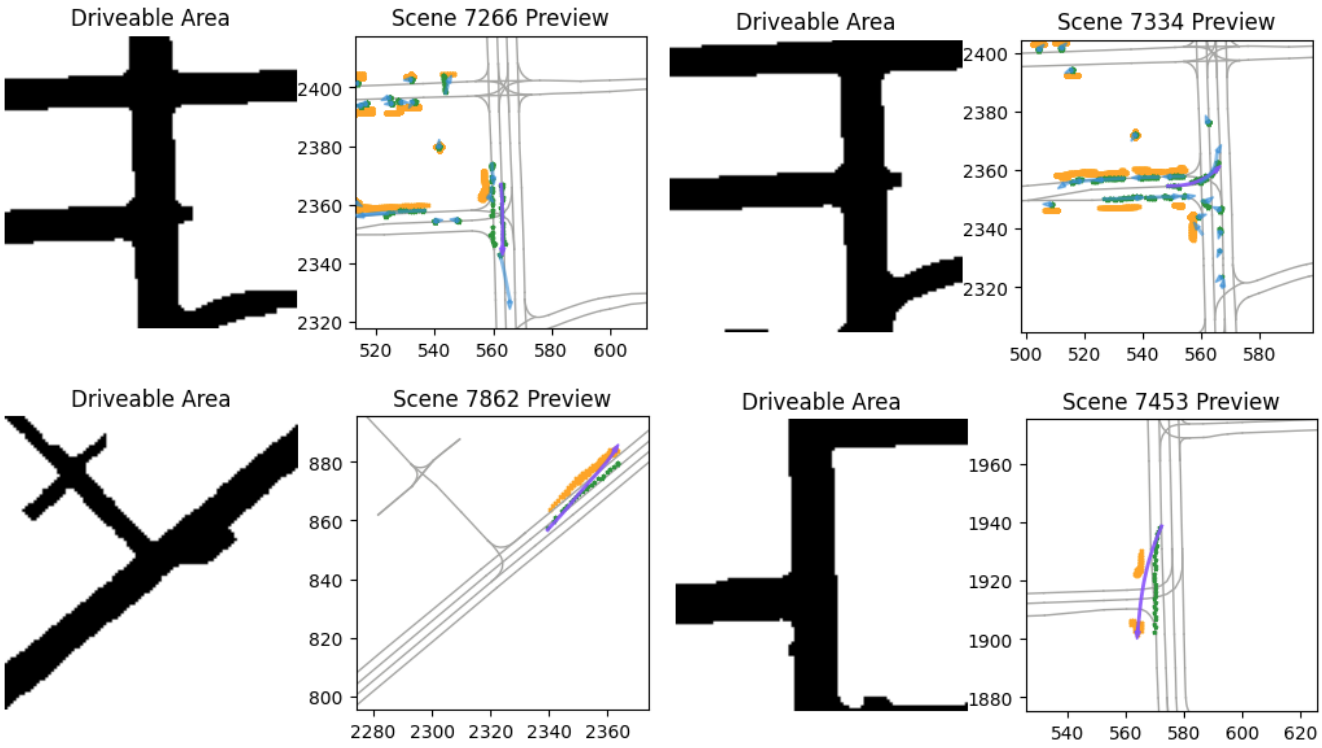

Figure 2: Visualisations of two non filtered (top) and two filtered (bottom) scenes.
Purple: Main Agent, Blue: Other Agents, Green: Positive Samples, Orange: Negative Samples

4 Experimentation

Several experiments were run at different stages of the project. The results spreadsheet is made available in the repository attached to the code.

Available choices of varying parameters include: number of epochs, learning rate, scene attack power, missing data completion, filtering strategy, sampling strategy and number of samples, contrastive parameter, and loss function weights. In the scope of this project, the time constraint did not allow us to run significant experiments using the latest version of the implementation, tuning these parameters could be enough work for a semester project in and of itself. Furthermore, the addition of the contrastive loss to the network increases the runtime a whopping 5-6x. There is likely room for optimization, probably regarding data management, transfers between cpu and gpu could be further explored.

However, preliminary results from what we do have seem to indicate that the learning rate is the most important parameter influencing our metrics. It does not directly imply that a contrastive loss is not helping. In fact, on some occasions, the network seemed to converge faster and even beat the baseline metrics by several points, see experiment 27. Again, the experimentation was clearly insufficient to reach any sort of explicit conclusion. This is just a statement to mention that learning rate is not to be neglected should this project be taken any further in the future and that the training time has been significantly increased as a result of adding this loss function.

5 Conclusion

Altogether, after a strenuous setup period, the project evolved a lot with different ides popping out here and there all the time. Following discussions, only the most promising ones were tried and kept. We believe all attempts were somehow justified and likely to reduce our off road metric but most of them failed. Given the performance of SocialNCE in collision avoidance for pedestrian trajectory forecasting, it probably has a greater potential than we could witness during this 6-months project and it might be worth exploring further for vehicles in their environment.

Personally, this was my first true research project and it has motivated me to pursue this path for my master's thesis (on an unrelated topic). I also learned several neat Python tricks and improved my understanding of how PyTorch works along the way. Especially, it helped me develop a coding workflow and with every runtime test taking close to 10min at times, I unlocked new levels of patience.

To conclude I'd like to thank Hossein for supervising this project, helping me understand what I was dealing with, personally digging through some code when it simply didn't work, as well as considering my ideas as worthy of trying out.