Agent Trajectory Prediction in Urban Traffic Environments via Deep Reward Learning

Khaled Saleh[1], Adriana-Simona Mihaita[2] and Stephan Chalup[1]
[1] School of Information and Physical Sciences, University of Newcastle, Australia
[2] Faculty of Engineering and IT, University of Technology Sydney, Australia
Email: khaled.saleh@newcastle.edu.au

***Abstract*— In this paper, we address the problem of learning and modelling the behaviours of agents in urban traffic environments such as pedestrians using their trajectories. Existing state-of-the-art methods primarily rely on data-driven approaches to predict future trajectories. However, these approaches often overlook the influence of the physical environment on agents' decisions and struggle to model longer sequential trajectory data effectively. To overcome these limitations, we propose a novel hybrid framework in this paper that uses the attributes of the physical environment to predict the future trajectory that a travel agent might take on the road. First, we capture agents' preferences in various urban traffic environments using a deep reward learning technique. Next, leveraging the learned reward map and short past motion trajectories of the agents, we employ a probabilistic data-driven sequential model based on transformer networks to provide robust long-term forecasting of agents' trajectories. In our experiments, the proposed framework was evaluated on a large-scale real-world dataset of agents in urban traffic environments. Compared to state-of-the-art techniques, our framework achieves a substantial improvement by a significant margin.**

I. Introduction

Learning and modelling behaviours of goal-based agents such as pedestrians, cyclists and motorists in urban traffic environments have been extensively studied over the past few years. Recently, planning-based approaches, especially those based on deep reinforcement learning (DRL) techniques [1], have been achieving state-of-the-art results on a number of benchmarks. That being said, almost all of these benchmarks are based on either a controlled, simulated, or gamified environment [2], [3] where the state-space is deterministic and/or there is a clear reward signal that can encourage/penalize the agent from performing certain actions. On the other hand, in a real dynamic environment with its uncertain and stochastic nature such as the urban traffic environment, the notion of reward is not quite prevalent. Additionally, when it comes to goal-based agents, their goals are commonly hidden (latent) and hard to infer. As a result, when DRL techniques are transferred from such controlled environments to real dynamic environments they often face challenges when it comes to modelling/learning agents trajectories.

In the literature, the problem of modelling agents' behaviours in urban traffic environments is commonly achieved via predicting their future trajectories which acts as a proxy for inferring their intentions. Planning-based and data-driven approaches are considered some of the most commonly used approaches when it comes to the problem of agent trajectory forecasting. In planning-based approaches, it is assumed that the agent is rational and is governed by some hidden goal. Thus, all planning-based approaches [4]–[6] start first with a goal inference stage that estimates all potential end goals for the agent. For example, in [4], they utilised particle filter and Gaussian Mixture Models (GMMs) for goal inference, and with the help of a map of the surrounding environment, they forecast a probability distribution over potential paths to the estimated goals. On the other hand in data-driven approaches [7]–[10], and unlike planning-based approaches, the goals of the agents are not required for their operation, they directly learn complex behaviours of agents using large datasets. The majority of data-driven approaches rely on deep recurrent neural network architectures such as long short-term memory (LSTM). Despite the capability of LSTMs to implicitly model and capture the inherent dependency between the consecutive observations of agents' trajectories, they are argued to be inefficient when it comes to modelling longer sequential data [11], [12]. Furthermore, the majority of data-driven approaches based on LSTMs were only modelling the interactions between agents, neglecting the effect of the physical environment on the agents' actions [7], [13].

Given the limitations of the aforementioned approaches, in this work, we are proposing a framework that combines the best of the two worlds of planning-based approaches and data-driven approaches. We first learn the reward function of the physical traffic environment by just observing the demonstrated historical trajectories of agents via the deep inverse reinforcement learning (IRL) technique. Then, using the learned reward function alongside short past motion trajectory of agents, we learn a probabilistic data-driven sequential model based on the transformer networks architecture [14] that have been achieving state-of-the-art (SOTA) results on a number of tasks such as natural language processing and computer vision.

The remaining sections of this paper are structured as follows: Section II outlines the problem formulation and introduces the proposed solution. The datasets employed and the validation method are elaborated in Section III. Lastly, Section IV concludes the paper.

II. Proposed Method

In this section, we first start with our formulation for the trajectory prediction problem. Then, we present the different contextual information we took into account as input to our proposed framework (shown in Figure 1). Then, we describe the architecture of our proposed context-augmented transformer model and its implementation details.

A. Problem Formulation

In the formulation for the agent behaviour modelling problem via trajectory prediction in a traffic environment, we cast the problem as a probabilistic sequence prediction problem. Given a sequence of past trajectory observations x as well as a reward map r that represents the agent's preference in an urban traffic environment. In return, we will anticipate the probability density $P(y|x,r)$ of the agent's future trajectory y. In order to achieve a probabilistic sequence prediction model, we are proposing a novel architecture that is consisting of the famous transformer networks model with a mixture density network on top of it [15]. For recovering a reward map that can accurately capture the agents' preferences and actions, we will rely on a deep IRL technique, also known as deep reward learning [16].

B. Preliminaries

Before we get into the details of the deep IRL technique, we will be relying on in our proposed methodology. We start with a brief summary about some of the preliminaries associated with IRL. The setup for any IRL technique is mainly governed by some type of Markov Decision Process (MDP). MDP is a commonly employed framework for modelling the dynamics of decision-making processes [17]. It provides a structured approach to understanding and optimizing complex systems. At its core, an MDP can be defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$, encompassing the elements necessary for analysis and decision-making. Here, $\mathcal{S}$ represents the state space of the system, $\mathcal{A}$ denotes the possible actions, $\mathcal{T}$ captures the transition model which defines the system dynamics, and finally r represents the reward function. Operating within an MDP involves navigating through a series of interconnected states and actions. As the process unfolds, a sequence of state-action pairs $\{s_0, a_0, s_1, a_1, \ldots\}$ emerges. To guide decision-making, a policy π, is employed, mapping these sequences as $(\mu_0, \mu_1, \ldots)$. At any given time t, the mapping $\mu_t(\cdot)$ specifies the action $a_t = \mu_t(s_t)$ to undertake while in state s_t. Within an MDP, the ultimate objective is to discover an optimal policy π^* that maximizes the expected sum of rewards accumulated over time. This entails finding the most effective strategy for achieving long-term goals and optimizing the decision-making process within the system.

In a real-world setting, we have access to the specifications of Markov Decision Processes (MDP), except for the unknown reward function r. Instead, we are provided with a collection of demonstrations $\mathcal{D} = \{\zeta_0, \zeta_1, \ldots, \zeta_N\}$ by a demonstrator. Each demonstration trajectory ζ_i in $\mathcal{D}$ consists of state-action pairs, denoted as $\zeta_i = \{(s_0, a_0), (s_1, a_1), \ldots, (s_T, a_T)\}$. The main objective of IRL is to recover the reward function r from the given demonstrations $\mathcal{D}$, which effectively captures the agent's preferences. In practical applications, obtaining a reward function for every action-state pair in $\mathcal{D}$ can be challenging, especially when the state space is extensive. Therefore, a common approach in IRL methods is to extract a feature vector f that provides the most relevant characterization for each possible action based on the available demonstrations $\mathcal{D}$.

The advantage of the proposed deep MaxEnt over the traditional MaxEnt technique is that the feature representation f can be easily learned directly without being hand-crafted based on prepossessing such as segmentation and manually defined distance metrics. Additionally, the weighted linear combination of feature values for reward approximation in the traditional MaxEnt technique makes it sub-optimal if the true reward can not be accurately approximated by a linear model.

C. Deep Reward Learning via IRL

In our work, we will be utilising one of the recent successful IRL techniques that rely on deep neural networks, the deep MaxEnt technique proposed in [16], [18] to approximate and characterise each possible action from the set of demonstrations $\mathcal{D}$. In the traditional MaxEnt IRL technique, the reward function is determined by combining the feature values vector f using a weighted linear calculation according to Eq. 1.

$$\begin{aligned} r &= h(f, \theta) \\ &= \theta^T f \end{aligned} \tag{1}$$

where θ is the weights of the features vector. On the other hand, in our proposed deep MaxEnt IRL approach, a deep neural network takes as an input the state features x and maps them to state reward r using the proposed deep neural network (Deep MaxEnt IRL) which is governed by the network parameters $\theta_{1,2,..n}$ according to Eq. 2.

$$\begin{aligned} r &\approx h(f, \theta_1, \theta_2, \ldots, \theta_n) \\ &= h_1\left(h_2\left(\ldots\left(h_n\left(f, \theta_n\right), \ldots\right), \theta_2\right), \theta_1\right). \end{aligned} \tag{2}$$

In our use-case, the state features x of our urban-traffic environment is a bird's eye view representation image I of the static scene around our agents. Using the proposed deep MaxEnt IRL technique, the exponentiated sum of rewards along the trajectory ζ_i proportionally determines the probability distribution of the trajectory ζ_i, which encapsulates the preference of our agent of interest. The calculation of ζ_i can be easily accomplished using Eq 3 after the substitution in Eq 2.

$$P(\zeta_i) \propto \exp \sum_{(s,a)\in\zeta_i} r_{s,a} \tag{3}$$

Hence, our deep reward learning MaxEnt IRL approach aims to learn a reward function where the log-likelihood of the observed expert demonstrations $\mathcal{D}$ in the training dataset is maximised. In order to solve this, the log-likelihood

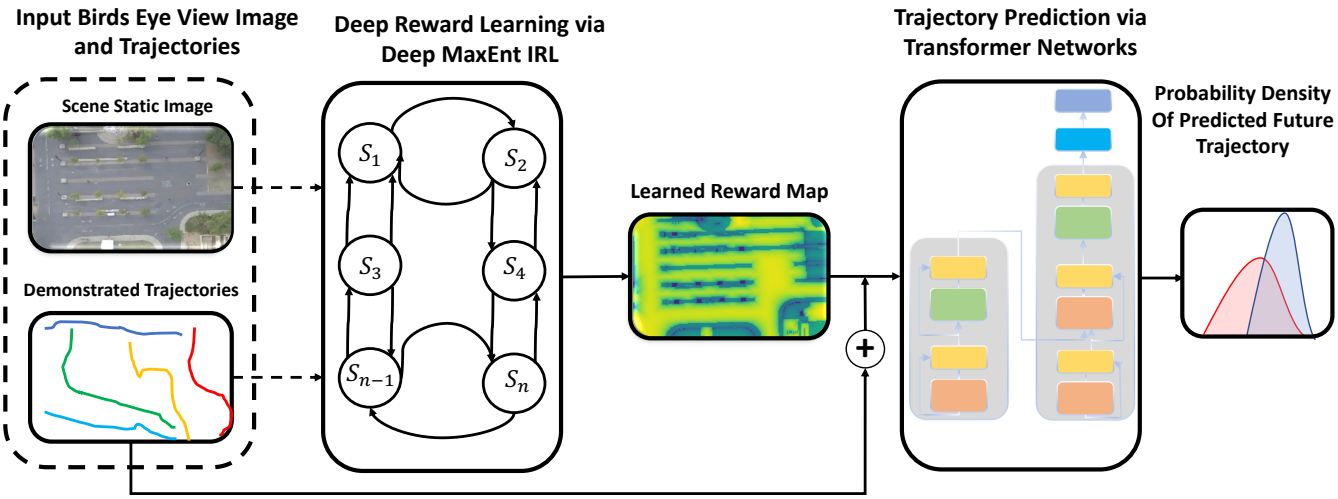

Fig. 1. The proposed framework for long-term trajectory prediction of agents in urban traffic environment.

Algorithm 1: Deep Maximum Entropy IRL

input : $\mathcal{S}, \mathcal{A}, \mathcal{T}, f, \phi_{\zeta_i}$
output: Optimal set of weights $\hat{\theta}$
$\theta^1 = IRL\_initWeights()$
for *n=1:N* **do**
  $r^n = ConvNet\_forward(f, \theta^n)$
  $\pi^n = IRL\_valIter(\mathcal{S}, \mathcal{A}, \mathcal{T}, r^n)$
  $\phi_\theta = IRL\_stVisitFreq(\mathcal{S}, \mathcal{A}, \mathcal{T}, \pi^n)$
  $\frac{d\mathcal{L}^n_\theta}{dr^n} = \phi_{\zeta_i} - \phi_\theta$
  $\frac{d\mathcal{L}^n_\theta}{d\theta^n} = ConvNet\_backprop(f, \theta^n, \frac{d\mathcal{L}^n_\theta}{dr^n})$
  $\theta^{n+1} = IRL\_weightsUpdate(\theta^n, \frac{d\mathcal{L}^n_\theta}{d\theta^n})$
end

Algorithm 2: IRL_valIter

$V(s) = -\infty$
for *n=N:1* **do**
  $V(s_{goal}) = 0$
  $Q^n(s,a) = r(s,a) + _{\mathcal{T}(s,a,s')}[V^n(s')]$
  $V^{n-1}(s) = \text{soft}\max_a Q^n(s,a)$
end
$\pi(a,s) = \exp^{Q(s,a)-V(s)}$

Algorithm 3: IRL_stVisitFreq

$\phi(s_{start}) = 1$
for *n=1:N* **do**
  $\phi_{s_{goal}} = 0$
  $\phi_s^{n+1} = \sum_{s',a} \mathcal{T}(s,a,s')\pi(a,s')\phi^n(s')$
end
$\phi_\theta = \sum_n \phi_s^n$

gradient $\mathcal{L}_\theta$ can be approximated via the stochastic gradient descent algorithm as follows:

$$d\mathcal{L}_\theta / d\theta = \sum_{\zeta_i \in \mathcal{D}} \left(\phi_{\zeta_i} - \phi_\theta\right) dr_\theta / d\theta \tag{4}$$

where ϕ_{ζ_i} represents the actual frequency of the state visitations (SVF) according to the trajectory ζ_i demonstrated by the expert agent and ϕ_θ is the expected SVF required by the MaxEnt policy given the parameters for the current reward θ. ϕ_θ and $dr_\theta d\theta$ can be calculated according to Algorithm 1, 2 and 3. From Algorithm 1, the reward r^n is obtained using a deep convolutional neural network (ConvNet) model. In our implementation, the ConvNet model consists mainly of convolutional and pooling layers, allowing the learned reward function to apply to novel unseen urban traffic environments with different configuration of scene elements. The backbone of the ConvNet model is the first two residual blocks of a pre-trained ResNet34 model on ImageNet [19] which acts as an automatic spatial feature extractor of the input bird's eye view representation image I of the static scene around our agents of interest. The extracted spatial features are 2D representations corresponding to our state space $\mathcal{S}$. Since the reward function that captures the agents' preference in an urban traffic environment does not rely only on the spatial features. Thus, the history of the agent's trajectory needs to be taken into account as input to the deep ConvNet model. Similar to [20], the scene spatial features are concatenated with the agents' motion trajectory, and the locations of the grid cells which constitute the input feature maps f to the ConvNet reward model. Internally, within the ConvNet model and following the first two residual blocks, the concatenated feature maps f are passed through three additional convolution layers. The first convolution layer has a 2×2 kernel size and the last two convolution layers have a kernel size of 1×1

D. Probabilistic Trajectory Prediction via Transformer Networks

Given the learned reward map from the previous section, we fed it into a novel probabilistic transformer networks model for the task of agents' trajectory prediction. Traditional transformer networks provide deterministic real target values predictions, however in real-life applications, there

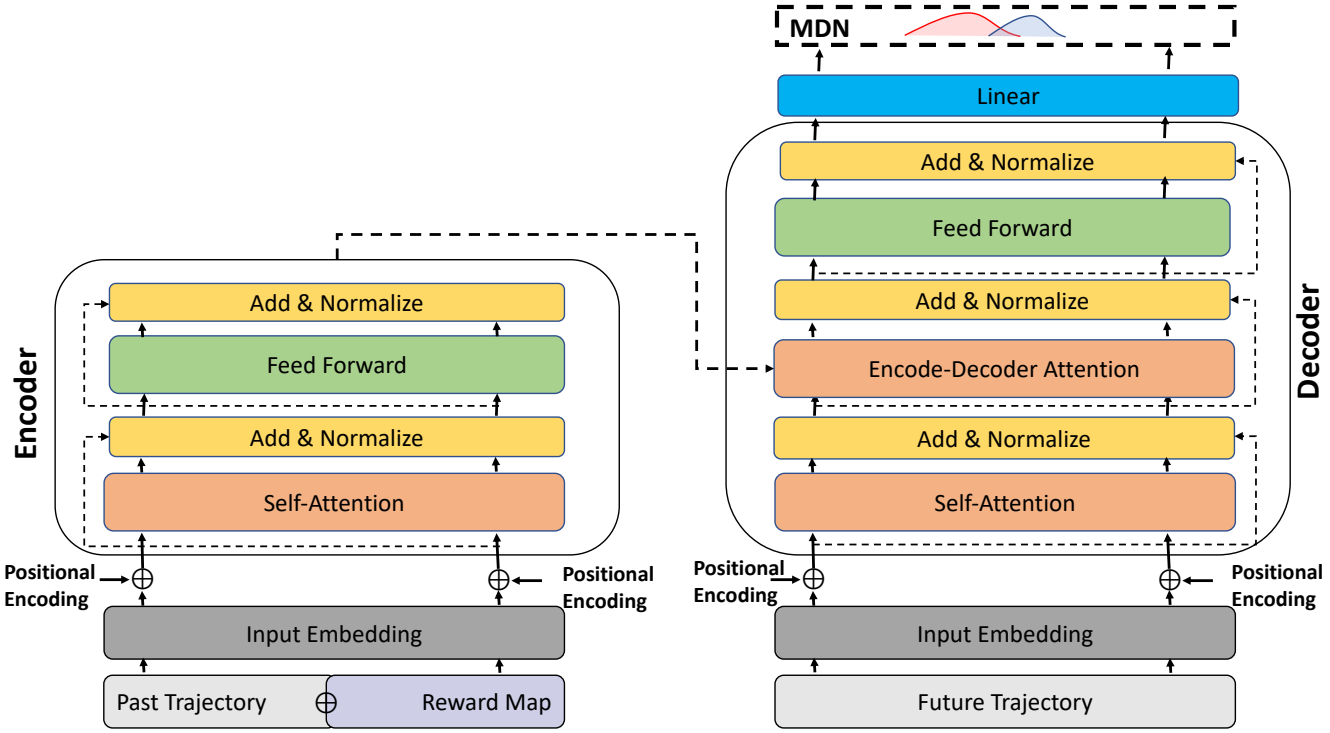

Fig. 2. The building blocks of our probabilistic transformer networks model.

is an inherent uncertainty in agents' actions, especially in urban traffic environments. Thus, in our proposed model, we combine the traditional transformer network architecture with the mixture density network (MDN) [15] which provides probabilistic forecasting about agents' positions rather than deterministic ones. MDN do so by generating a weighted sum of multiple probability distributions that take into consideration the uncertainty associated with agents' movement in urban traffic environments. Additionally, unlike the traditional transformer networks model introduced by Guliari et al. [21] which only considers positional information as input information for their model, our probabilistic transformer networks model exploits the learned reward map in Section II-C. The rationale behind including the learned reward map beside the past trajectory information is to model the preference of agents in different physical parts of an urban traffic environment that is encoded within the learned reward map.

The proposed probabilistic transformer networks model (shown in Figure 2) exhibits the overall architecture, following the same encoder/decoder paradigm commonly found in LSTM-based approaches. However, it achieves greater efficiency by eliminating the recurrence loops present in LSTM networks. The model comprises several main building blocks, including embedding layers, positional encoding layers, a self-attention mechanism, fully-connected feed-forward layers, and an MDN layer. In both the encoder and decoder stages, the embedding layers, serve the purpose of embedding the observed past trajectory (along with the concatenated learned reward map) and the future trajectory of the agents (only utilized during training) into a higher dimensional space d_{model}.

The embedding layer function acts as learnable linear transformations facilitated by weight matrices. Similarly, positional encoding layers, located at the start of the encoder/decoder stages, also play a crucial role. Given the absence of recurrence in the transformer network model, positional encoding layers introduce the concept of order in the input/output trajectory data, essentially providing a time-stamping mechanism. Our model's encoder and decoder units consist of a total of 6 blocks, with each block internally comprising a self-attention head and feed-forward fully connected sub-layers. Additionally, two residual connections and a normalization operation follow each sub-layer. The mapping between the so-called 'query' vectors and the (key, value) vectors is the basis for the functioning of multi-head self-attention, also known as multi-scaled dot-product attention. The dimensions of the query and key vectors are denoted as d_k, while the value vector dimension is denoted as d_v. The attention operation itself involves computing the dot product between the query and key vectors, divided by the square root of d_k, and then passing the result through the softmax function to obtain the weights. Given that the scaled dot-product attention operation is performed a number of times, the query, key, and value vectors are scaled into matrices Q,K,V respectively. The following formula describes how the scaled dot-product attention operation is calculated:

$$\text{Attention } (Q,K,V) = \text{softmax}\left(QK^T/\sqrt{d_k}\right)V \qquad (5)$$

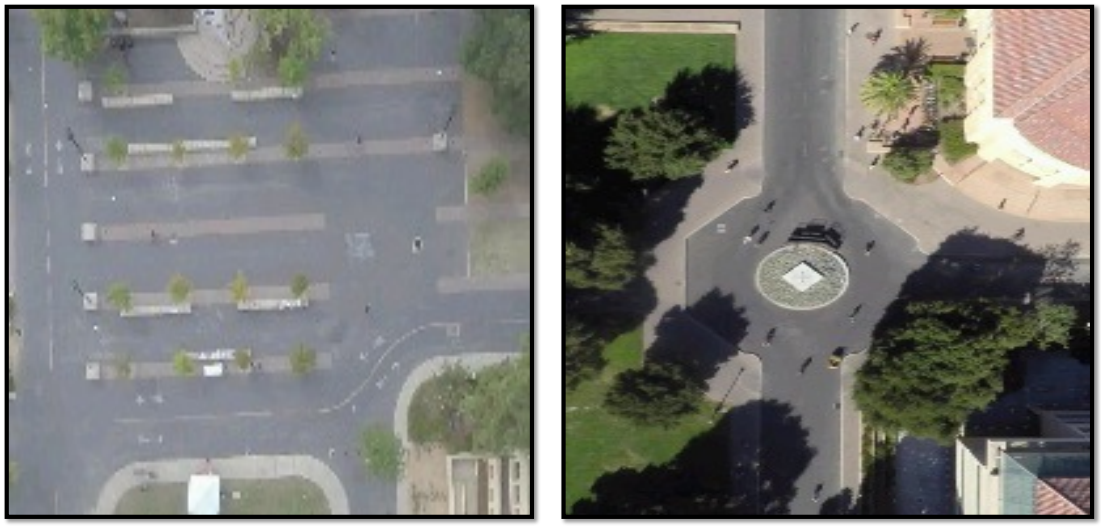

Fig. 3. Sample reference images of the bird's eye view representation of two different scenes from SDD [22].

The encoder stage encodes the embedding input and produces the matrices of queries, keys, and values as the output, which are calculated in Equation 5. These matrices are subsequently transferred to the decoder stage, where, at each prediction step, the decoder compares its own newly generated queries matrix with the keys and values matrices from the encoder, along with the previously decoded prediction. The decoder then iteratively repeats this procedure until it achieves all the required predicted positions of the trajectory. Thanks to the MDN output layer that exists on top of the decoder block, the final output of the model is the probability distribution over the future trajectory of the agents. In order to achieve so, we followed [13], [23], and used the mixture of Gaussian as our probability density function (PDF) for the MDN, which is calculated as follows:

$$P(y_t \mid \mathscr{N}_t) = \sum_{m=1}^{M} \alpha_t^m \mathscr{N}(y_t \mid \mu_t^m, \sigma_t^m, \rho_t^m) \tag{6}$$

$$\begin{aligned} \alpha_t^m &= \frac{\exp(\tilde{\alpha}_t^m)}{\sum_{i=1}^{M} \exp(\tilde{\alpha}_i^m))}, \\ \sigma_t^m &= \exp(\tilde{\sigma}_t^m), \\ \rho_t^m &= \tanh(\tilde{\rho}_t^m) \end{aligned} \tag{7}$$

where $\tilde{\alpha}_t^m$ is the m-th mixture Gaussian weight of the PDF, $\tilde{\sigma}_t^m$ is the m-th mixture Gaussian variance of the PDF and $\tilde{\rho}_t^m$ is the m-th mixture Gaussian correlation of the PDF to be predicted by our model.

Consequently, the log-likelihood of the normal Gaussian distribution against the input real-valued training data can be minimised to achieve the training of the proposed probabilistic transformer networks mode as follows:

$$L(X) = \sum_{t=1}^{T} -\log\left(\sum_{m=1}^{M} \alpha_t^m \mathscr{N}(y_t \mid \mu_t^m, \sigma_t^m, \rho_t^m)\right) \tag{8}$$

where T is the length of the input past trajectory.

III. Experiments and Results

In this section, we will first present the datasets we utilised for training and evaluating the performance of our proposed approach. Then, the details of the setup of our experiment will be outlined. Finally, the quantitative and qualitative results of our proposed approach on real-life datasets will be evaluated and discussed.

A. Dataset

The Stanford drone dataset (SDD) [22] (shown in Figure 3), which is one of the largest datasets used for modelling agent behaviour, has recently been made publicly accessible. SDD was created by capturing video footage using a bird's eye view camera mounted on a drone, flying over the vicinity of the Stanford University campus. The dataset includes annotated bounding boxes for moving targets such as pedestrians, cyclists, and cars in each frame of the video, captured at an approximate frame rate of 28 frames per second. It consists of 60 different scenes, with multiple targets annotated in the videos.

For our experiments, we focused on scenes that had a diverse range of agents and also included other static or dynamic objects commonly found in urban traffic environments. These objects include sidewalks, roads/roundabouts, cars, grass, and buildings. To train, validate, and test our proposed framework, we used the dataset split defined in the TrajNet benchmark [24]. Each split, namely training, validation, and testing, contains different scenes selected from the total 60 scenes available in the dataset. This approach allows us to objectively assess the performance of our framework, particularly in unfamiliar scenes that were not seen during the training phase.

B. Experimental Setup

Similar to the standard approach followed by the previous techniques introduced in the literature, the past trajectory T of 3.2 seconds of each agent's full trajectory in SDD was

TABLE I

PERFORMANCE OF THE PROPOSED FRAMEWORK IN COMPARISON TO OTHER BASELINE APPROACHES ON THE TEST SPLIT OF SDD [22]. THE LOWER IS THE BETTER.

Model	MinADE $_5$	MinADE $_{20}$	MinFDE $_5$	MinFDE $_{20}$
S-GAN [7]	–	27.25	–	41.44
Desire [8]	19.25	–	**34.05**	–
MATF [9]	–	22.59	–	33.53
SoPhie [10]	–	16.27	–	29.38
TF_{DIRL} (ours)	**18.36**	**12.85**	34.57	**21.75**

TABLE II

ABLATION STUDY ON THE EFFECT OF LEARNED REWARD ON THE PERFORMANCE OF OUR PROBABILISTIC TRANSFORMER NETWORK FRAMEWORK USING THE TEST SPLIT OF SDD [22]. THE LOWER IS THE BETTER.

Model	MinADE $_5$	MinADE $_{20}$	MinFDE $_5$	MinFDE $_{20}$
$TF_{Vanilla}$	19.61	13.32	36.97	22.19
TF_{DIRL} (ours)	**18.36**	**12.85**	**34.57**	**21.75**

used as an input to our model and a prediction horizon of 4.8 seconds to be forecasted from our model. Following [25], the adopted frame of reference is from the agent's perspective where the x-axis is aligned along the agents' direction of motion. For the hyper-parameters of the deep reward learning ConvNet model, the input bird's eye view image's resolution is 200×200. On the other hand, for the hyper-parameters of the proposed probabilistic transformer networks model, we chose d_{model} to be 512 and we utilised 8 self-attention heads in both the encoder and decoder stages. We trained both the deep reward learning model and the probabilistic transformer networks model for 250 epochs using Adam optimiser with a learning rate of 0.001.

C. Experimental Results

To quantitatively evaluate the performance of our proposed probabilistic framework, we require a metric to measure the deviation of predictions from the actual future trajectory. Our model, with the assistance of the MDN layer, can generate M mixtures from a Gaussian distribution. Therefore, we need a metric that does not penalize reasonable trajectories generated by the model, even if they do not precisely match the ground truth. Thus, we utilise two metrics, namely, the minimum of M average displacement error (MinADE_M) and the final displacement error (MinFDE_M), which are similar to metrics used in previous studies on trajectory forecasting [7]–[10]. MinADE_M (Equation 9) calculates the average prediction error, considering the L2 norm between the ground truth future trajectory and the closest forecasted trajectory. Conversely, MinFDE_M only focuses on the prediction error for the final predicted location.

$$\text{Min ADE}_M = \min_{i \in \{1,\ldots,M\}} \frac{1}{T_f} \sum_{t=1}^{T_f} \left\| y_t^{GT} - y_t^{(i)} \right\|_2 \tag{9}$$

In the literature, the majority of the works [7], [9], [10] have reported MinADE_M and MinFDE_M on SDD for M=20, the only exception was the approach introduced in [8], which only reported results for M=5. Thus, we report our results in Table I for both M values of 5 and 20. Additionally, we have compared our framework (TF_{DIRL}) with prior developed models that represent the state-of-the-art (SOTA) for the SDD. It is worth noting that the error values reported are in pixels.

The first baseline model is the Social-GAN (S-GAN) [7], which was one of the SOTA techniques for pedestrians' trajectory prediction in crowded spaces. S-GAN model, as the name implies, relies on Generative Adversarial Networks (GANs) to capture and generate socially-acceptable trajectories exploiting the interactions between pedestrians on sidewalks. The second baseline model is the Desire framework introduced in [8], which is a conditional generative model. In Desire, it tries to map the input context and a sample from a simple latent distribution to a trajectory output. The third baseline is the SoPhie [10] approach which is another conditional generative model similar to the Desire framework. The last baseline is the MATF [9] approach, which encodes multiple agents' past trajectory and the scene context into a so-called 'Multi-agent Tensor', then applies convolutional fusion to capture the social interactions between agents.

D. Performance Evaluation

As it can be shown from Table I, our proposed TF_{DIRL} framework has achieved resilient results in terms of MinADE $_5$ and MinADE $_{20}$ and it has outperformed almost all of the baseline models. The only exception was only for the MinFDE $_5$ of the Desire approach which was slightly better than our proposed TF_{DIRL} framework. In order to further evaluate the performance of our proposed framework, In Table II, we are performing an ablation study to validate whether the learned reward map has actually made a difference in the overall performance of our proposed probabilistic transformer network framework. Thus, we have trained another version of our probabilistic transformer network framework, but without taking into account the learned reward map as an input. We refer to this version of the model as $TF_{Vanilla}$. As it can be shown in Table II, our TF_{DIRL} framework continued to achieve better results than the $TF_{Vanilla}$ which proves the importance of the learned reward map on the overall performance of our proposed probabilistic transformer network framework

IV. CONCLUSION

In this work, we have proposed a modular framework for the problem of agents' behaviour modelling via trajectory prediction in urban traffic environments using deep reward learning. We cast the problem as a sequence probabilistic prediction problem, where we first start by learning a reward map that encapsulates the preference of agents in urban traffic environments. The reward map was learned using a deep convolution neural network and an inverse reinforcement learning technique. The learned reward map along with the past agents' trajectories was passed as an

input to a novel probabilistic transformer network model that can provide a multi-modal probability distribution over the agents' most probable future trajectories. The proposed framework was evaluated using two different evaluation metrics on a large-scale real-world dataset of agents in urban traffic environments and it has achieved resilient results. Moreover, the proposed framework was compared against five different state-of-the-art techniques from the literature and it outperformed them by a large margin. In our future work, we will try to explore other sensor modalities other than RGB images to be the input to our framework such as thermal images, LiDAR, RADAR and/or hyper-spectral satellite images.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 50–58.

[5] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3931–3936.

[6] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV 12*. Springer, 2012, pp. 201–214.

[7] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[8] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[9] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 126–12 134.

[10] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[11] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[12] K. Saleh, "Pedestrian trajectory prediction for real-time autonomous systems via context-augmented transformer networks," *Sensors*, vol. 22, no. 19, p. 7495, 2022.

[13] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] C. M. Bishop, "Mixture density networks," 1994.

[16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[17] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.

[18] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, "Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories," *arXiv preprint arXiv:1810.07225*, 2018.

[21] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[22] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[23] K. Saleh, M. Hossny, and S. Nahavandi, "Contextual recurrent predictive model for long-term intent prediction of vulnerable road users," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3398–3408, 2019.

[24] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "Trajnet: Towards a benchmark for human trajectory prediction," *arXiv preprint*, 2018.

[25] N. Deo and M. M. Trivedi, "Trajectory forecasts in unknown environments conditioned on grid-based plans," *arXiv preprint arXiv:2001.00735*, 2020.