Analyzing the Robustness of PECNet

Aryan Garg
Indian Institute of Technology, Mandi
b19153@students.iitmandi.ac.in

Renu Rameshan
Indian Institute of Technology, Mandi
renum.r@gmail.com

Abstract

We perform a robustness analysis on PECNet [25], a pedestrian trajectory prediction system for autonomous vehicles, with various clustering analyzes, novel metrics for distribution analysis and classifications. We use synthetic data augmentation techniques ranging from Newtonian mechanics to Deep Reinforcement Learning based simulations to improve and test the system. We also decouple the system and perform various ablation studies alongside other experiments. We improved the state-of-the-art results on the Final Displacement Error(FDE) metric by 9.5%. We introduce a novel architectural change using implicit neural representations: Sinusoidal Representation Networks (SIRENs) [33] for higher precision results to validate our robustness hypothesis surrounding PECNet. We propose a novel, built upon PECNet, multi-modal system as well.

1. Introduction

Autonomous driving cars [35] is a booming concept and the race to go all the way from SAE level 0 to level 5 [36] is always on. Autonomous Vehicles aim to solve multiple issues like traffic congestions, improve driver comfort levels, and in the possible future make traveling on the roads safer and faster. However, there exist many social dilemmas and ethical questions about their incorporation in society [9]. Nonetheless, we don't aim to answer these questions through this study. To accomplish the aforementioned goals, we need extremely robust, adaptive, flexible and general systems that additionally, react quickly due to the high stakes' dynamic environment. Pedestrians add to that complexity due to their unknown destination and constantly evolving individual route planning and course correction based on the circumstances in their field of view. This rules out using simple Newtonian mechanics [6] to predict their motion. Autonomous cars are essentially social agents [8] and for that they need excellent anticipation systems to ensure people's safety without compromising on efficiency and performance. The problem is popularly known as *Human Motion / Pedestrian Trajectory Prediction* in the literature.

With the unceasing acceleration in the research, design and deployment of better artificially intelligent systems, we need a deeper understanding and sanity checks on all state-of-the-art models to ensure robustness and bridge the gap between deployment and research. Here robustness shouldn't be confused with adversarial robustness [39].

However, recently there have been multiple state-of-the-art benchmarks that grossly overfit by finding a pareto optimal solution(parameters) for a specific dataset. These models prove to be simply undeployable. The process of creating a model that benchmarks is given higher importance (for the sake of publications) than actual robust and potentially deployable system [29]. We propose a robustness/deployment fitness metric in section Discussion 5.

One such benchmark is: PECNet [25] that could not even generalize on Newtonian Trajectories and did even worse on noisy Newtonian Trajectories which are closer to the distribution of actual pedestrian trajectories.

2. Related Work

Variational Autoencoders(VAEs) introduced by Kingma et al [20] fall under the umbrella of generative models that are far from standard autoencoders [7, 22, 26, 37] as they first estimate the underlying distribution and not a fixed feature space using the encoder. Then the decoder samples, which is inherently stochastic, from this distribution and gives a generated output by reducing the Estimated Lower Bound (ELBO). The bottleneck has a stochastic node, from which the decoder queries, through which gradients are made to flow using the reparameterization trick. A great tutorial to understand vanilla VAEs was given by Carl Doersch in 2016 [11]. Few improvements and/or variations of the vanilla VAEs are as follows: β-VAE [17], Vector Quantized VaritionalAutoEncoder (VQ-VAE) by Oord *et al*. in 2017 [27] followed by VQ-VAE2 by Razavi *et al*. in 2019 [30], Temporal Difference Variational AutoEncoder (TD-VAE) by Gregor *et al*. in 2019 [15]. The inspiration to create the Reinforcement Learning based dataset was derived from the underlying ideas of TD-VAE.

Generally, the assumptions of a VAE are weak, and train-

Dataset	Trajectories(Pedestrian)	FPS
ETH	750	2.5
UCY	786	2.5
SDD	5232	30

Table 1. Most common datasets [4]. Other popular ones include: CITR, Town Center, KITTI, Waymo and Forking Garden Paths (synthetic) [24]

Model	Year	ADE	FDE	K
DESIRE	2017	19.25	34.05	20
Social GAN	2018	27.23	41.44	20
Sophie	2018	16.27	29.38	20
CGNS	2019	15.6	28.2	20
CF-VAE	2019	12.60	22.30	20
P2TIRL	2020	12.58	22.07	20
PECNet	2020	9.96	15.88	20
Y-Net	2021	7.85	11.85	20
V^2-Net	2021	7.12	11.39	20
NSP-SFM	2022	6.52	10.61	20

Table 2. ADE - Average Displacement Error. FDE - Final Displacement Error. K - number of samples allowed for multi-modal prediction

ing is fast. They are able to generalize well unlike generative adversarial networks [13] and hence have become quite popular within three years of their birth. Even though VAEs have been extremely successful in unsupervised learning applications such as generating faces [31], physical models [21], house numbers [14], segmentation [34] and forecasting from static images [38], (which is the primary application discussed here) they commonly suffer from the problem of vanishing Kullback-Leibler(KL) divergence. Cyclic annealing schedule as introduced by Fu *et al*. [12] is useful to mitigate this issue and we implement the same for certain experiments.

Moving forward to the 'human future prediction task' we take note of the most common datasets. See table 1

State-of-the-art benchmarks on the Stanford Drone Dataset(SDD) are briefly mentioned in table 2.

3. Methodology

A clearly outlined plan for PECNet was initially proposed. It incorporated analyzing PECNet, Stanford Drone Dataset(SDD), ETH [28] & UCY [23] datasets and all the literature around the work, in the first phase. Then, dependent on the analysis, the second phase would potentially include improving PECNet if deemed fit.

3.1. Clustering Analysis

We carried out an analysis of the Stanford Drone Dataset [32] first, by visualizing and clustering the trajectories, using various norms (see figures 1, 2, 3, 4). It is evident that most trajectories are not spatially spread out and the dataset possesses rotational equivariance.

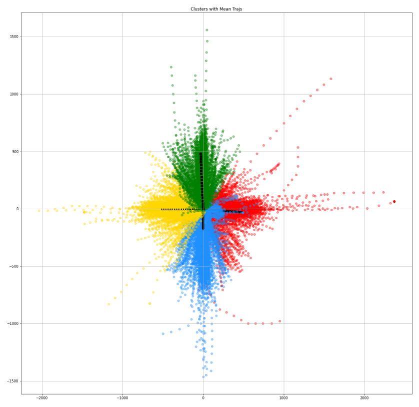


Figure 1. Full training dataset clustered based on the Frobenius norm, with black trajectories representing the cluster.

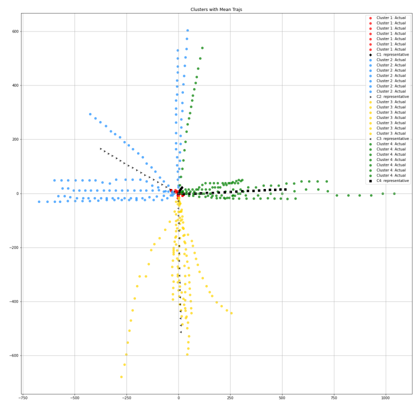


Figure 2. Clustering based on L1-norm. Black trajectories are representatives of the cluster.

3.2. Non-linearity analysis: Abruptness-Score

Then we clustered trajectories based on the bounding boxes to get an idea of the maximal displacement and turn in each. Also, came up with a novel and simple metric: *Abruptness Score* to measure the turns and variability or

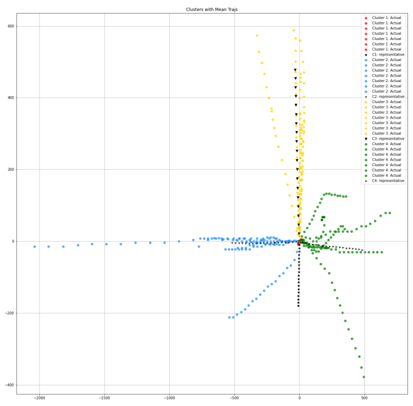


Figure 3. Clustering based on L2-norm. Black trajectories are representatives of the cluster.

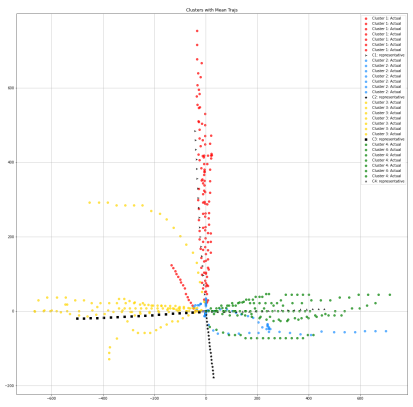


Figure 4. Clustering based on L-inf norm. Black trajectories are representatives of the cluster.

non-linearity in each trajectory. An areal-scaled and and unscaled version of the metric is used for analysis and outlier detection. The intuition and mathematical formulation are as follows:

In the example figure 5, the trajectories $\zeta_1 = \{A, B, C1\}$ and $\zeta_2 = \{A, B, C2\}$ are shown. The dotted blue line is the normal to the red danger zone. Points that fall under this danger zone will form an obtuse turn trajectory like ζ_2. Naturally, we want the score to assign a larger value to ζ_2 than ζ_1 since the turn is huge and the trajectory (more) abruptly changes direction.

Mathematically,

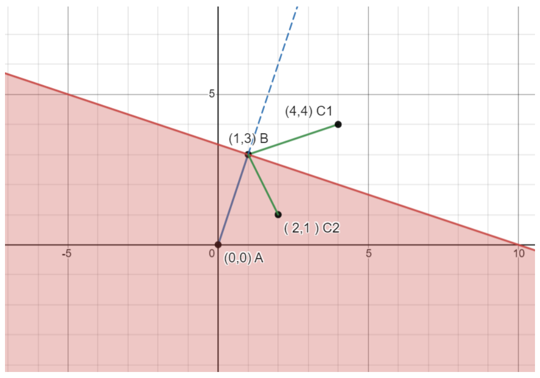


Figure 5. Defining Turns

Statistic	Value
Maximum	494866.374
Minimum	0.0
Mean	3430.665
Std. Deviation	11987.34

Table 3. SDD: AbScore Non-linearity analysis

$$AbScore = \left\lceil \frac{180.\theta}{10\pi} \right\rceil |\vec{a} \times \vec{b}| \tag{1}$$

where

$$\vec{a} = \vec{AB}, \vec{b} = \vec{BC} \tag{2}$$

$$\theta = |\arcsin \frac{\vec{a} \times \vec{b}}{|\vec{a}||\vec{b}|}| \tag{3}$$

If θ is obtuse, we add $pi/2$ to θ before sending it to equation 1.

For scaling we simply divide the abruptness score by the area of the tightest-bounding-box of the trajectory or divide by $(max(\zeta_x) - min(\zeta_x)) * (max(\zeta_y) - min(\zeta_y)$. For perfectly linear trajectories, we use the length of the trajectory for scaling.

Why do we need such areal-scaling?

To compare trajectories that span different sizes or regions in terms of their non-linearity.

Based on this metric we analyze SDD and report the findings in table 3 and figure 6.

3.3. Unique-points & Qualitative Analysis

We perform a classification of the training dataset based on the number of unique points in each trajectory. See table 4

Then we manually identify 7 qualitative classes for each trajectory as follows:

- Type 1: Stationary pedestrians

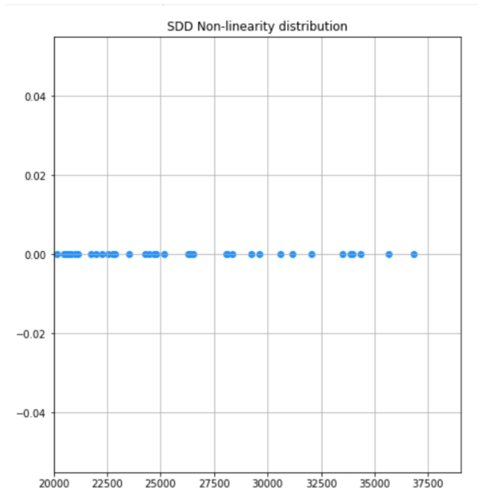


Figure 6. SDD's Non Linearity Distribution Tail. (Thresholded to remove outliers)

- Type 2: 3-8 unique points in trajectory bounded in a 5x5 box
- Type 2F: F means category Flying: A straight line trajectory will be shifted into the line starting from a new point if the perspective of the viewer (here: drone) changes. We call this scenario the flying category trajectory. The drone usually translates along an axis here.
- Type 3: 3-9 unique points loosely bounded in a 100x100 box
- Type 3F: (Flying) Same as 2F
- Type 4: Start and Goal points are within a 5x5 box
- Type 5: Flying randomly. This is different from 2F and 3F in the sense that the drone translated haphazardly here.
- Type 6: Backtracker: The pedestrian re-traces his steps after a while. Usually after 6-7 steps.
- Type 7: Perfectly Linear to Moderately linear trajectories that could be modelled by simple Newtonian mechanics.

These two classifications were done to emulate the statistical properties of the training dataset for augmentation purposes.

These two analyses gave us a clear idea about the trajectories that the model was learning on.

Unique Points	Trajectories	% dataset
1	145	5.13%
2	62	2.19 %
3	71	2.51 %
4	69	2.44%
5	57	2.01%
6	41	1.45%
7	51	1.80%
8	28	0.99%
9	26	0.92%
10	24	0.85%
11	22	0.78%
12	25	0.88%
13	17	0.60%
14	24	0.85%
15	22	0.78%
16	22	0.78%
17	39	1.38%
18	30	1.06%
19	76	2.69%
20	1978	69.92%

Table 4. Trajectories' unique points. 145 trajectories had 1 unique point, i.e, the goal and starting point as the same with all other points being sampled there itself: A stationary pedestrian.

Learning Rate	Best ADE	Best FDE
0.001	11.01	15.62
0.0005	12.52	15.68
0.0003	**10.47**	**15.60**
0.0002	10.65	15.78
0.0001	10.65	15.78

Table 5. Sanity Run on different learning rates. Figure 8. Bold - benchmark reproduction.

3.4. Benchmark Validation & Sanity Runs

Moving onto the system, we successfully reproduce the benchmark-S (without social pooling layer) metricS: Average Displacement Error (ADE) and Final Displacement Error (FDE) [2, 3]. This was done as a sanity check for our experiments and tests. Figure: 7

We also change the learning rates for another sanity run. Figure 8

3.5. Improvements

After this analysis, we moved on to the second phase of the plan: improvement. Naturally, the first idea for us was data augmentation due to the intensive clustering and classification of trajectories already performed. We created

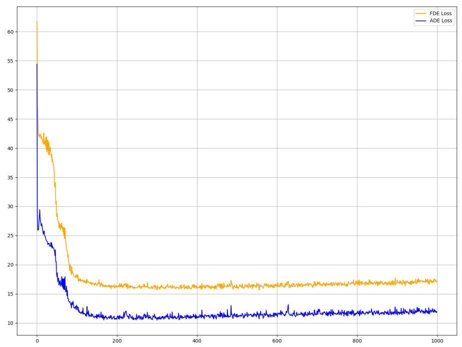


Figure 7. ADE & FDE Loss vs Epochs. Reproduction of the benchmark. Adam Optimizer, learning rate = 0.0003, epochs = 1000, $K = 20$. Best FDE : 15.60. ADE : 10.47. Model converges at 556^{th} epoch and the results are as quoted in [25].

five different synthetic datasets to augment.

1. **Purely Newtonian Trajectories:** Consists of two types of accelerations: static and variable but within bounds. [1]

2. **Noisy Newtonian Trajectories** The synthetic dataset created in the former was re-created with noise [1].

3. **Trajectories that followed circuits, loops, spirals, and other geometric curves with both variable and fixed sampling of the points [1].**

4. **Interaction modeling of trajectories using a novel Hidden Markov Model(HMM):** HMMs were considered for the interaction modeling due to their probabilistic nature and success in the tasks which depict structure in time [10] [1]. See figure 9 for an overview of the HMM model with states and action spaces and figure 10 for the samples generated.

5. **Reinforcement Learning based** agents that interacted with each other based on two inherent goals that model humans: Reaching the goal fast and avoiding collisions with fellow pedestrians. Agents were given control of their acceleration instead of their velocity to get smoother, human-like trajectories. Apart from acceleration, stopping for another crossing pedestrian, or being considerate which was observed in many actual trajectories, was implicitly decided by the agent's sociability, fitness, and patience attributes. These attributes remain fixed over a simulation and are randomly initialized for diversity. We observe that agents initially learn the greedy approach to reach their goals as fast as possible leading to multiple collisions which are then heavily penalized by the learning objective. This in turn leads to agents choosing sub-optimal paths that avoid maximal collisions. We add a detection mechanism to the agents so they can observe other agents in the scene too. This leads into super-natural trajectories where agents slow down or speed up when in the proximity of other agents who might potentially collide [1].

[1]Code of clustering analysis and the synthetic datasets' creation can be found at: Synthetic-Dataset-Creation-notebook

Mathematically, the reward function is formulated as follows:

$$R_t = \frac{AF^t\,(n_{ICS}+1)^{(AS+AP)}}{t^2\,(1+\|G-x_t\|)} \qquad (4)$$

$$where\ AF,\ AP,\ and\ AS\ \epsilon\ [0,1] \qquad (5)$$

where,
n_{ICS} := Number of times agent entered impending collision state but didn't enter collision state,

AS := Agent's sociability (an inherent and unchanging quality of the agent),

AP := Agent's Patience (an inherent and unchanging quality of the agent)

AF := Agent's Fitness. Usually a discount factor in standard RL reward function formulation, but here it signifies how quickly the agent gets tired. If an agent it extremely unfit, the more it walks, the lesser rewards it gets as it gets fatigued and naturally prefers walking less.

t := Time elapsed

G := Goal or end-point

x_t := Current position

$\|.\|$:= $L_2\,norm$

R_t := Reward received at time t

and loss function is the one used in standard deep-policy gradients methods:

$$J(\phi) = -\sum_{t=1}^{t=N} log(P_\phi(a_t|s_t))\,R_t \qquad (6)$$

where,
$P(.)$ is parameterized by ϕ modelled by a function ap-

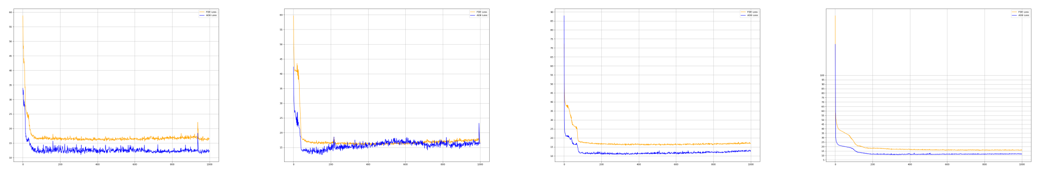


Figure 8. ADE & FDE Loss vs Epochs plots. Sanity-runs. See table 5. 1000 epochs.

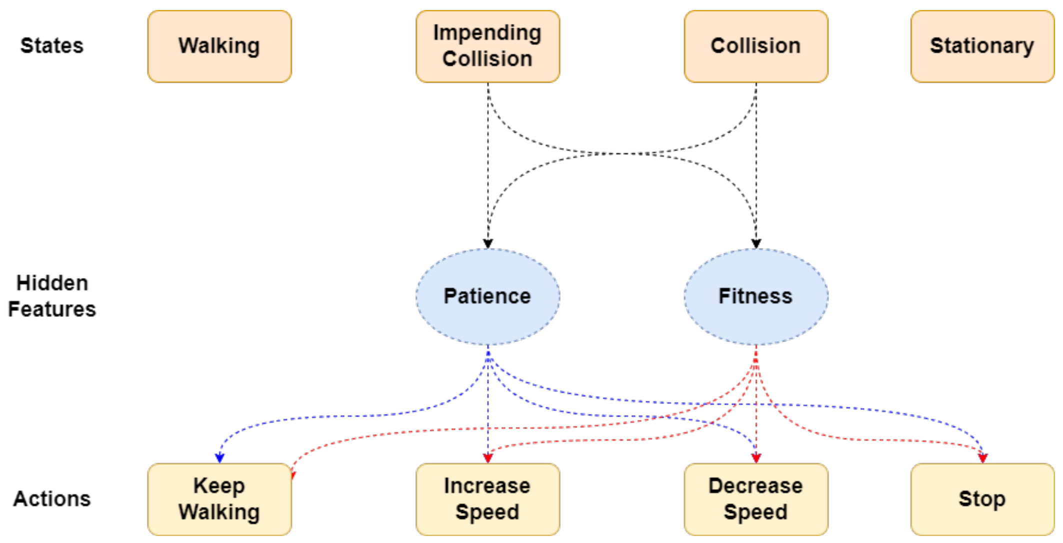


Figure 9. Interaction modeling graph of the hidden markov model (HMM) to emulate multiple human trajectories as produced in any real static-drone dataset.

proximator(DNN) that outputs $\mu\, and\, \sigma^2$ vectors corresponding to actions and state at time 't'.

Some samples produced using this approach are produced in figure 11

An overview of the generation process is in figure 12

6. **Synthetic SDD(syn-SDD)** is the re-creation of the SDD based on the qualitative classification in the same proportions [1].

We augment SDD with each category of synthetic dataset in different proportions, and note significant worsening in performance on both ADE and FDE. However, the FDE improved at the cost of ADE, when SDD was augmented with specific proportions of syn-SDD.

Assuming that the dataset doesn't represent the trajectories accurately enough and that the gradient terrain doesn't have a sharp creek which Adam [18](which is generally considered very sensitive) finds early on due to the hyper-parameter fine tuning methods, (which are not discussed in the paper: PECNet), we moved towards deep-learning based improvements. The improvements that we proposed, implemented and tested were as follows:

1. Hyper-parameter tuning (it was soon discovered that the model was using a pareto optimal solution).

2. Cyclical annealing schedule or more colloquially, β-annealing for stable training was implemented. KL vanishing problem was observed on some augmented datasets.

3. Replacing the standard multilayer perceptron layers(MLPs) with Sinusoidal Representational Networks (SIRENs) [33]

We also speculated using a Partially Observable Markov Decision Process for the same task [5]. More on this in Discussion section: 5

4. Experiments

With the objectives of testing the robustness of PECNet and understanding how the model behaves under new and challenging situations, which a real system would undergo on a daily basis, we perform experiments where we generate synthetic datasets for augmentation and testing, replace standard Multi-Layer-Perceptrons(MLPs) with SIREN MLPs. We finally carry out an ablation study.

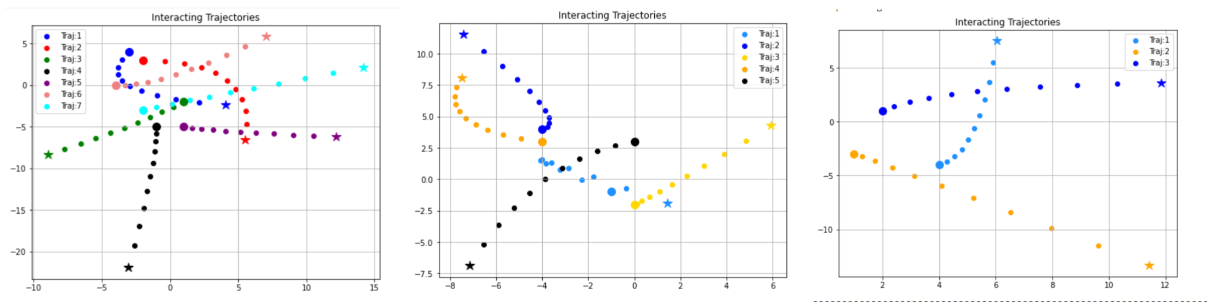


Figure 10. Interaction modeling samples generated.

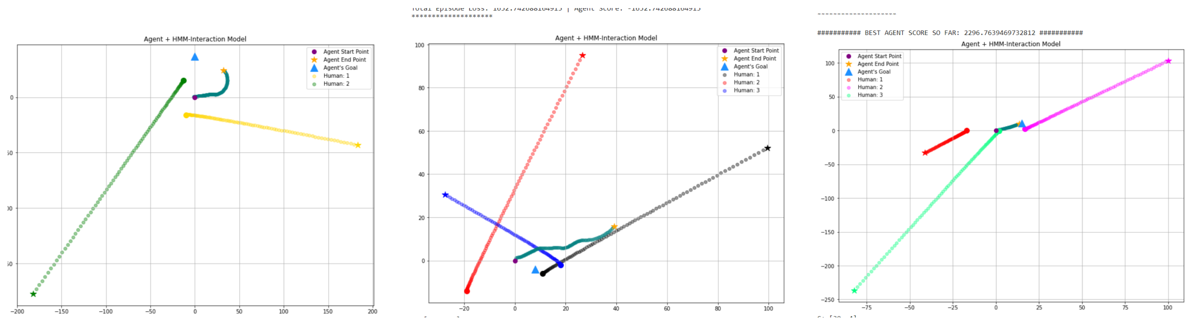


Figure 11. Agent inserted and trained in the HMM interaction model. Agent's trajectory is turquoise. Evolution of the samples produced(L-to-R): First, Initial Stages of Training: Agent learns to turn. Second depicts a complicated scene where the agent learns to avoid multiple collisions.Third depicts a scene where the agent successfully avoids a collision and reaches it's goal for the first time.

4.1. Synthetic Datasets

Based on the quantitative (table: 4) and qualitative analysis(7 representative classes) under section Methodology: 3, we augment the training dataset while keeping the statistical properties of the training dataset intact. Augmentations vary from 1% to 18%. We report the results in table 6 and observe that even small quantities of noise in the original dataset, due to augmentation, make the system worthless for deployment. PECNet heavily overfits on the SDD dataset and it is highly probable that Adam [19] finds a deep crevice in the gradient surface. An evolutionary optimization strategy like Covariance Matrix Adaptation (CMA-ES) [16] would highlight the shortcomings of the robustness of PECNet.

We create a pure synthetic dataset and perform three experiments:

1. Testing on ours where the test-set is generated as per the qualitative statistics of the testing set.
2. Purely synthetic dataset training and testing on original test-set.
3. Augmented training-set and testing on ours.

% Augmented	Total Trajectories	ADE	FDE
1%	18328	64.34	19.18
3%	19048	53.22	15.63
5%	19766	45.17	15.72
6%	20126	51.73	**15.43**
8%	20844	46.37	15.75
10%	21564	51.51	15.54
13%	22642	**40.76**	15.90
15%	23360	51.40	18.36
18%	24438	56.56	15.90

Table 6. Standard learning rate: $3e-4$, 1000 epochs and no social pooling.Note that the training dataset has 18k trajectories due to simple augmentations(rotations & translations) introduced in the original paper [25].

4.2. SIREN Improvement

Human Trajectories are inherently physical and natural entities with spatial and temporal derivatives. A trajectory, therefore, can be treated as a physical signal as well. The paradigm of implicit neural representation of continuous and derivable physical signals has been on the rise in the recent times. One such weight-initialization and periodic activation function, for finer detail representation in natural signals, is a method introduced by Sitzmann *et al.* [33] with impressive results on a wide-range of tasks and a thorough explanation of the mathematical background in the appendix.

We implement the idea in [33] and replace all standard MLPs of PECNet with SIREN MLPs. We expected an improvement or atleast the constancy of the benchmark results. However, we observed an adverse effect on both metrics which reinforced our belief in the hypothesis that PECNet abysmally overfits on SDD.

We present the results of our novel architectural contribution for 'human trajectory prediction task' in table 7.

Learning Rate	ADE	FDE	Best FDE epoch
0.001	>50	15.91	724
0.0005	>50	15.83	**345**
0.0003	>50	15.77	480
0.0002	>50	**15.66**	445
0.0001	>50	15.80	588

Table 7. SIREN-PECNet Results. No decoupling. 1000 epochs. For Loss vs Epoch plots, see figure 13

4.3. Ablation Studies

We performed two ablation studies. First, by decoupling the ADE and the FDE metrics. See table 8 and figure 14.

Learning Rate	ADE	FDE	Best FDE epoch
0.001	>50	15.68	457
0.0005	>50	15.76	301
0.0003	>50	15.9	541
0.0002	>50	15.65	420
0.0001	>50	15.92	391

Table 8. Decoupled system (ADE and FDE). No social pooling. See log loss plots (figure: 14)

Second, without the standardization parameter on the input trajectories. We note an adverse effect on ADE and a significant state-of-the-art improvement on FDE, by removing standardization. See table 9 and figure 15.

We notice a striking absurdity in the codebase where the standardization parameter($= 1.86$) is also dividing the loss. See figure 16. This is a grave mistake since a Variational Autoencoder using ReLU [1] as an activation funciton, is by no means a linear predictive machine.

5. Discussion

We propose three further changes that could help in further testing PECNet: Employing CMA-ES instead of Adam

Learning Rate	ADE	FDE	Best FDE epoch
0.001	**22.20**	9.32	915
0.0005	29.91	9.05	834
0.0003	25.92	9.37	998
0.0002	26.75	**9.04**	908
0.0001	25.57	9.05	**235**

Table 9. State-of-the-art FDE. No Standardization. No decoupling. No social pooling. See log-loss plots (figure: 15)

to skip deep crevices and explore the gradient surface more naturally. The second addition would be implementing a cyclic annealing scheduling to stabilize the Variational AutoEncoder training. The third would be to add another generative model in conjunction to produce multi-modal outputs, see Figure and also introduce a confidence metric for better predictions and larger controllability.

We also lay emphasis on the widening bridge between academic research and real-world applications of the same based on over-fitting of the datasets to set benchmarks for publications. For the same reasons we call for a robustness and deployment fitness score based on thorough code-reviewing and testing under simulations that would be encountered in a real world scenario.

A deep learning based idea, for the same, is to let all models(academic and industrial) exist as test-sets for a generative model, let's say robGAN, trained on deployed systems. When robGAN sees a model, it's discriminator should be able to tell whether the model is fit for deployment.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018. 8

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 4

[3] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-aware large-scale crowd forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 4

[4] Javad Amirian, Bingqing Zhang, Francisco Valente Castro, Juan Jose Baldelomar, Jean-Bernard Hayet, and Julien Pettre. Opentraj: Assessing prediction complexity in human trajectories datasets. In *Asian Conference on Computer Vision (ACCV)*, number CONF. Springer, 2020. 2

[5] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *2015 ieee international conference on robotics and automation (icra)*, pages 454–460. IEEE, 2015. 6

[6] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009. Reinforcement learning and higher cognition. 1

[7] Yoshua Bengio, Éric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop, 2013. 1

[8] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Learning motion patterns of persons for mobile service robots. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pages 3601–3606, 2002. 1

[9] Jean-François Bonnefon, Azim Shariff, and Iyad Rahwan. The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576, 2016. 1

[10] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, 1997. 5

[11] Carl Doersch. Tutorial on variational autoencoders, 2016. 1

[12] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing, 2019. 2

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[14] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation, 2015. 2

[15] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder, 2018. 1

[16] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2016. 7

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. 1

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 6

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 7

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. 1

[21] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 2

[22] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. 1

[23] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26, 2007. 2

[24] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction, 2019. 2

[25] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. *CoRR*, abs/2004.02025, 2020. 1, 5, 7

[26] Field D. Olshausen, B. Emergence of simple-cell receptive field properties by learning a sparse code for natural images, 1996. 1

[27] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. 1

[28] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009. 2

[29] Heimann R. Doing ai: A business-centric examination of ai culture, goals, and values, 2021. 1

[30] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 1

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014. 2

[32] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Stanford drone dataset. 2

[33] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. 1, 6, 8

[34] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 2

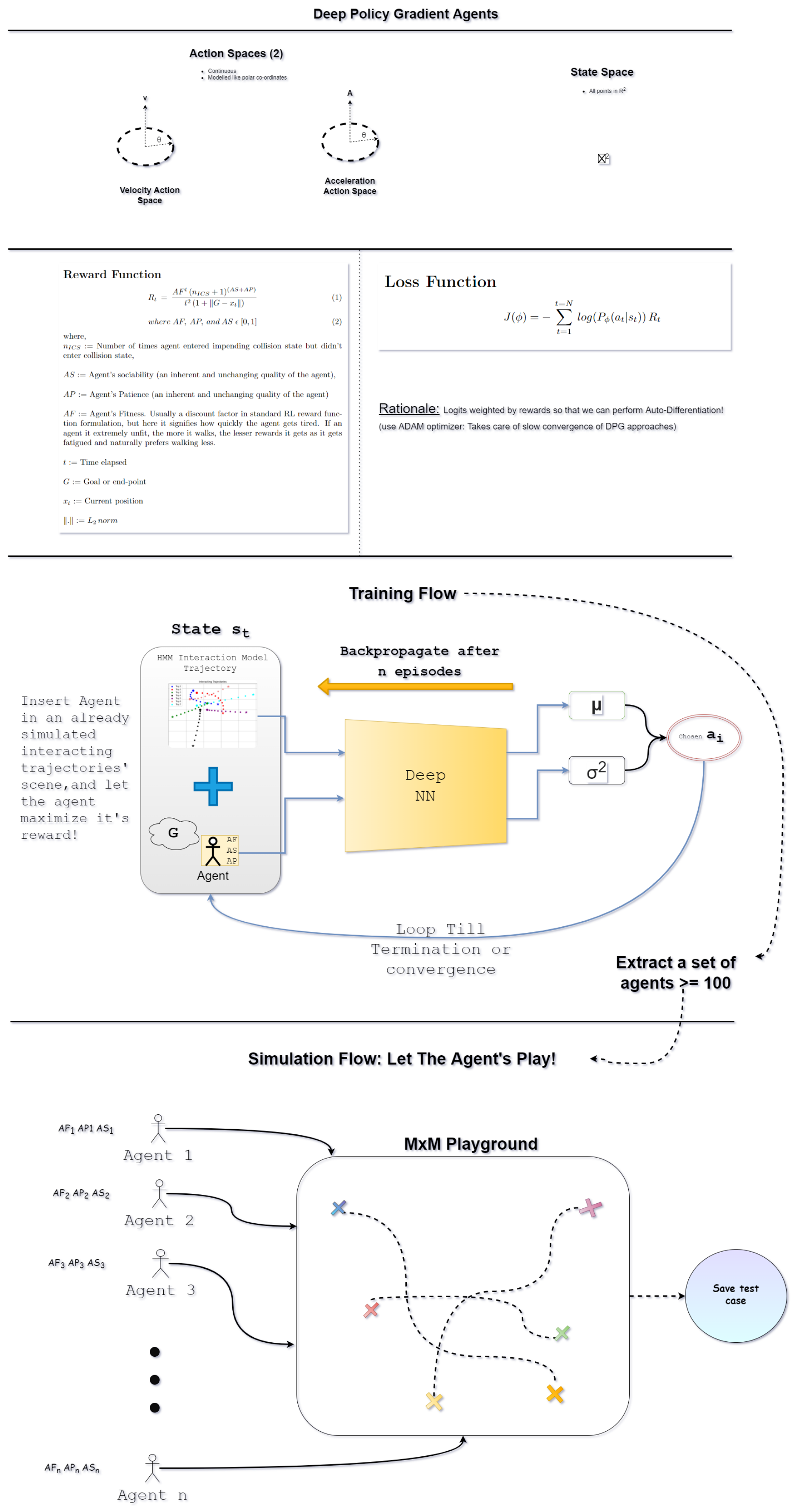
[35] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics (intelligent robotics and autonomous agents). 2005. 1

[36] David Ticoll. Driving changes: Automated vehicles in toronto. 10 2015. 1

[37] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. 1

[38] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders, 2016. 2

[39] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z. Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15159–15168, June 2022. 1


Deep Policy Gradient Agents
Action Spaces (2)
Continuous
Modelled like polar co-ordinates
V
θ
Velocity Action Space
A
θ
Acceleration Action Space
State Space
All points in R^2
Training Flow
State s_t
HMM Interaction Model Trajectory
Backpropagate after n episodes
Insert Agent in an already simulated interacting trajectories' scene, and let the agent maximize it's reward!
Deep NN
μ
σ2
Chosen a_i
G
AF
AS
AP
Agent
Loop Till Termination or convergence
Extract a set of agents >= 100
Simulation Flow: Let The Agent's Play!
AF_1 $AP1$ AS_1
Agent 1
AF_2 AP_2 AS_2
Agent 2
AF_3 AP_3 AS_3
Agent 3
AF_n AP_n AS_n
Agent n
MxM Playground
Save test case

Reward Function

$$R_t = \frac{AF^t\,(n_{ICS}+1)^{(AS+AP)}}{t^2\,(1+\|G-x_t\|)} \tag{1}$$

$$where\ AF,\ AP,\ and\ AS\ \epsilon\ [0,1] \tag{2}$$

where,
n_{ICS} := Number of times agent entered impending collision state but didn't enter collision state,

AS := Agent's sociability (an inherent and unchanging quality of the agent),

AP := Agent's Patience (an inherent and unchanging quality of the agent)

AF := Agent's Fitness. Usually a discount factor in standard RL reward function formulation, but here it signifies how quickly the agent gets tired. If an agent it extremely unfit, the more it walks, the lesser rewards it gets as it gets fatigued and naturally prefers walking less.

t := Time elapsed

G := Goal or end-point

x_t := Current position

$\|.\|$:= $L_2\, norm$

Loss Function

$$J(\phi) = -\sum_{t=1}^{t=N} log(P_\phi(a_t|s_t))\,R_t$$

Rationale: Logits weighted by rewards so that we can perform Auto-Differentiation!
(use ADAM optimizer: Takes care of slow convergence of DPG approaches)

Figure 12. Deep Policy Gradients Agents for emulating a live drone-shot scene with human-like agents reaching their goals.

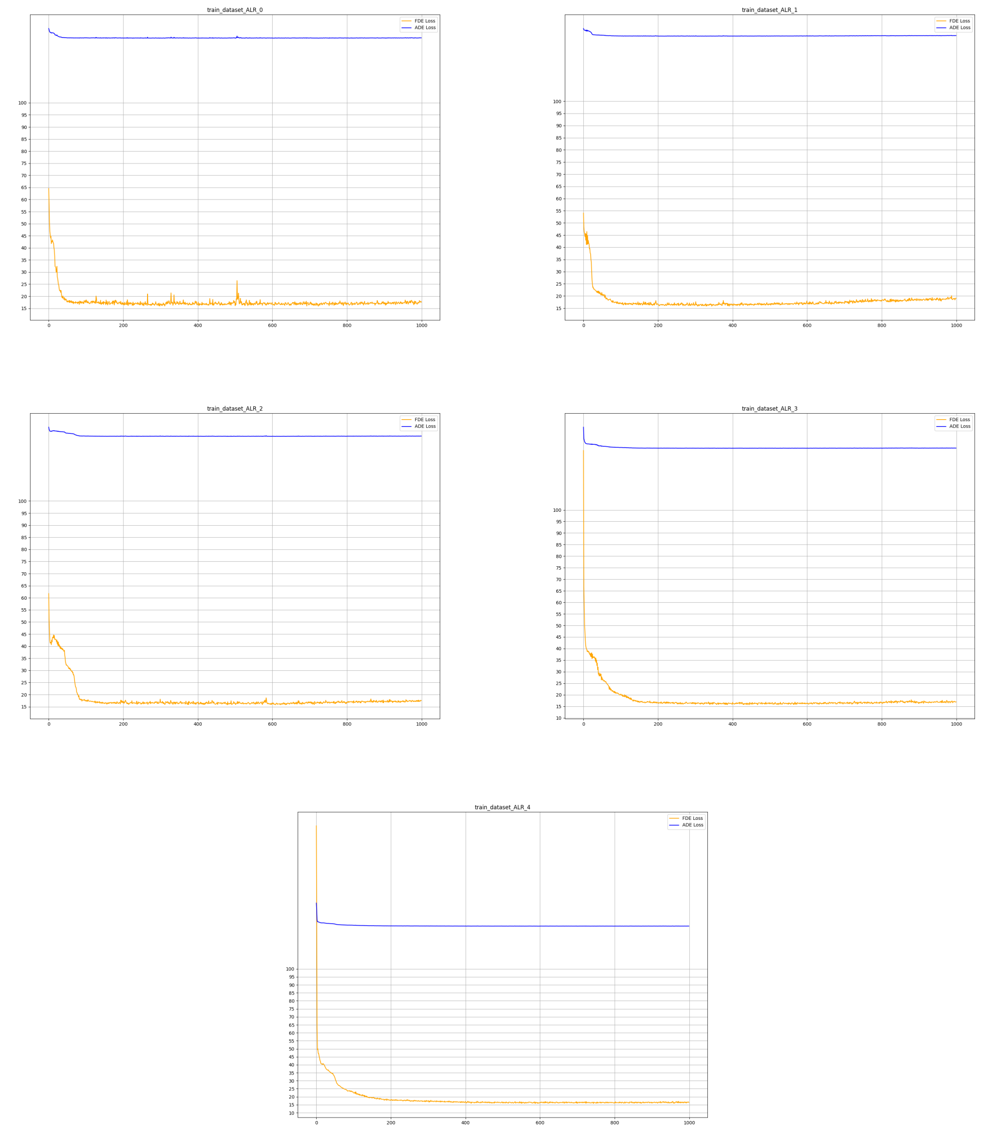


Figure 13. ADE & FDE Loss vs Epochs plots. Non-Decoupled, SIREN upgraded PECNet. Learning rates: 0.001, 0.0005, 0.0003, 0.0002 and 0.0001 respectively. See table 7

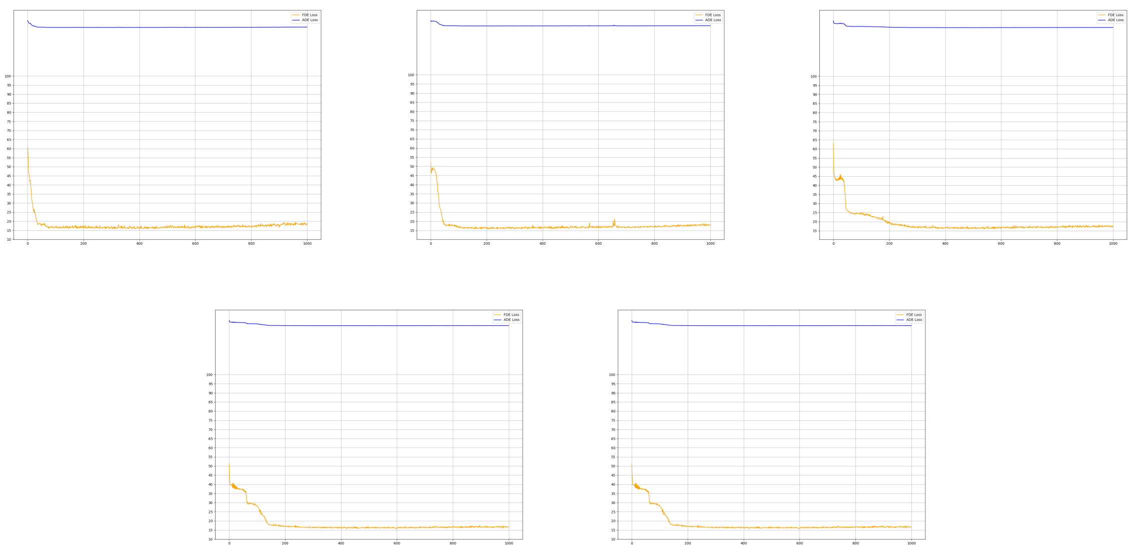


Figure 14. ADE & FDE Loss vs Epochs plots. Decoupled system. Learning rates: 0.001, 0.0005, 0.0003, 0.0002 and 0.0001 respectively. See table 8

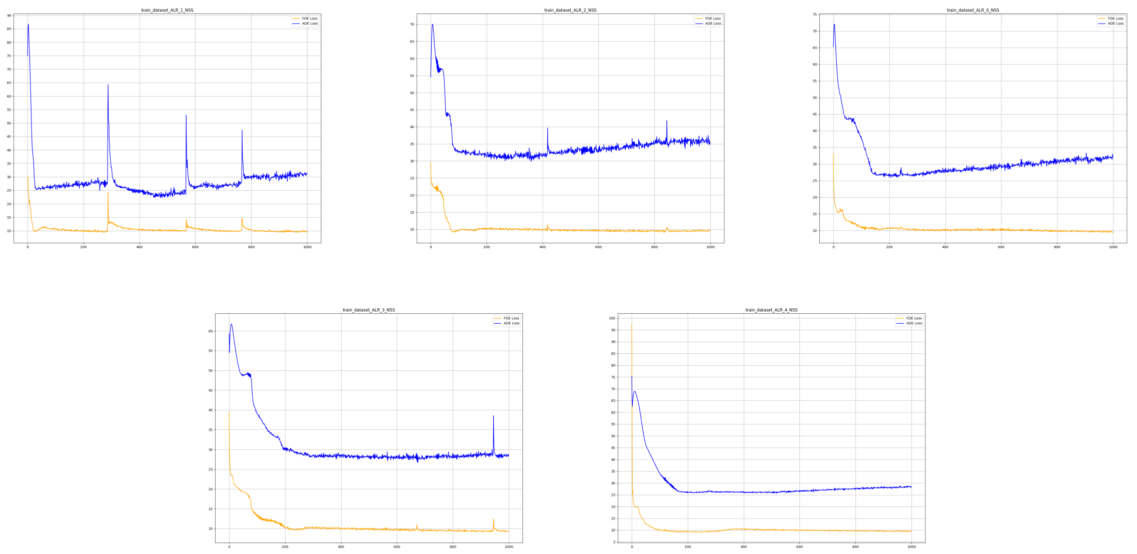
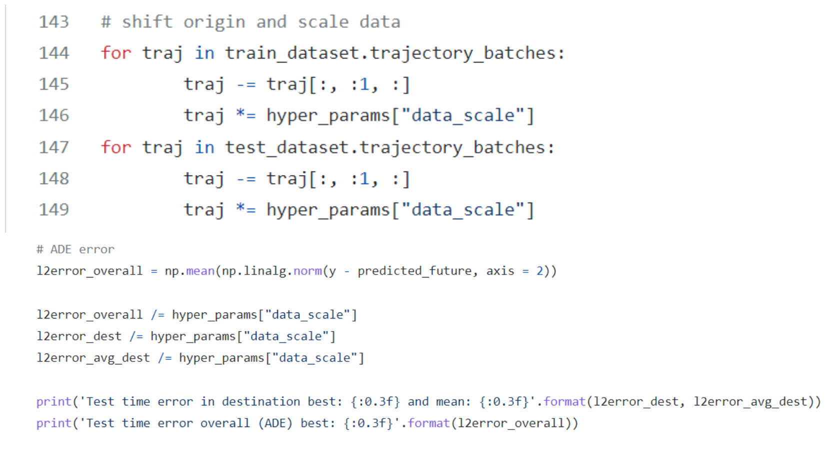


Figure 15. ADE & FDE Loss vs Epochs plots. No Standardization. Learning rates: 0.001, 0.0005, 0.0003, 0.0002 and 0.0001 respectively. See table 9



```
# shift origin and scale data
for traj in train_dataset.trajectory_batches:
        traj -= traj[:, :1, :]
        traj *= hyper_params["data_scale"]
for traj in test_dataset.trajectory_batches:
        traj -= traj[:, :1, :]
        traj *= hyper_params["data_scale"]
```

```
# ADE error
l2error_overall = np.mean(np.linalg.norm(y - predicted_future, axis = 2))

l2error_overall /= hyper_params["data_scale"]
l2error_dest /= hyper_params["data_scale"]
l2error_avg_dest /= hyper_params["data_scale"]

print('Test time error in destination best: {:0.3f} and mean: {:0.3f}'.format(l2error_dest, l2error_avg_dest))
print('Test time error overall (ADE) best: {:0.3f}'.format(l2error_overall))
```

Figure 16. Repository code of PECNet: Misuse of hyper-parameter: $data_scale$ to divide the ADE loss metric.

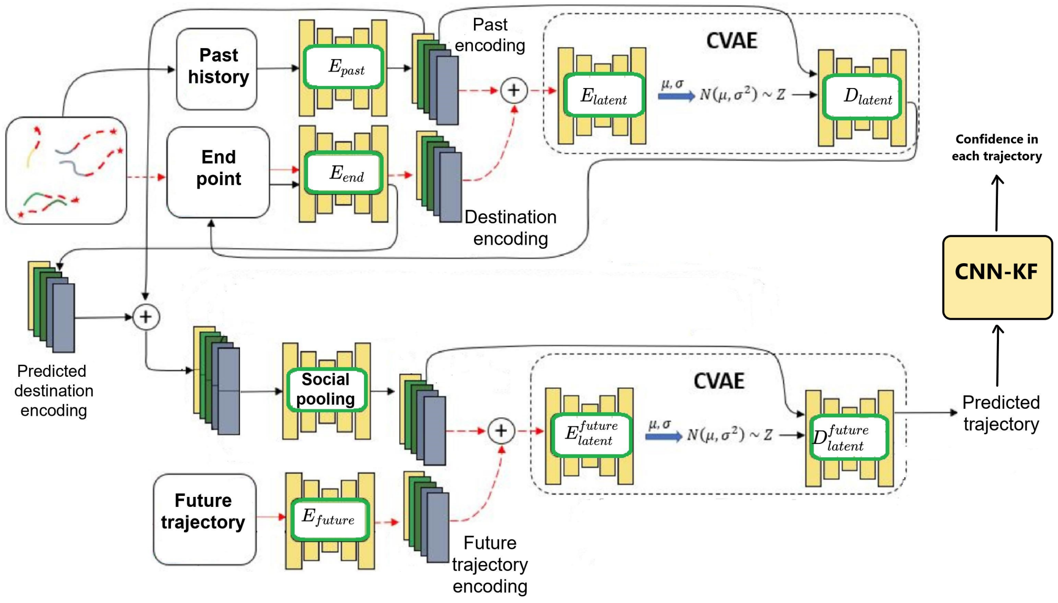


Figure 17. Proposed multi-modal PECNet