# Vehicle Trajectory Prediction At Signalised Intersections

Ayush Rai, Chin Pok Leung, Matthew Balzarolo, Wenhao Guo

## 1 Introduction

The objective of this project is to understand deep learning approaches for real-time motion prediction in autonomous driving systems. Accurate prediction of the movement of surrounding vehicles and objects is crucial for the decision-making process in autonomous driving, and requires efficiency, reliability, and the ability to handle challenges such as traffic lights and the actions of other motorists. This project aims to understand the mechanisms that enable both efficient and reliable implementation of such prediction systems; while focused on the trajectory prediction problem.

We specifically consider the applicability of AI methodologies on the trajectory prediction problem at signalised intersections, with a focus on safety-related fields. More specifically, this involves understanding and predicting the trajectory of a vehicle in a general sense, analysis of trajectories for vulnerable traffic participants, and the importance and impact of traffic light data in the trajectory prediction problem.

The developed methods we have applied in the context of this problem are the Knowledge Informed Generative Adversarial Network (KI-GAN), as proposed by Wei et al.,[1] and the Dynamic Neural Relational Inference (dNRI) model, as proposed by Graber et al. [2]. We utilised the SIND dataset [3], which consisted of traffic data obtained from a signalised intersection in Tianjin, China.

The KI-GAN consists of a trajectory encoder, motion encoder, a physical attributes encoder, and traffic encoder, and incorporates social interaction through the Vehicle Attention Pooling Net (VAP-Net) to interpret interactions between the trajectories. The KI-GAN decoder consists of an LSTM framework to predict future trajectories, and the discriminator evaluates the generated trajectories against actual movements in order to refine its prediction capability.

The dNRI is a conditional variational autoencoder which aims to model agent interactions (latent variables) in a scene as a fully connected graph. The encoder takes historical observations and predictions types of interactions between agents at the current time step, and uses a graph neural network to calculate edge features, a forward LSTM to predict the prior distribution of the latent variable, and a bidirectional LSTM for the interaction posterior distribution. The decoder then conditions latent variables to predict the next point in the agent's trajectory

Our experiments with the KI-GAN were predominantly in line with the results from the Wei et al. paper, including a relatively low Average Displacement Error (ADE) and Final Displacement Error (FDE) of 0.06 meters and 0.13 meters respectively on a 6-second trajectory prediction; and of 0.12 meters and 0.25 meters respectively on a 9-second trajectory prediction. However, when masking the traffic encoder input and retraining the model, significant variance appeared between the results demonstrated by Wei et al. and the experiments detailed in this report. Wei et al. highlighted a significant deterioration of results, with FDEs exceeding a 1-meter error in both the 6-second prediction and 9 second prediction; while our results remained relatively consistent (for the 6-second prediction, an ADE of 0.06 meters and an FDE of 0.12 meters; and for the 9-second prediction, an

ADE of 0.10 meters and an FDE of 0.22 meters). We hypothesise that this was driven by a variance in approach between training and validating the model with the traffic encoder masked, and testing on a consistent data set, versus determining the impact of removing the traffic encoder from a model expecting this input. This raises questions about the importance of encoded traffic signals versus a traditional physics-based trajectory approach in the context of the KI-GAN

We also performed experiments with the KI-GAN for a 12-second trajectory prediction, which was outside the scope included by Wei et al. This resulted in an ADE and FDE of 0.23 meters and 0.48 meters respectively when including the traffic encoder, and an ADE and FDE of 0.21 meters and 0.44 meters respectively when masking the traffic encoder.

In addition, we observed that the KI-GAN struggled with certain classes of vulnerable traffic participants in its ADE prediction, including motorcycles and bicycles. We suspect that this is driven by a combination of nimbleness of these traffic participants and frequent traffic violations committed by these classes in the dataset.

Our experiments with the dNRI appeared to be broadly consistent with the results of Dax et al., [4], in which the dNRI was utilised with the iND dataset (an intersection dataset which did not include traffic light data). More specifically, we performed similar experiments to the KI-GAN with 6 second and 9 second trajectory predictions with a traffic encoder enabled and masked. For the 6-second prediction, with the traffic encoder enabled, we obtain an ADE of 0.78 meters, and an FDE of 1.98 meters. Masking the traffic encoder for the same prediction length results in an ADE of 1.00 meters and an FDE of 2.48 meters, which was a significantly worse result. For the 9-second prediction, with the traffic encoder enabled, we obtained an ADE of 0.86 meters, and an FDE of 2.09 meters. When masking the traffic encoder, we obtained an ADE of 1.17 meters, and an FDE of 2.86 meters - once again, demonstrating a deterioration of results after excluding the traffic encoder.

For completeness, Dax et al. reported an ADE of 0.56 meters and an FDE of 4.29 meters, but this was performed on a different dataset, with a condensed observation window, and did not disclose the prediction time frame.

In addition, the dNRI appeared to struggle with the same classes of vulnerable traffic participants in its ADE prediction - motorcycles and bicycles. We suspect that this is for the same reasons as with the KI-GAN (nimbleness and traffic violations).

In contrast with the KI-GAN, the dNRI seemed to report a demonstrable benefit due to the inclusion of the traffic encoder. While the ADE and FDE of the dNRI are too high to be of practical use without further tuning and additional computational resources, they do demonstrate that traffic encoder information may be useful in certain contexts, and the dNRI may benefit further from the incorporation of additional contextual information.

# 2 Related Work

As part of this project, we undertook a literature review in order to understand the varied approaches taken to the vehicle trajectory prediction problem, both generally and specifically at intersections. More broadly, the related work in this field can be classified into the following categories:

- Deep learning models for vehicle trajectory prediction;
- GANs and the incorporation of traffic signals into trajectory prediction;
- Graph-based and relational models;
- Transformer-based approaches; and
- Diffusion models for trajectory prediction.

## Deep Learning Models For Vehicle Trajectory Prediction

Several researchers have developed deep learning models tailored for predicting vehicle trajectories at signalised intersections. Abdelraouf et al.[5] utilised sequence-to-sequence Long Short-Term Memory (LSTM) networks to predict future vehicle positions and orientations across intersections. Kawasaki and Tasaki[6] created a model focused on predicting turning trajectories by considering the intersection's geometry and vehicle speeds. Cao et al.[7] introduced a vision-based model that predicts vehicle maneuvers, such as left turns, right turns, and going straight, to prevent traffic collisions. Despite these advancements, many of these tailored deep learning models still fail to adequately account for the influence of traffic signals, which play a crucial role in the behaviour of traffic participants at intersections.

## GANs and the Incorporation of Traffic Signals Into Trajectory Prediction

This led to the logical development of researches taking different approaches to trajectory prediction at intersections; and ultimately, the incorporation of traffic signals into trajectory prediction. The key usage of GANs in trajectory prediction started from social GANs[8] which introduced a new pooling mechanism that captures human-human interactions by creating a global pooling vector which outperformed all existing models. Roy et al.[9] incorporated vehicle size into their GAN-based model for both signalised and non-signalised intersections, but did not directly consider traffic signal status. Sophie[10] adds onto the Roy by jointly modelling social and physical interactions. Social BiGAT[11] enhances the social interaction by introducing Graph attention. Oh and Peng[12] subsequently focused on how traffic lights impact vehicle speed rather than trajectory variation. Zhang et al.'s D2-TPred model [13] stands out as it uses a spatial dynamic interaction graph (SDG) and a behaviour dependency graph (BDG) to account for traffic light-induced trajectory discontinuities. Finally, the approach taken by Wei et al. in the KI-GAN model [1] aimed to demonstrate the inclusion of traffic signals through the use of a traffic encoder into a social-GAN model could improve signalised intersection trajectory prediction performance.

These approaches demonstrate the recognition that these contextual clues can have an impact on driver behaviour.

## Graph-based and relational models

Another approach taken to the vehicle trajectory problem is through the use of graph-based and relational models which use VAE. The Variational Autoencoder (VAE) was introduced by Kingma et al.[14] (2013) featuring the reparameterisation trick to model P(Y—z). Sohn et al. (2015)[15] further introduced the Conditional VAE (CVAE) framework, P(Y—X,z), particularly for computer vision tasks. To advance the interaction representations, Kipf et al. (2018)[16] developed the Neural Relational Inference (NRI) model, which encodes edges as categorical latent variables and was evaluated on NBA player trajectories. Salzman et al. (2020)[17] extended this with Trajectron++ by utilising a heterogeneous Graph Neural Network (GNN) to predict physically feasible trajectories and encoding nodes as categorical latents. Graber et al., 2020 introduced the dynamic Neural Relational Inference (dNRI)[2] model which explicitly captures the evolving interactions between entities, predicting future trajectories rather than just analysing existing data. While some trajectory prediction models use latent variables to represent states or scenes, dNRI focused on interactions, with potential for further improvement by incorporating auxiliary losses into the Evidence Lower Bound (ELBO). Tang et al. (2023)[18] applied the NRI framework to reinforcement learning-based autonomous driving in the GNRI model, grounding edge latent variables using an expert-designed reward function. Recently, Dax et al. (2024)[19] proposed the dynamic Graph VAE (dG-VAE), leveraging dynamic NRI (Graber et al., 2020) for multi-vehicle motion prediction.

## Transformer-based approaches

Another approach taken to the trajectory prediction problem includes transformer-based approaches. One such method is the MTR++ [20], in which Shi et al. (2023) jointly predicted the distribution of the future trajectories (vehicles, cyclists, pedestrians) given the geometry of lanes and past trajectories. MTR++ utilises a transformer

for pooling, and learnable intention query to refine the predictions. Based on this model, it appears that this could potentially be used as a standalone approach, or leveraged through the intention query module.

### Diffusion Models

Finally, one other category of approaches taken to this problem are based on diffusion models. Methods, like the reversed motion indeterminacy diffusion (MID)[21] , utilise a parameterised Markov chain and a transformer-based diffusion model. This allowed the researchers to present a framework to reduce the trajectory indeterminacy progressively and balance the diversity and indeterminacy by adjusting the length of the chain. Another example is the MotionDiffuser[22] approach, which aims to predict the joint distribution of multi-agent trajectories.

## 3 Methods

### Methods Summary

In this section, we consider the application of two methods to the Vehicle Trajectory Prediction Problem - the KI-GAN[1] and the dNRI.[2] As noted in the Related Work section, we also initially considered the application of the MRI++ model[20] and the MID model[21]; however, due to time limitations, we focused our efforts on the former two methodologies.

### KI-GAN

#### Method Description

**Framework of KI-GAN:** The Knowledge-Informed Generative Adversarial Network (KI-GAN)[1] is created to predict vehicle trajectories at intersections. It uses multiple encoders to process diverse data series, capturing complex vehicle dynamics and interactions. Encoders: There are four specialised encoders:

- Trajectory Encoder: Processes spatial coordinates of vehicles.
- Motion Encoder: Handles dynamic vehicle states like velocity and acceleration.
- Physical Attributes Encoder: Encodes vehicle-specific information such as type and dimensions.
- Traffic Encoder: Processes traffic state information, including traffic light status.

$$F_{\text{combined}} = \text{Concat}(E_{\text{traj}}, E_{\text{motion}}, E_{\text{phy}}) \tag{1}$$

**Social Interaction:** Features from the encoders are combined and fed into a pooling layer, which aggregates contextual information from neighbouring trajectories. The Vehicle Attention Pooling Net (VAP-Net)2 is used to interpret interactions between vehicle trajectories.

$$P = \text{VAPNet}(F_{\text{combined}}, (X, Y), V) \tag{2}$$

**Decoder:** The decoder uses an LSTM framework4 to forecast future vehicle trajectories, integrating the pooled features and encoded traffic state. Some noise is introduced here to add degree of randomness, hence generating a range of future trajectories. This contributes to the robustness of the model.

$$F_{\text{recombined}} = \text{Concat}(P, F_{\text{combined}}, E_{\text{traffic}}) \tag{3}$$

$$\text{Trajectory}_{\text{future}} = \text{LSTM}([\text{F}_{\text{recombined}}; z]) \tag{4}$$

**Discriminator:** Evaluates the generated paths against actual vehicle movements, enhancing the fidelity of trajectory generation through feedback to the Generator. The Discriminator uses a GAN loss function5 and L2 loss metric6 to refine trajectory predictions.

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{5}$$

$$L = \min_k \|Y_i - \hat{Y}_i^{(k)}\|_2 \tag{6}$$

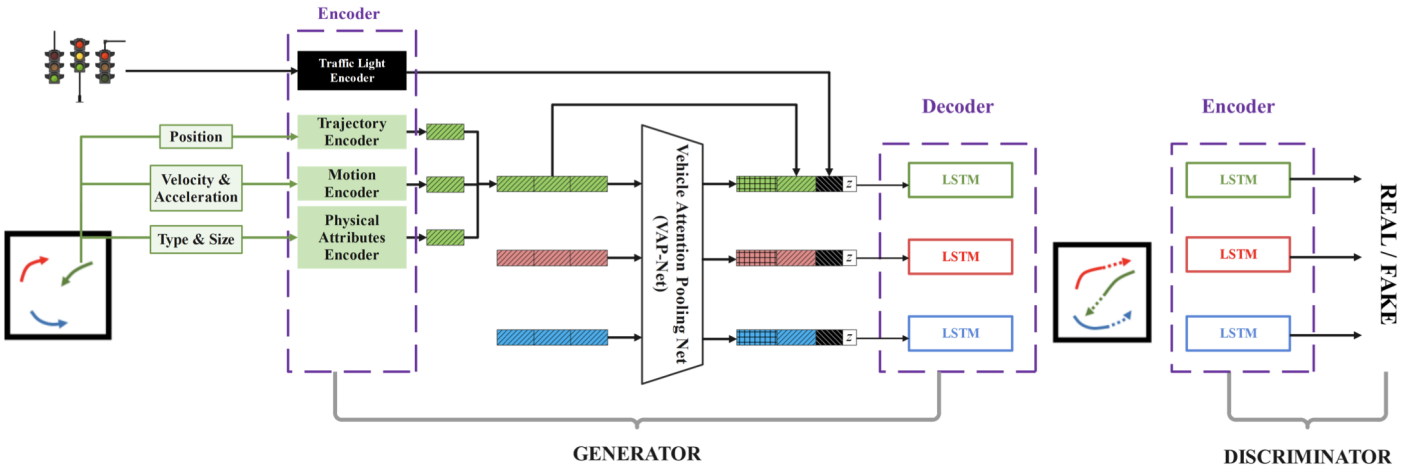



Figure 1: KI-GAN Architecture [1]

### Data Input Format

As shown in Figure 2, the input of Trajectory Encoder is of shape(12, N, 2). Similarly, the Motion Encoder takes in a tensor of shape(12, N, 4), corresponding to the ”state” section in Figure 2. The input of Physical Attributes Encoder are two tensors: one is shape(12, N, 2) representing the sizes and the other is shape(12, N, 1) representing traffic signals.

### Hyperparameters and Other Details

Regarding hyperparameters, the researchers selected a batch size of 64 and trained the model for 50 epochs. The learning rate for the Generator was set to 0.001, and for the Discriminator, it was set to be 0.0005. For the purposes of our experiments, we have retained these values. Increasing the batch size to improve the efficiency of the model appeared to have an adverse impact on the path generation made by the model; and training the model for additional epochs resulted in less flexibility to train for multiple scenarios due to computational constraints.

The initial model captured also did not have noise enabled, which led to straight-line trajectories only. After visualising these trajectories, uniform noise was incorporated as an argument to allow for variance in paths generated, which allowed the model to predict curved trajectories. Uniform noise was selected here due to its inherent randomness over Gaussian noise.

## Dynamic Neural Relational Inference

### Method Description

dNRI [2] is a conditional variational autoencoder. Consider the agents in the scene as a scene graph, where each agent is a node in the graph. dNRI encodes the edges of a fully connected directed scene graph (potential interactions between agents) into latent variables. Then, the decoder, being conditioned on the latent variables,

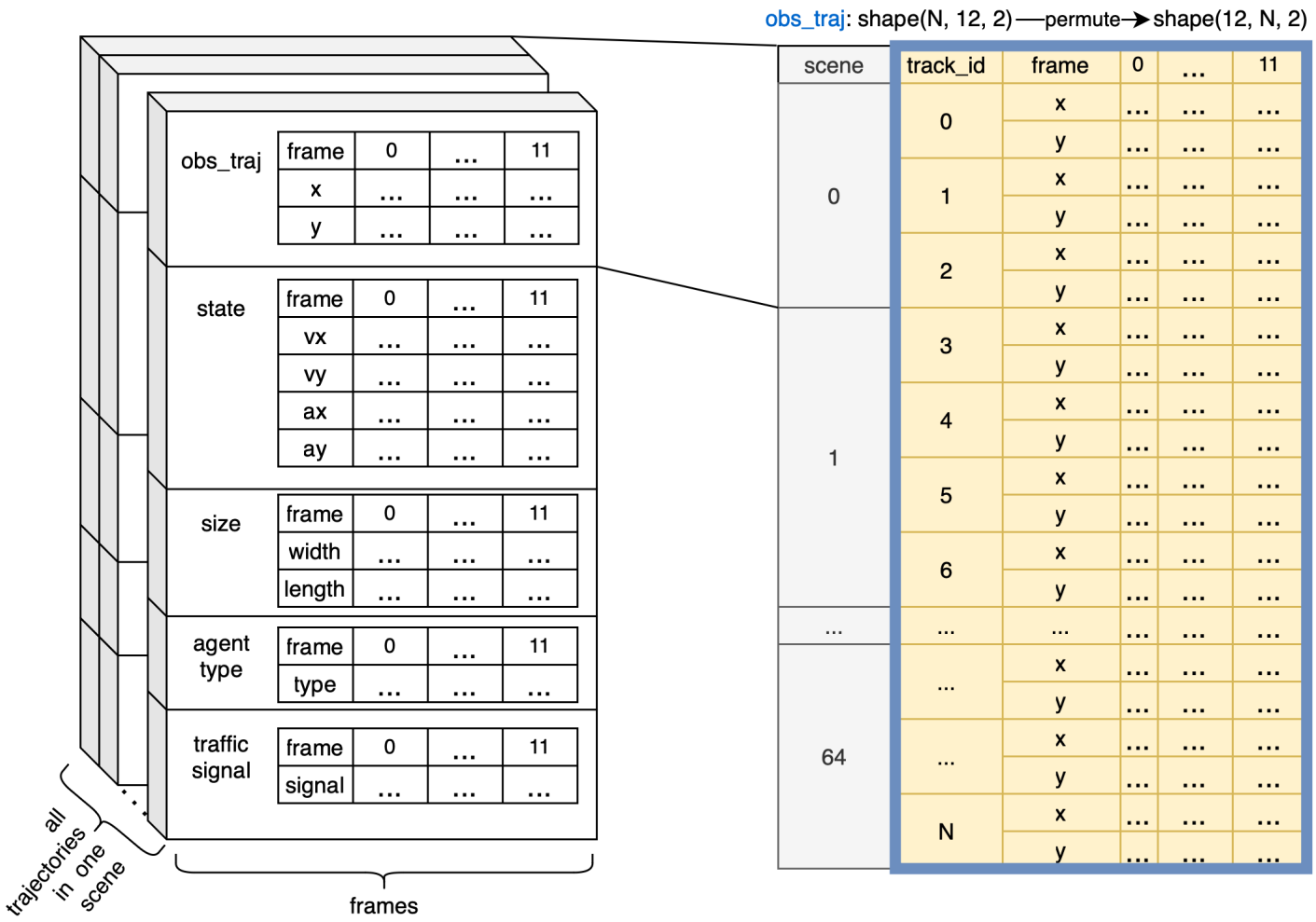



Figure 2: KI-GAN Data Input. Observation trajectories from different scenes are stacked together to form an input tensor into the model. N is the number of trajectories in one batch. For illustration purposes, we selected the number of observation frames to be 12. And for programming convenience, the tensor dimensions have been permuted.

recurrently predicts the next point in the trajectory $\hat{\mathbf{x}}^{(t+1)}$.

**Encoder:** Given the embedding of previous observations $\mathbf{x}_{\text{emb}}^{(1:T)}$ from time $t = 1, \ldots, T$, the encoder predicts categorical distributions of the types of interactions (latent variable $\mathbf{z}^{(T)}$) between agents in the scene at the current time step $t = T$.

The architecture can be seen in Figure (3). First, edge features at time $t$ are computed over a graph neural network (GNN). Then the encoder predicts 2 distributions for the latent variable $\mathbf{z}^{(t)}$: 1) the prior $p(\mathbf{z}^{(t)}|\mathbf{x}_{\text{emb}}^{(1:t)})$ to using a forward LSTM, and 2) the interaction posterior $q(\mathbf{z}^{(t)}|\mathbf{x}_{\text{emb}})$ using a bidirectional LSTM.

During training, we feed the embedding of the ground truth trajectories, both past and future, into the encoder. And the interaction posterior $q_\phi(\mathbf{z}^{(t)}|\mathbf{x}_{\text{emb}})$ into the decoder. The KL-divergence between $p_\phi$ and $q_\phi$ is minimised in the encoder loss in Equation (7) such that the prior $p_\phi$ learns from $q_\phi$.

$$\mathscr{L}_{\text{enc}} = \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\phi(\mathbf{z}|\mathbf{x}_{\text{emb}})) \tag{7}$$

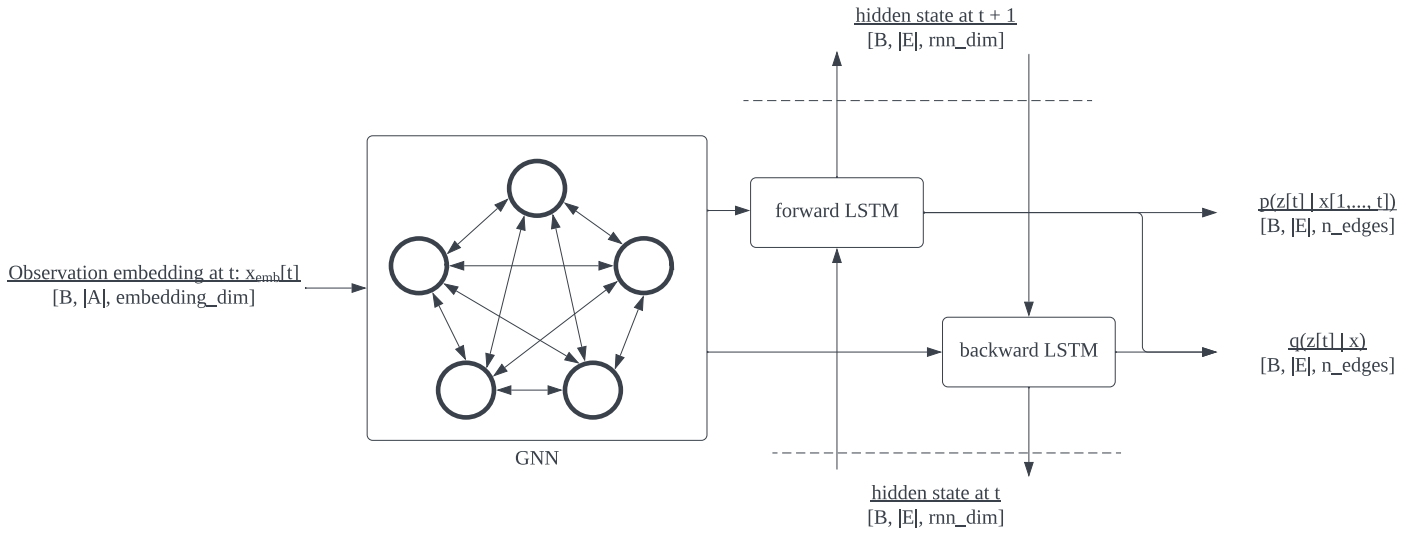



Figure 3: dNRI encoder architecture
Where $B$ denotes batch size, $|A|$ denotes no. of agents, $|E| = |A|(|A|-1)$ denotes no. of potential interactions, `embedding_dim` denotes embedding dimension, `n_edges` denotes no. of interaction types, and `rnn_dim` denotes the dimension of the hidden state.

During inference, we only provide the past true trajectories. And we feed the learned prior $p_\phi(\mathbf{z}^{(t)}|\mathbf{x}_{\text{emb}}^{(1:t)})$ into the decoder. We then auto-regressively feed in the embedding of the decoder prediction at $t$ to predict $p_\phi(\mathbf{z}^{(t+1)}|\mathbf{x}_{\text{emb}}^{(1:t)}, \hat{\mathbf{x}}_{\text{emb}}^{(t+1)})$, the process repeats until we reach the prediction horizon.

**Edge sampling:** During training, we apply gumbel-softmax sampling to the predicted logits of the distribution of $\mathbf{z}$. The sampling process is described in Equation (8).

$$\Pr(\mathbf{z}_{i,j}) = \text{softmax}\left((\mathbf{z}_{i,j} + \mathbf{g})/\tau\right) \tag{8}$$

Where $\mathbf{z}_{i,j}$ here denotes the logits of the distribution of the interaction between agent $i$ and agent $j$. $\mathbf{g} \sim \text{Gumbel}(0,1)$ is the sampled random variable, and $\tau$ is the sampling temperature. In our experiments, we used $\tau = 0.5$.

During inference, we use hard-softmax in Equation (9) to select the most probable interaction type.

$$\Pr(\mathbf{z}_{i,j}) = \mathbb{1}\left[\mathbf{z}_{i,j} = \max \mathbf{z}_{i,j}\right] \tag{9}$$

Where $\mathbb{1}[\cdot]$ denotes the indicator function.

**Decoder:** Given the embedding of previous observations $\mathbf{x}_{\text{emb}}^{(1:t)}$, and the latent distributions $\Pr(\mathbf{z}^{(1:t)})$, the decoder predicts the distribution of the next observation $p_\theta(\mathbf{x}^{(t+1)}|\mathbf{x}_{\text{emb}}^{(1:t)}, \mathbf{z}^{(1:t)})$. A gated recurrent unit (GRU) is used in the decoder to predict non-Markovian trajectories [16, 2].

The architecture can be seen in Figure (4). The hidden state of the GRU is first updated by the GNN as in Equation (10).

$$\hat{\mathbf{h}}_i^{(t)} = \sum_{j \neq i} \sum_{e=1}^{n_{\text{edges}}} \Pr(\mathbf{z}_{i,j}^{(t)} = e) f_e(\mathbf{h}_i^{(t)} \| \mathbf{h}_j^{(t)}) \tag{10}$$

Where $\mathbf{h}_i^{(t)}$ and $\hat{\mathbf{h}}_i^{(t)}$ denotes the hidden state and the hidden state after GNN of agent $i$ and time $t$. $e$ denote the interaction type latent variable, $\cdot\|\cdot$ denotes the vector concatenation operator. $f_e$ denotes the interaction function

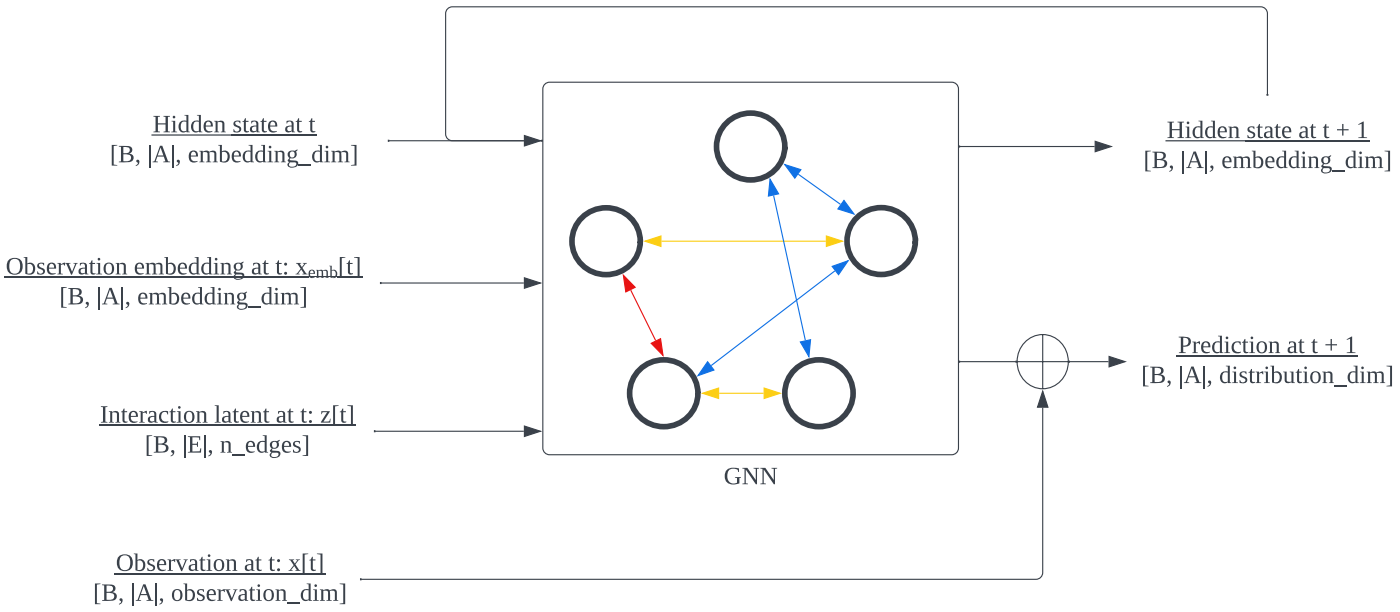



Figure 4: dNRI decoder architecture
Where `observation_dim` $= 12$ denotes the dimension of the observation vector, see Figure (13) for details.
And `distribution_dim` $= 29$ denotes the dimension of the distribution parameters

for edge type $e$, we used multi-layer perceptron (MLP) in our model. $e = 1$ is interpreted as no interaction, hence, $f_1 = \mathbf{0}$.

Finally, $\mathbf{x}_{\text{emb}}^{(t)}$ and $\hat{\mathbf{h}}^{(t)}$ is fed into the GRU to predict the distribution of change in the next time step $\Pr(\Delta\mathbf{x}^{(t)}|\mathbf{x}^{(1:t)})$ and the next hidden state $\mathbf{h}^{t+1}$. The distribution $\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(1:t)}, \mathbf{z}^{(1:t)})$ is obtained by adding $\mathbf{x}^{(t)}$ to the mean parameters of $\Pr(\Delta\mathbf{x}^{(t)}|\mathbf{x}^{(1:t)})$. Details for the parameterisation used can be found in Appendix B.

The hidden state is initialised by feeding the past trajectories $\mathbf{x}_{\text{emb}}^{(1:t)}$, and the latent distribution $\Pr(\mathbf{z}^{(1:t)})$. This initialisation process is called "burn-in" in NRI ([16]) and dNRI [2]. Then the embedding of the most probable prediction $\hat{\mathbf{x}}_{\text{emb}}^{(t+1)}$ and $\mathbf{z}^{(t+1)}$ is fed into the decoder to predict $\Pr(\mathbf{x}^{(t+2)}|\mathbf{x}_{\text{emb}}^{(1:t)}, \hat{\mathbf{x}}_{\text{emb}}^{(t+1)}, \mathbf{z}^{(1:t+1)})$. This process is repeated until the prediction horizon is reached.

During training, we minimise the reconstruction loss (negative log-likelihood) in Equation (11).

$$\mathscr{L}_{\text{dec}} = -\frac{1}{|A|T}\sum_{i\in A}\sum_{t=1}^{T}\log p_\theta(\mathbf{x}_i^{(t)}|\mathbf{z}^{(t)}) \tag{11}$$

Where $A$ is the set of agents in the batch and $T$ here is the prediction length.

**Signal prediction:** To inject traffic signal information into dNRI, we introduce a signal model to predict future signals $\mathbf{s}^{(t_{\text{obs}}+1:T)}$ given past signals $\mathbf{s}^{(1:t_{\text{obs}})}$, where $t_{\text{obs}}$ is the length of the observed trajectories, and $T$ here is the total sequence length (including both the past and future segment).

We used a seq2seq [23] hidden Markov model. First, past signals $\mathbf{s}^{(1:t_{\text{obs}})}$ are iteratively fed into the LSTM encoder to initialize the hidden state. Then we use the final hidden state as the initial hidden state of the LSTM decoder. Where the start-of-sequence token is iteratively fed into the decoder. The decoder outputs the logits for the distribution of the signal state at the next time step. The architecture can be seen in Figure (5).

The signals is then injected into the observation embedding for dNRI encoder and decoder at each time step through the mixing operation in Equation (12).

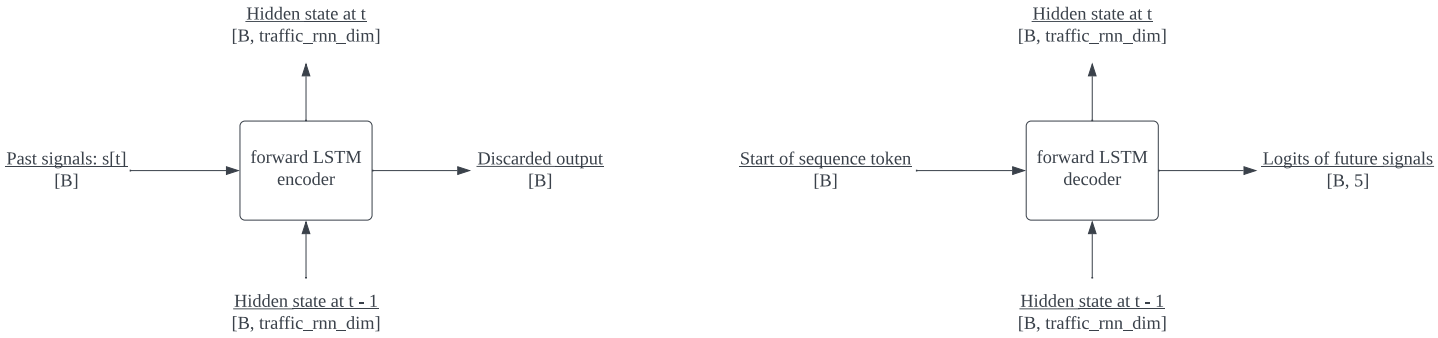



Figure 5: dNRI signal model architecture

$$\mathbf{x}_{\text{emb}}^{(t)} = \text{MLP}(\mathbf{x}^{(t)} \| \mathbf{s}_{\text{emb}}^{(t)}) \tag{12}$$

Where $\mathbf{s}_{\text{emb}}^{(t)}$ is the embedding of $\mathbf{s}^{(t)}$ used in the signal model.

During training, embedding of the ground truth signals are used to compute $x_{\text{emb}}$. While the signal model is trained by minimizing the cross-entropy loss between the predictions and the ground truth in Equation (13).

$$\mathscr{L}_{\text{signal}} = \frac{1}{T} \sum_{t=1}^{T} H(\hat{\mathbf{s}^{(t)}}, \mathbf{s}^{(t)}) \tag{13}$$

Where $T$ here is the length of the future segment.

During inference, predictions made by the signal model will be used to compute $\mathbf{x}_{\text{emb}}$.

**Agent feature conditioning:** Agent features (type and size) are incorporated into dNRI as the initial hidden state of the encoder and the decoder.

$$\text{encoder hidden state, decoder hidden state} = \text{MLP}(\text{agent size} \| \text{agent type embedding})$$

**Loss function:** Combining Equation (7, 11, 13), we derive the overall loss in Equation (14).

$$\mathscr{L} = \lambda_{\text{enc}} \cdot \mathscr{L}_{\text{enc}} + \lambda_{\text{dec}} \cdot \mathscr{L}_{\text{dec}} + \lambda_{\text{signal}} \cdot \mathscr{L}_{\text{signal}} \tag{14}$$

In our experiments, we used $\lambda_{\text{enc}} = \lambda_{\text{dec}} = \lambda_{\text{signal}} = 1$.

### Data Input Format

As illustrated in Figure 13, the trajectory tensor is used for observation embedding. The initial hidden state is generated by agent types embedding and observation embedding. The traffic signals are fed into a Sequence-to-Sequence model to predict future signals, which are then combined with observation embedding to create a mixed representation.

### Hyperparameters and Other Details

We used `n_edges` $= 4$. Due to constraint in GPU memory, we used 64 for all hidden state dimension (`embedding_dim` and `rnn_dim`) in dNRI, and 32 for the hidden state dimension in the signal model. Note that the number of trainable parameters is significantly less than KI-GAN [1] and dG-VAE [4]

The model is trained using the Adam optimiser with default learning rate (`lr` = 0.001) over 20 epochs with batch size 64.

Unlike KI-GAN, dNRI accepts incomplete trajectories (i.e. agents not presenting in all frames). We train the model including incomplete trajectories, and evaluate the model only with complete trajectories.

# 4 Experimental Setup

## Experimental Setup Summary

We first outline our considerations for the various datasets containing signalized intersection data, before settling on the Signalized INtersection Dataset (”SIND”), proposed by Xu et al.[3] Following this, we outline our selected evaluation criteria for the prediction of a trajectory, consisting of the Average Displacement Error (”ADE”) and Final Displacement Error (”FDE”), which is consistent with the approach taken in the literature; and appears to be a reasonable approach to understanding the accuracy of trajectory prediction. [1], [4] Finally, we outline the steps and associated challenges in processing a time-series scenario-based dataset in order to train, validate, and test the models.

## Dataset Selection

In order to set up the trajectory prediction experiment, we considered a number of aerial-captured intersection datasets, which had already been pre-processed from images into numerical data.

- We first considered the inD dataset, as proposed by Bock et al. [24]. This dataset consisted of approximately 10 hours of footage and 12,000 traffic interactions from an intersection in Aachen, Germany. However, this dataset did not include traffic signals or motorcycles. As such, we did not consider it appropriate for our purposes of understanding the vehicle trajectory prediction problem from the lens of vulnerable traffic participants, and understanding the importance of traffic signals.
- We also considered the INTERACTION dataset, as proposed by Zhan et al. [25]. The INTERACTION dataset consisted of a variety of intersections in various locations, with approximately 16 hours of footage of 40,000 traffic interactions. However, the INTERACTION dataset also lacked traffic signal data and motorcycle participants, and was therefore rejected for the same reason as the inD dataset.
- We ultimately selected the SIND Tianjin Dataset [3], as noted above. The SIND Tianjin dataset consisted of 7 hours of footage from an intersection in Tianjin, China, with 13,000 traffic participants. The primary reasons for selection were the inclusion of signalised intersection data and numerous vulnerable traffic participants. In addition, the SIND Tianjin Dataset also had a higher density of interactions over its 7-hour duration over a single intersection (approximately 1,900 per hour, as compared to 1,200 per hour in the inD dataset), and included statistics on traffic violations.

Following the selection of the SIND Tianjin Dataset and during the initial analysis phase, the authors released additional datasets for other intersections, including Chongqing, Changchun, and Xi’an. However, the original Tianjin Dataset was considered most appropriate due to it being a less geometrically complex intersection with a larger variety of participants. As such, the experiment setup focused on use of the SIND Tianjin Dataset.

## Evaluation Metrics

The ADE and FDE displacement errors, as described in KI-GAN [1], were selected to evaluate the accuracy of the model in trajectory prediction.

$$\text{ADE} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T}\sqrt{(x_i^{(t)}-\hat{x}_i^{(t)})^2+(y_i^{(t)}-\hat{y}_i^{(t)})^2} \tag{15}$$

$$\text{FDE} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i^{(T)} - \hat{x}_i^{(T)})^2 + (y_i^{(T)} - \hat{y}_i^{(T)})^2} \tag{16}$$

Where $N$ is the number of trajectories, $T$ is the length of the predicted trajectories. $x_i^{(t)}$ and $y_i^{(t)}$ denotes $x$ and $y$ coordinates of the ground truth location at time step $t$ of the $i$-th trajectory, $\hat{x}$ and $\hat{y}$ are the predicted locations.

These evaluation metrics were considered the most appropriate to evaluate a trajectory prediction path for the following reasons:

- The ADE calculates the average distance between the predicted path and the ground truth, over the predicted trajectory. From a vehicle safety perspective, the path of the vehicle matters as much as the final destination, and therefore capturing this cumulative error is important from a safety perspective. Furthermore, this metric is more useful in extended prediction scenarios.
- In contrast, the FDE calculates the distance between the predicted final point, and the final point in the ground truth. The FDE measures a distance in a single point, which is more useful for sudden stopping, turning or collision detection.

The combination of these metrics provides a more complete view of the performance of a trajectory prediction model.

## Dataset Processing

As illustrated in Figure 6, The original data is transformed and segmented, then filtered to get the result. See also Figure 6 for more details.

The conceptualisation of the SIND dataset in order to process it was one of the key challenges associated with this analysis. In particular, the SIND dataset could be represented as a number of CSV files, each of which contained positional information for a given traffic participant at a given frame. Therefore, each traffic participant could be represented by a time-series in the given captured scenario, in which the participant moved as the series progressed. Therefore, the objective of the dataset preprocessing was to enable the tracking of the singular participant, on a frame-by-frame basis, as it interacted with other elements in the scene (including other traffic participants and the contextual traffic light information).

Since agent size (length, width) data are not available for pedestrians in SinD, we used the mean chest depth and mean shoulder breadth (averaged over male and female) statistics from US Army [26] as length and width respectively.

Heading angle, yaw angle, lateral/longitudinal velocity and acceleration of pedestrian agents are calculated using Equation (17). Refer to Table (6) for symbol notations.

$$\begin{aligned} \theta_{\text{heading}} &= \text{atan2}(a_y, a_x) \\ \theta_{\text{yaw}} &= \text{atan2}(a_y, a_x) \\ v_{\text{lat}} &= 0 \\ v_{\text{lon}} &= v_x \\ a_{\text{lat}} &= 0 \\ a_{\text{lon}} &= a_x \end{aligned} \tag{17}$$

# 5 Results and Discussion

## Summary of Results - KI-GAN

A summary of the results from our KI-GAN experiments are as follows:

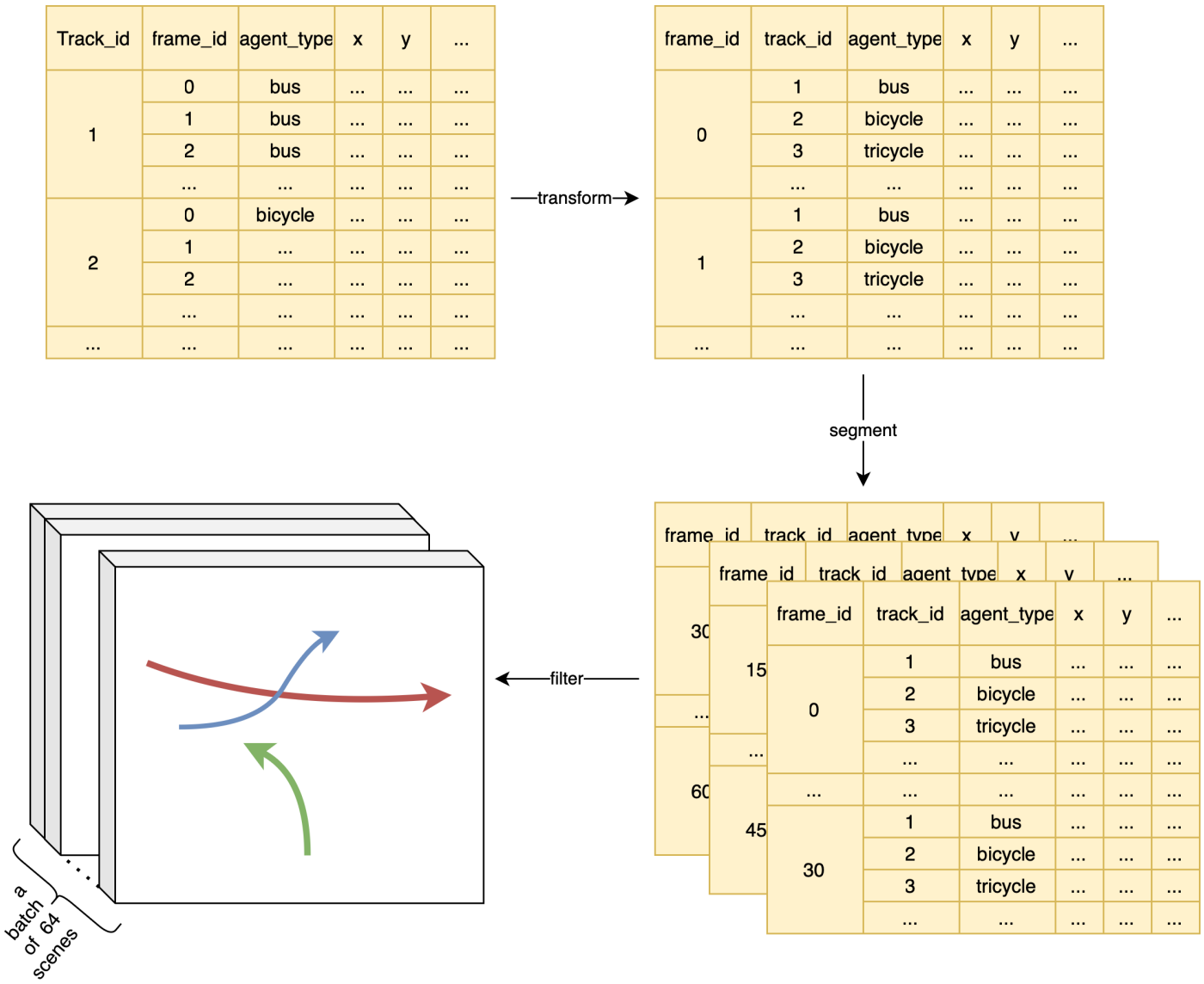



Figure 6: Pre-processing Steps. The original dataset is initially aggregated by track_id. We then transform the dataset by aggregating it by frame_id. Subsequently, the dataset is segmented into scenes by a stride of 15 (resulting in a 2 frames per second output, per the approach in the KI-GAN paper). Trajectories that do not span the entire scene from start to finish are filtered out. Additionally, scenes without any vulnerable road users are discarded. In this example, we select a batch size of 64, an observation length of 12, and a prediction length of 18 for illustration purpose.

**Vehicle Trajectory Prediction Capability**

- The results of our experiments with the KI-GAN were predominantly in line with the results from Wei et al. [1].
- There was a significant decay when experimenting with a larger prediction size, which captures the importance of constantly updating the predicted inputs.
- The KI-GAN typically performed better on straight line trajectory prediction than curves. We have attributed this to potentially insufficient noise in the generator.

**Vulnerable Traffic Participant Performance**

- We utilised a class-based analysis (focused on ADE performance) to determine the performance of vulnerable traffic participants.

- There were some minor variations due to the inclusion and exclusion of the traffic encoder on vulnerable participants.
- We note that motorcycles and bicycles class prediction suffered from poor performance. This may have been due to the higher number of traffic violations for this class, but also the nimbleness of these traffic participants.
- The pedestrian prediction is average compared to the other classes.

**Influence of Traffic Lights and Traffic Light Encoder**

- The masking of the traffic encoder appeared to improve the prediction capability within our models by a small margin, which may not be statistically significant.
- This was in contrast to the results from Wei et al. [1], in which the exclusion of the traffic encoder significantly worsened results. We suspect that this may be attributed to methodology differences (training the model with masked data versus removing the encoder in the test set on a previously trained model).

## Summary of Results - NRI

A summary of the results from our NRI experiments are as follows:

**Vehicle Trajectory Prediction Capability**

- The performance of our dNRI experiments was consistent with the result in Dax et al.[4] However, our overall results appeared to be slightly worse - this may be attributed to a lack of computational resources.
- There was a high error associated with the ”truck” class prediction.
- The overall ADE and FDE results were relatively high, and less likely to be useful practically without further tuning.

**Vulnerable Traffic Participant Performance**

- dNRI appears to struggle with bicycle and motorcycle prediction. Similar to the KI-GAN results, this may have been due to the higher number of traffic violations for this class, but also the nimbleness of these traffic participants.
- Similarly to the KI-GAN, the pedestrian prediction is average compared to the other classes.

**Influence of Traffic Lights and Traffic Light Encoder**

- In contrast to the KI-GAN, masking the traffic encoder slightly worsens the prediction capability in the dNRI experiments.
- This suggests that the dNRI might benefit from further contextual information outside physics-based information.

## KI-GAN Detailed Results

The figure 7 summarises our results applying the KI-GAN model with ADE and FDE in meters (lower is better).

A few observations become apparent based on 7 and 1:

- As we increase the prediction frames, we can see a deterioration in the results (capping out at the 12 second, or 24 frame prediction). This is to be expected, as the further we move away from the observed 12 frames (6 seconds), the more likely it is that additional factors may influence the trajectory of traffic participants. The inclusion of the 24 frame prediction (12 seconds) was not in the original KI-GAN paper [1], but was included in these experiments to further understand the decay.

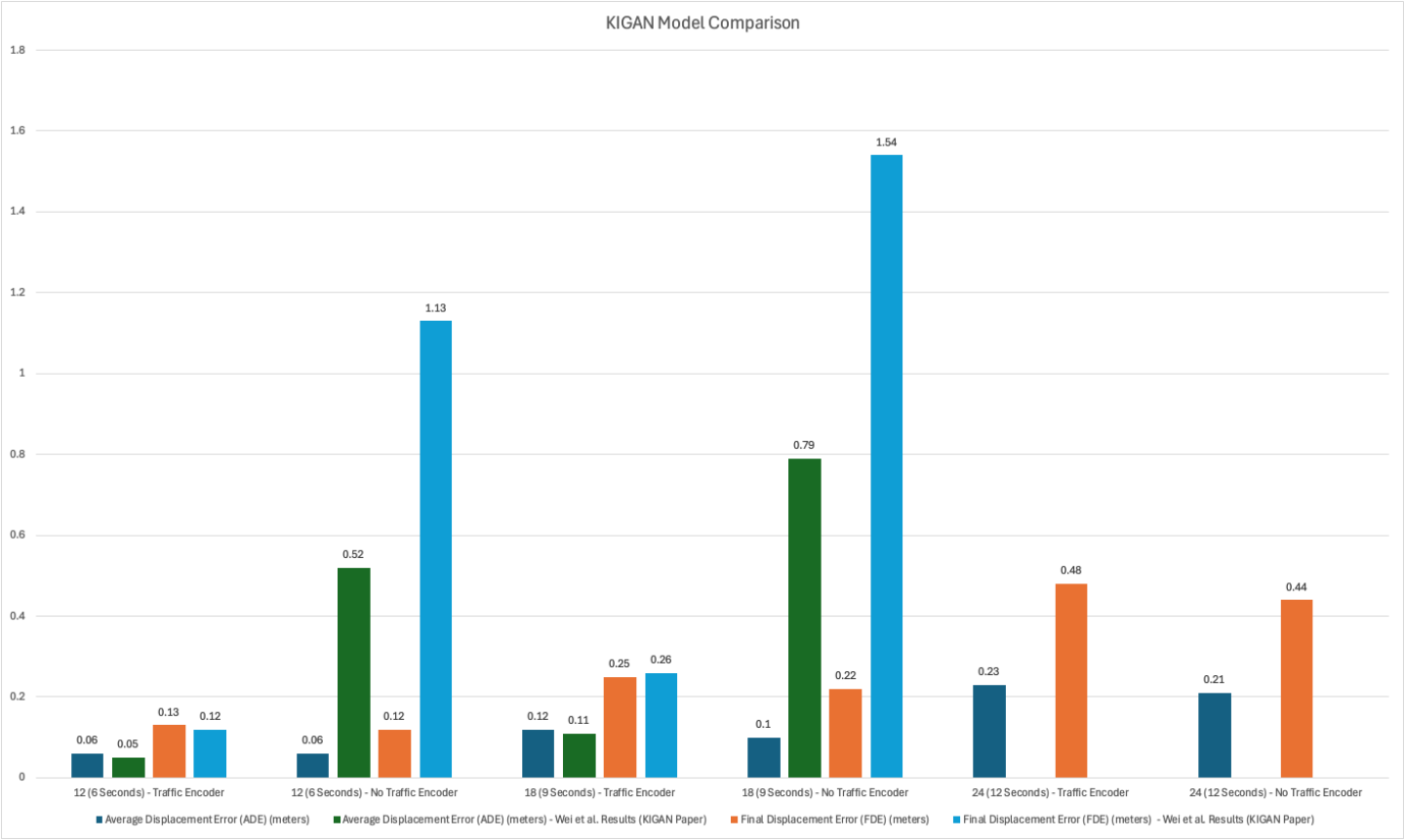



Figure 7: KI-GAN Model Results (including Wei et al. results)

| Model Origin | Number of Frames Observed | Frames Predicted | Traffic Encoder Enabled / Masked | Average Displacement Error (ADE) (meters) | Final Displacement Error (FDE) (meters) |
|---|---|---|---|---|---|
| KI-GAN Model Results | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Enabled | 0.06 | 0.13 |
| Wei et al. Results | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Enabled | 0.05 | 0.12 |
| KI-GAN Model Results | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Masked | 0.06 | 0.12 |
| Wei et al. Results | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Masked | 0.52 | 1.13 |
| KI-GAN Model Results | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Enabled | 0.12 | 0.25 |
| Wei et al. Results | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Enabled | 0.11 | 0.26 |
| KI-GAN Model Results | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Masked | 0.10 | 0.22 |
| Wei et al. Results | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Masked | 0.79 | 1.54 |
| KI-GAN Model Results | 12 Frames, 6 Seconds | 24 Frames, 12 Seconds | Enabled | 0.23 | 0.48 |
| KI-GAN Model Results | 12 Frames, 6 Seconds | 24 Frames, 12 Seconds | Masked | 0.21 | 0.44 |

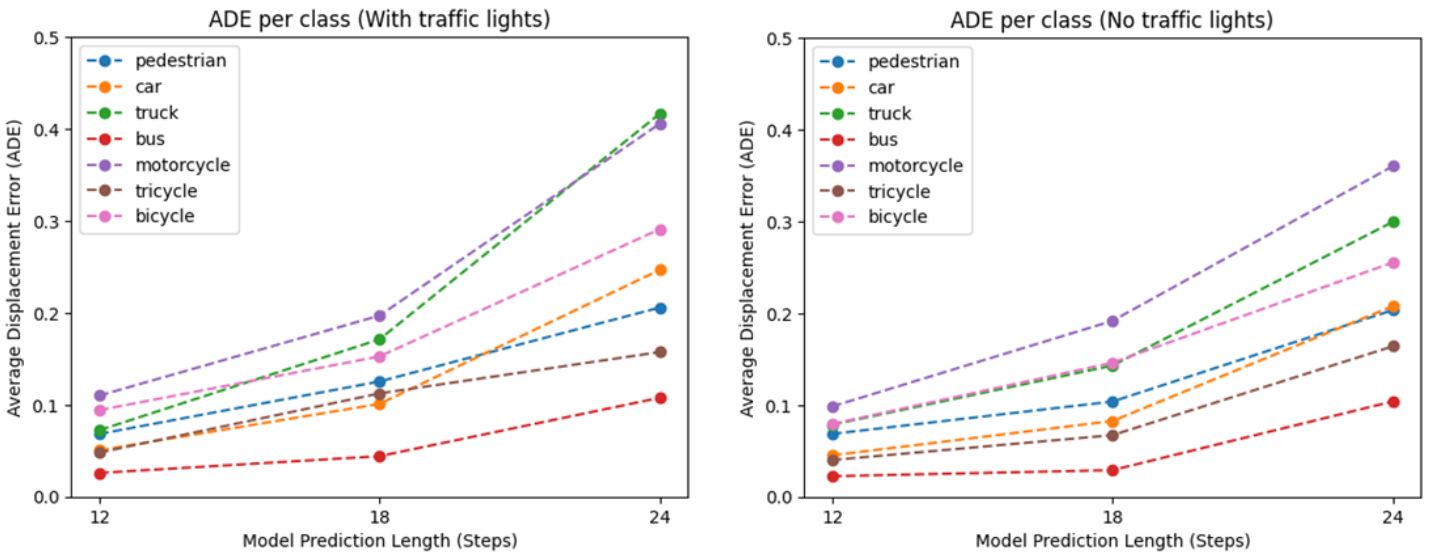
Table 1: KI-GAN Model Results (including Wei et al. results) - Tabular Format

- As noted in the experiment setup, the approach taken to the ”No Traffic Encoder” experiment was to mask the traffic encoder inputs with 0 values across the entire data set (for training, validation and testing). There does not appear to be statistically significant results between the inclusion and exclusion of the traffic encoder (with a maximum of a few centimetres of prediction difference). However, the results included in the KI-GAN paper[1] suggest that the exclusion of the traffic encoder results in significantly worsened results. Given that these models were trained on the same GPU hardware (NVIDIA RTX 3090), and the paper is silent on how these results were achieved, it is likely that this difference can be attributed to disabling the traffic encoder after training with it enabled. Therefore, while this comparison is not strictly in an identical use case, we can make the observation that the inclusion of the traffic encoder can result in a benefit, but if it is excluded and the model is trained for the same number of epochs, the trained model likely leans into the physics-based encoders to compensate.

In addition, we considered the per-class ADE results of each of the classes in 8a and in 8b, to better understand the individual class performance in the models. In particular, this allowed us to consider the performance associated with the prediction of vulnerable traffic participants.

The following observations can be made based on 8a, 8b, and 2:

- Observing the vulnerable traffic participants, it appears that the KI-GAN performs better for tricycles and




(a) KI-GAN - ADE Per Class (with traffic encoder)

(b) KI-GAN - ADE Per Class (with masked traffic encoder)

Figure 8: Comparison of KI-GAN ADE Per Class with and without traffic encoder.

| Class | ADE - 12 Step Prediction, Traffic Encoder | ADE - 12 Step Prediction, Traffic Encoder Masked | ADE - 18 Step Prediction, Traffic Encoder | ADE - 18 Step Prediction, Traffic Encoder Masked | ADE - 24 Step Prediction, Traffic Encoder | ADE - 24 Step Prediction, Traffic Encoder Masked |
|---|---|---|---|---|---|---|
| Pedestrian | 0.07 | 0.07 | 0.13 | 0.10 | 0.21 | 0.20 |
| Car | 0.05 | 0.05 | 0.10 | 0.08 | 0.25 | 0.21 |
| Truck | 0.07 | 0.08 | 0.17 | 0.14 | 0.42 | 0.30 |
| Bus | 0.03 | 0.02 | 0.04 | 0.03 | 0.11 | 0.10 |
| Motorcycle | 0.11 | 0.10 | 0.20 | 0.19 | 0.41 | 0.36 |
| Tricycle | 0.05 | 0.04 | 0.11 | 0.07 | 0.16 | 0.16 |
| Bicycle | 0.09 | 0.08 | 0.15 | 0.15 | 0.29 | 0.26 |

Table 2: KI-GAN - ADE Results Per Class

pedestrians than for other traffic participants. However, it appears that the models (both including and excluding the traffic encoder) struggle with motorcycle and bicycle classes. This may be due to the size and nimbleness of these traffic participants. This may further be exacerbated by the high frequency of traffic violations (approximately 32% of the motorcycle trajectories included in the SIND set were red-light traffic violations). The relatively small change in performance between including the traffic encoder and masking the traffic encoder may be attributed to this, with the model potentially learning to focus on physics-based prediction as a priority.

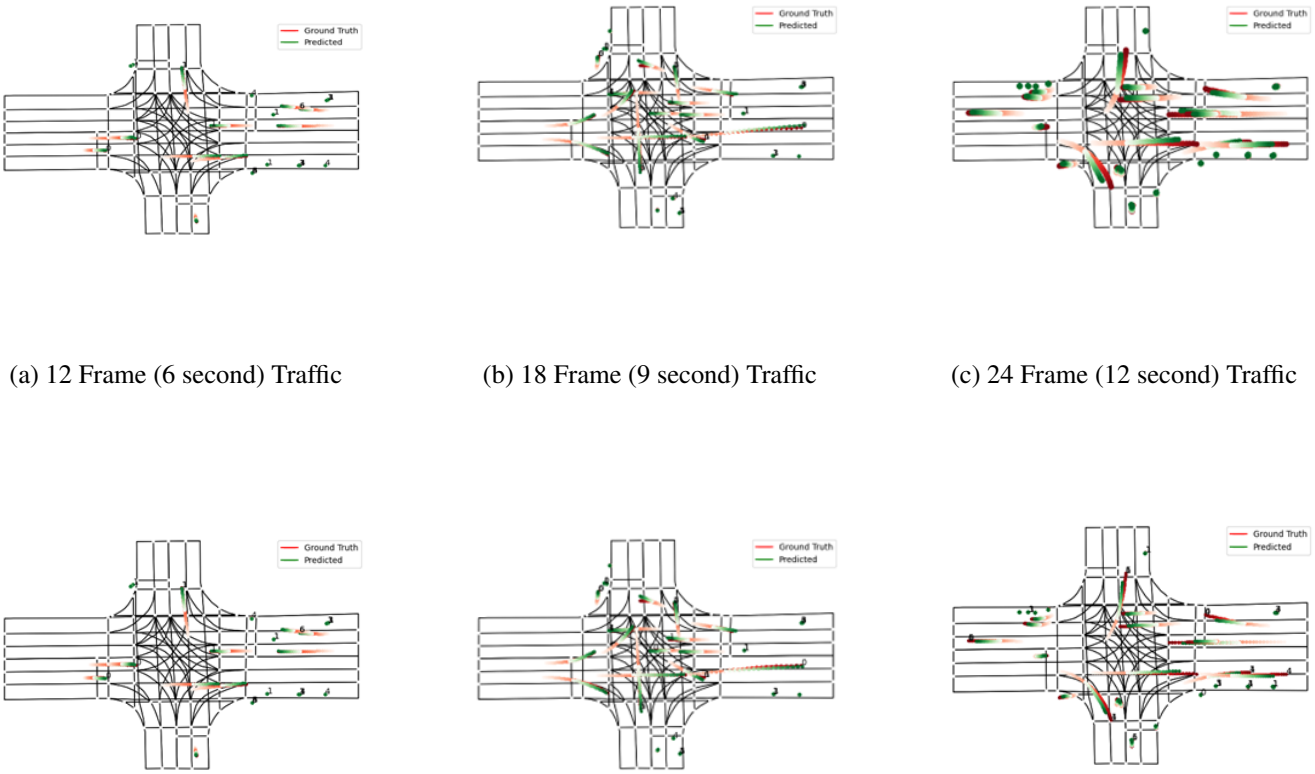
- It is unclear why tricycles performed well, but it is likely these represent three-wheeled rickshaw-style vehicles with limited turning capacity.
- It is unclear why trucks performed poorly, but this may be due to the very limited sample size (0.8%).
- The overall changes for other classes were minimal, but these results suggest that looking at what the driver is currently doing is a better prediction for future action than the context of the scene.

For completeness, we also visualised a sample of the KI-GAN trajectory paths across the intersection with the same samples over the six different KI-GAN models to perform a more qualitative analysis. The visualisations are included in 9a, 9d, 9b, 9e, 9c and 9f.

Visually, there does not appear to be an impact with regard to the inclusion and exclusion of the traffic encoder. The results and predictions appear to be broadly the same, with some minor variations. Given the models are being trained for the same number of epochs, this suggests that the masking of the traffic encoder data results in additional model effort being redirected to the physics-based side - resulting in the physics-based components compensating for the overall change. The model may need additional training to infer the traffic state internally.

In addition, we can visually see the prediction decay as the prediction frames increase. This is consistent with the graph results presented earlier.

Finally, we can see that the model prediction paths do have a curve to them, which can be attributed to the introduction of uniform noise across all models. The initial models trained lacked this, which resulted in the KI-GAN producing straight-line trajectory predictions.




(a) 12 Frame (6 second) Traffic (b) 18 Frame (9 second) Traffic (c) 24 Frame (12 second) Traffic

(d) 12 Frame (6 second) Masked Traffic (e) 18 Frame (9 second) Masked Traffic (f) 24 Frame (12 second) Masked Traffic

Figure 9: KI-GAN prediction visualization

## dNRI Detailed Results

The figure 10 summarises our results applying the dNRI model with ADE and FDE in meters (lower is better).

| Model Origin | Number of Frames Observed | Number of Frames Predicted | Traffic Encoder Enabled / Masked | Average Displacement Error (meters) | Final Displacement Error (meters) |
|---|---|---|---|---|---|
| dNRI Experiment | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Enabled | 0.78 | 1.98 |
| dNRI Experiment | 12 Frames, 6 Seconds | 12 Frames, 6 Seconds | Masked | 1.00 | 2.48 |
| dNRI Experiment | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Enabled | 0.86 | 2.09 |
| dNRI Experiment | 12 Frames, 6 Seconds | 18 Frames, 9 Seconds | Masked | 1.17 | 2.86 |
| dNRI inD Performance - Dax et al. | 20 Frames, 4 Seconds | Not Specified | N/A | 0.56 | 4.29 |

Table 3: dNRI Model Results Table
- Includes dNRI inD Performance For Comparison

Upon examining the trajectories predicted by dNRI, we observed that dNRI fails to predict trajectories of moving agents. A few examples can be found in Figure (12).

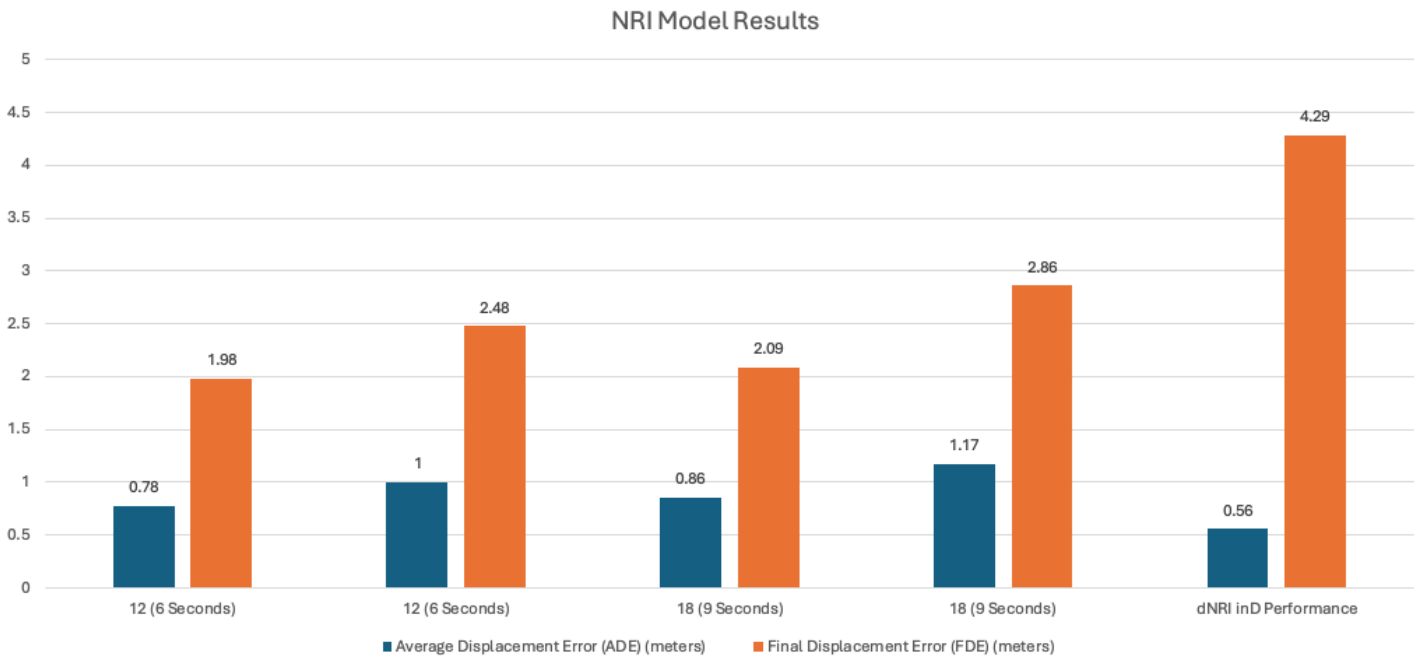



Figure 10: dNRI Model Results Graph - includes dNRI inD Performance for Comparison

### Comparison of Results

Based on the above results, we can make a few key observations:

- The KI-GAN generally performed better than the dNRI on an ADE and FDE basis. This suggests it would be more practical to utilise the KI-GAN over the dNRI in real-world use-cases.
- However, the dNRI demonstrated a net benefit from the inclusion of the traffic encoder. This suggests that a potential hybrid approach may yield additional accuracy.
- Both models suffered when assessed over their performance for vulnerable traffic participants. This suggests that both models may have weaknesses associated with nimble traffic participants, or have issues when traffic violations occur.

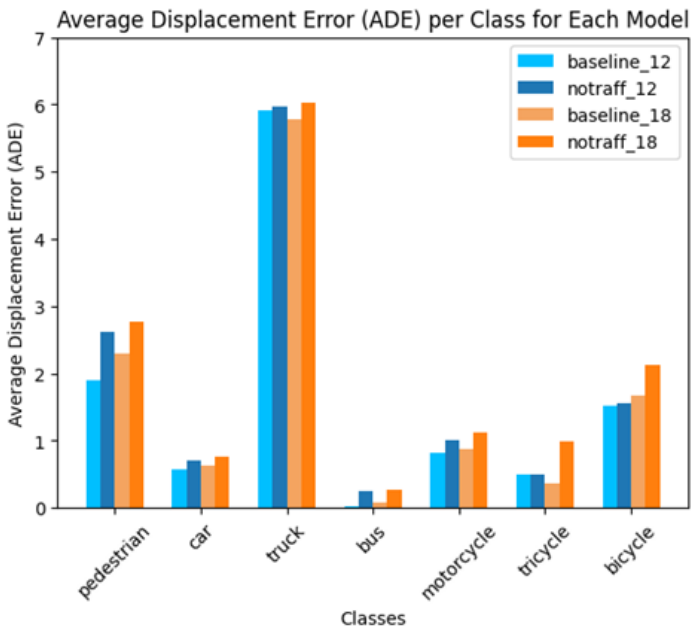
## 6 Conclusions

### Summary of Study

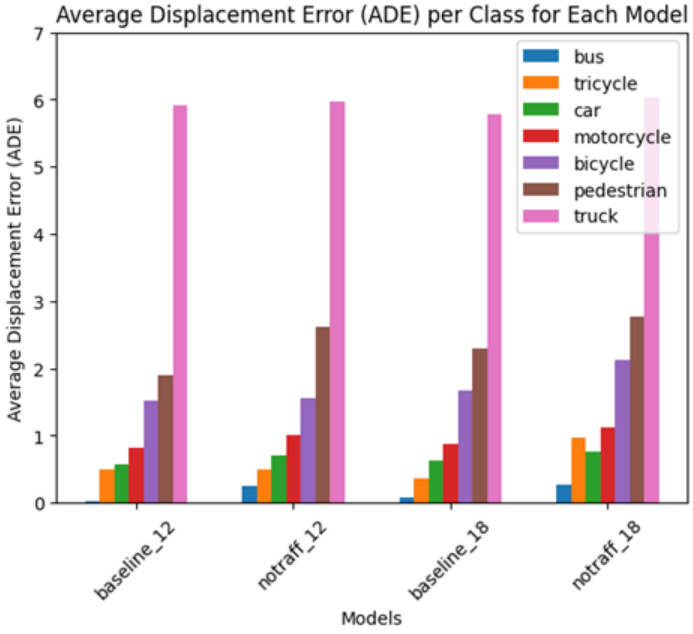
Our consideration of the KI-GAN and dNRI models allow us to draw some interesting insights into the trajectory prediction problem.

Firstly, the overall performance of KI-GAN when the traffic encoder is masked suggests that in this particular scenario, the physics-based encoders have a tendency to dominate the traffic encoder. In addition, there appears to be a minor decrease in performance when the traffic encoder is included in our experiments (which does not appear to be statistically significant). This decrease suggests that either the traffic encoder information may be detrimental to vehicle trajectory prediction, or that the KI-GAN can obtain further physics-based gains with increased training (and then may subsequently obtain additional gains through the inclusion of the traffic encoder).

The dNRI overall performed worse than the KI-GAN, but demonstrated that the addition of contextual information (i.e. the traffic encoder) can provide a benefit. In particular, this may be due to the graph-based nature and the prediction of latent variables in the dNRI structure; lending itself to take into account social interactions (and contextual interactions) better than the KI-GAN.




(a) dNRI - Comparison of Per-Class Performance




(b) dNRI - ADE Per Model

Figure 11: Comparison of dNRI Performance Metrics

Both models appeared to struggle with faster moving, more nimble traffic participants, including motorcycles and bicyclists. Therefore, additional methods should be employed for autonomous driving prediction in order to ensure the safety of these traffic participants.

## Directions for Future Improvement

### Combination of KI-GAN and dNRI to Determine Applicability of Hybrid Model

As noted above, the results of the KI-GAN and the dNRI suggest different areas of strength. More specifically:

- The results of the KI-GAN suggest that it was dominated by the physics-based encoders which captured the objects in motion, and that the inclusion of the traffic encoder information decreased the overall ADE and FDE accuracy of the predictions.
- The results of the dNRI, while weaker than those of the KI-GAN, did see an improvement in ADE and FDE accuracy with the inclusion of the traffic encoder.

Given these factors, it would be worth exploring a combined model, which was able to take the strengths of both models to improve overall accuracy. There may also be a benefit for both models if additional training and computational resources are made available (as noted in the Summary of Study section).

### Isolation of Model Classes In Training and Verification

As part of the approach taken, we considered the results of all classes simultaneously. This approach was primarily practical in nature, since we trained the models on a scene across a period of time, where traffic participants acted and interacted with one another.

However, it is worth exploring training the models by individual class to see if additional performance can be obtained (and to isolate the idiosyncrasies of each class). In particular, the results of the motorcycle and bicycle classes (and their different physical nature to other vehicles, like cars) suggests that there could be additional performance gains.

| Violation Type | Car | Truck | Bus | Motorcycle | Bicycle | Tricycle | Total | Violation % |
|---|---|---|---|---|---|---|---|---|
| Yellow Light Violation | 122 | 0 | 0 | 86 | 69 | 14 | 291 | 3% |
| Red Light Violation | 29 | 3 | 0 | 1,072 | 698 | 108 | 1,910 | 16% |
| No Violation | 4,641 | 104 | 51 | 2,202 | 2,040 | 357 | 9,395 | 81% |
| Total | 4,792 | 107 | 51 | 3,306 | 2,807 | 479 | 11,596 | |

Table 4: SIND Traffic Violation Summary

In order to isolate the different classes in training and verification, the scenario-based datasets would likely need to be pruned on a per-class basis. For example, if the motorcycle class was selected to be trained independently, the motorcycle trajectories specifically would need to be truncated while keeping all other paths as part of the ground truth. However, the downside of this approach would be the removal of the social-interaction component of the models.

Therefore, additional work determining how the isolation of model classes can be achieved while retaining social interaction may be beneficial.

**Investigate the Unusually High Error Associated with Trucks**

The truck class had an unusually high ADE/FDE error, especially compared to other vehicles. While this may be attributed to data availability (truck trajectories only made up 0.8% (107 total observations)), we would like to understand if a more balanced dataset could produce improved results, or if there are other features driving this error.

**Increase of Computational Resources Available for Future Training**

One of the key limitations we faced was the training of complex models in a reasonable time frame. With the use of an RTX 3090 with 24GB of VRAM, we were able to train models at a relatively efficient pace. However, with access to extended computational resources, we would likely be able to further refine the performance of the models trained, and consider additional scenarios (including a 24 frame / 12 second prediction for the dNRI).

**Predicting acceleration and using acceleration to improve prediction of larger time steps**

The reconstruction loss of KI-GAN only considers the positional error, a large portion of features in the dataset are not utilized. In addition, MTR [27] has shown that the inclusion of an auxiliary L1 loss for position and velocity improves the accuracy of trajectory prediction. Additional work can be done to explore whether utilizing other features of the trajectories in SinD improve the accuracy of KI-GAN.

While for dNRI, the 7 distributions (see Appendix B for details) predicted by the decoder are assumed to be independent when computing the NLL loss. This is certainly not true in reality (e.g. velocity and acceleration are obviously strongly correlated). Physical rules and constraints can be implemented like in [17] such that the model can learn to predict feasible trajectories.

**Further Exploration of Impact of Traffic Encoders**

One factor which was flagged by Xu et al. in the SIND paper [3] was numerous traffic violations at the viewed traffic intersection. The table below summarises a provided breakdown of the violations (as provided on the author's GitHub):

The impact of traffic actors ignoring the traffic signals should not be understated, as if enough actors actively ignore the results, the trained model should compensate by relying on trajectory and physics-based inputs. The reverse is also true - if the trained model believes that all participants will follow the relevant traffic signals, this means that it will fail to identify situations were a traffic participant violates the signal. In particular, we can see

evidence of these when considering the results of the motorcycle class in the KI-GAN model. There does not appear to be a significant change in the inclusion versus exclusion of the traffic encoder for this class, and this may be attributed to the significant number of traffic violations.

There are a few connected takeaways from this:

- Additional work should be considered to understand what sort of weight should be given to the traffic encoder specifically. We wish to determine whether there is some ideal weight in which the traffic encoder can add context to the model and improve prediction accuracy.
- Additional work should be considered to understand the impact of cultural driving norms. For example, the number of traffic violations as outlined here appears to be relatively high (particularly for motorcycle and bicycle classes). This result may not be the case in other countries, where different driving norms apply. In addition, the inclusion of red-light cameras may also have an impact on this behaviour (by attaching a financial penalty to the violation).

Therefore, consideration of different datasets to further explore these relationships is recommended.

### Consideration of new intersection data available with SIND

The SIND team has released data for three additional intersections and plans to release data for another eleven. The newly included intersections, characterised by their complexity and higher traffic density, will enhance the training of our models, leading to improved predictions.

### Consideration of Datasets With Occlusion

The SIND set and the other datasets we considered for the purposes of this problem were drone-based datasets. These datasets have an inherent advantage over the sensors that would be typically included in an autonomous vehicle.

At this stage, it appears impractical for an autonomous vehicle to have access to a real-time drone dataset. Therefore, we would seek to understand the impact on trajectory prediction when occlusion occurs from the perspective of an autonomous vehicle, and the associated impact with this. Such an approach would likely involve training and testing with occluded datasets based on the perspective of the vehicle to understand if this results in a significant performance degradation.

### Consideration of Incomplete Trajectories

In KI-GAN, agents not presenting all of the past and future frames are discarded. The model can only have a partial observation of the scene graph. In addition, discarding incomplete trajectories will cause the dataset to be biased towards scenes with red-lights as the agents are more likely to be stopped and be present in all frames.

### Cross Attention for Traffic Light Influence

In KI-GAN and our implementation of dNRI, the signal embedding is simply concatenated to the observation embedding. A more sophisticated approach for incorporating signal state is cross attention, which may allow the influence of signal state to be more dynamic to handle signal violation cases.

## References

[1] Chuheng Wei et al. *KI-GAN: Knowledge-Informed Generative Adversarial Networks for Enhanced Multi-Vehicle Trajectory Forecasting at Signalized Intersections*. 2024. arXiv: 2404.11181.

[2] Colin Graber and Alexander Schwing. “Dynamic Neural Relational Inference for Forecasting Trajectories”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 4383–4392. DOI: 10.1109/CVPRW50498.2020.00517.

[3] Yanchao Xu et al. *SIND: A Drone Dataset at Signalized Intersection in China*. 2022. arXiv: 2209.02297.

[4] Victoria M. Dax et al. "Disentangled Neural Relational Inference for Interpretable Motion Prediction". In: (2024). DOI: 10.1109/LRA.2023.3342554. arXiv: 2401.03599.

[5] Amr Abdelraouf et al. "Trajectory prediction for vehicle conflict identification at intersections using sequence-to-sequence recurrent neural networks". In: *arXiv preprint arXiv:2210.08009* (2022).

[6] A. Kawasaki and T. Tasaki. "Trajectory prediction of turning vehicles based on intersection geometry and observed velocities". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 511–516.

[7] Qianxia Cao et al. "Real-time vehicle trajectory prediction for traffic conflict detection at unsignalized intersections". In: *Journal of Advanced Transportation* (2021). DOI: 10.1155/2021/8453726.

[8] Agrim Gupta et al. "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2255–2264. DOI: 10.1109/CVPR.2018.00240.

[9] Debaditya Roy et al. "Vehicle trajectory prediction at intersections using interaction based generative adversarial networks". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 2318–2323.

[10] Amir Sadeghian et al. "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints". In: *SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints*. June 2019, pp. 1349–1358. DOI: 10.1109/CVPR.2019.00144.

[11] Vineet Kosaraju et al. *Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks*. July 2019.

[12] Geunseob Oh and Huei Peng. "Impact of traffic lights on trajectory forecasting of human-driven vehicles near signalized intersections". In: *arXiv preprint arXiv:1906.00486* (2019).

[13] Yuzhen Zhang et al. "D2-TPred: Discontinuous Dependency for Trajectory Prediction under Traffic Lights". In: *European Conference on Computer Vision*. Springer, 2022, pp. 522–539.

[14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013). URL: https://api.semanticscholar.org/CorpusID:216078090.

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[16] Thomas Kipf et al. *Neural Relational Inference for Interacting Systems*. 2018. arXiv: 1802.04687.

[17] Tim Salzmann et al. "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data". In: *European Conference on Computer Vision (ECCV)*. 2020. URL: https://arxiv.org/abs/2001.03093.

[18] Chen Tang et al. *Grounded Relational Inference: Domain Knowledge Driven Explainable Autonomous Driving*. Feb. 2021.

[19] Victoria M. Dax et al. "Disentangled Neural Relational Inference for Interpretable Motion Prediction". In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1452–1459. DOI: 10.1109/lra.2023.3342554. URL: https://arxiv.org/abs/2401.03599.

[20] Shaoshuai Shi et al. *MTR++: Multi-Agent Motion Prediction with Symmetric Scene Modeling and Guided Intention Querying*. 2023. arXiv: 2306.17770.

[21] Tianpei Gu et al. *Stochastic Trajectory Prediction via Motion Indeterminacy Diffusion*. 2022. arXiv: 2203.13777 [cs.CV].

[22] Chiyu Max Jiang et al. *MotionDiffuser: Controllable Multi-Agent Motion Prediction using Diffusion*. 2023. arXiv: 2306.03083 [cs.RO]. URL: https://arxiv.org/abs/2306.03083.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[24] Julian Bock et al. "The inD Dataset: A drone dataset of naturalistic road user trajectories at German intersections". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1929–1934.

[25] Wei Zhan et al. *INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps*. 2019. arXiv: 1910.03088 [cs.RO].

[26] Claire C. Gordon et al. "2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics". In: 2014. URL: https://api.semanticscholar.org/CorpusID:107330720.

[27] Shaoshuai Shi et al. *Motion Transformer with Global Intention Localization and Local Movement Refinement*. 2022. eprint: arXiv:2209.13508.

# Appendix A: Parameterisation of dNRI Decoder

In the decoder of dNRI, we predict the change of each feature at the next timestamp (denoted as $\Delta(\cdot)$).

For position, velocity and acceleration features, we predict 5 bivariate normal distributions described in Equation (18), listed in Table (5).

$$N_2\left(\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}, \begin{pmatrix}\sigma_{11}^2 & \sigma_{12}^2\\ \sigma_{12}^2 & \sigma_{22}^2\end{pmatrix}\right) \tag{18}$$

| Distribution type | Features |
|---|---|
| Bivariate normal | $\Delta x$, $\Delta y$ |
| Bivariate normal | $\Delta v_x$, $\Delta v_y$ |
| Bivariate normal | $\Delta a_x$, $\Delta a_y$ |
| Bivariate normal | $\Delta v_{\text{lat}}$, $\Delta v_{\text{long}}$ |
| Bivariate normal | $\Delta a_{\text{lat}}$, $\Delta a_{\text{long}}$ |
| von Mises | $\Delta\theta_{\text{head}}$ |
| von Miess | $\Delta\theta_{\text{yaw}}$ |

Table 5: Distribution predicted by dNRI decoder

| Symbol | Feature |
|---|---|
| $x$ | coordinate on x-axis |
| $y$ | coordinate on y-axis |
| $v_x$ | velocity on x-axis |
| $v_y$ | velocity on y-axis |
| $a_x$ | acceleration on x-axis |
| $a_y$ | acceleration on y-axis |
| $v_{\text{lat}}$ | lateral velocity |
| $v_{\text{long}}$ | longitudinal velocity |
| $a_{\text{lat}}$ | lateral acceleration |
| $a_{\text{long}}$ | longitudinal acceleration |
| $\theta_{\text{head}}$ | heading angle |
| $\theta_{\text{yaw}}$ | yaw angle |

Table 6: Feature symbol notations

For bivariate normal distributions, we used the parameterisation in Trajectron++ [17] to predict elements in the Cholesky decomposition of the covariance matrix. We used a fully connected layer to predict the distribution parameters $\begin{bmatrix}\mu_1 & \mu_2 & \log\sigma_{11} & \log\sigma_{22} & \operatorname{arctanh}(\rho_{12})\end{bmatrix}^\top$, where $\rho_{12}$ is the correlation coefficient of feature 1 and feature 2. The Cholesky decomposed covariance matrix is then computed as in Equation (19).

$$\begin{bmatrix}\sigma_{11}^2 & 0\\ \rho_{12}\sigma_{22} & \sigma_{22}\sqrt{1-\rho_{12}^2}\end{bmatrix} \tag{19}$$

For angular features, we predict 2 von Mises distributions, $\text{VonMises}(\mu,\kappa)$, listed in Table (5). We predict 3 parameters using a MLP: $\begin{bmatrix}\mu_a & \mu_b & \log\kappa\end{bmatrix}^\top$. For the mean $\mu$, we normalise $\begin{bmatrix}\mu_a & \mu_b\end{bmatrix}$ into a unit vector then compute the angle it forms with the x-axis using the $\text{atan2}(\cdot,\cdot)$ function.

In our implementation, we used `MultivariateNormal` (multivariate normal distribution) and `VonMises` (von Mises distribution) in `torch.distributions` to compute the negative log-likelihood (NLL) loss for the decoder. We assumed all the predicted distributions are independent when computing the loss.

# Appendix B: dNRI Visualisation

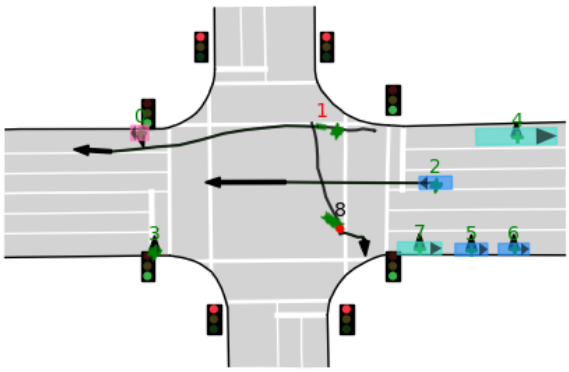
In Figure (12), ground truth (past, future) are drawn as black lines, predictions are drawn as green dots. Agents are drawn using the rule in Table (7).

| Agent Type | Shape |
|---|---|
| Car | dodgerblue rectangle |
| Motorcycle | orange rectangle |
| Bus | turquoise rectangle |
| Truck | yellow rectangle |
| Tricycle | hotpink rectangle |
| Pedestrian | red circle |

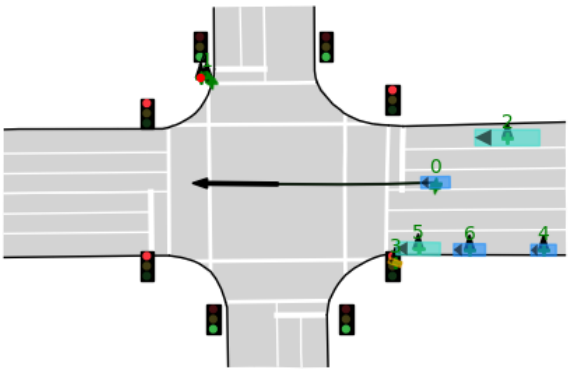
Table 7: dNRI agent visualisation rule

Number above agents are their IDs. Colour indicates their signal violation behaviour: 1) red indicates signal red-light running; 2) yellow indicates yellow-light running; 3) green indicates no violation; and 4) black indicates unknown (due to signal violation flag not labelled for pedestrians in the dataset).

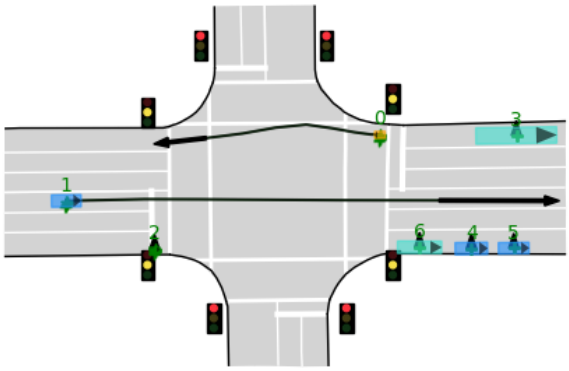
Code used for trajectory visualisation can be found in `NRI/experiments/visualization.py`.



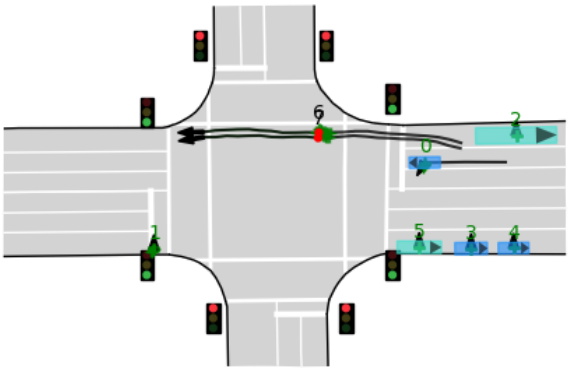
(a) Car (id: 2), bike (id: 1) and pedestrian (id: 8)



(b) Car (id: 0), no signal violation



(c) Car (id: 1) and motorcycle (id: 0)



(d) Car (id: 0) and pedestrians (id: 6, 7)

Figure 12: dNRI prediction visualisation
dNRI trained on 12 Frame (6 second) prediction with traffic

# Appendix C: dNRI Data Input

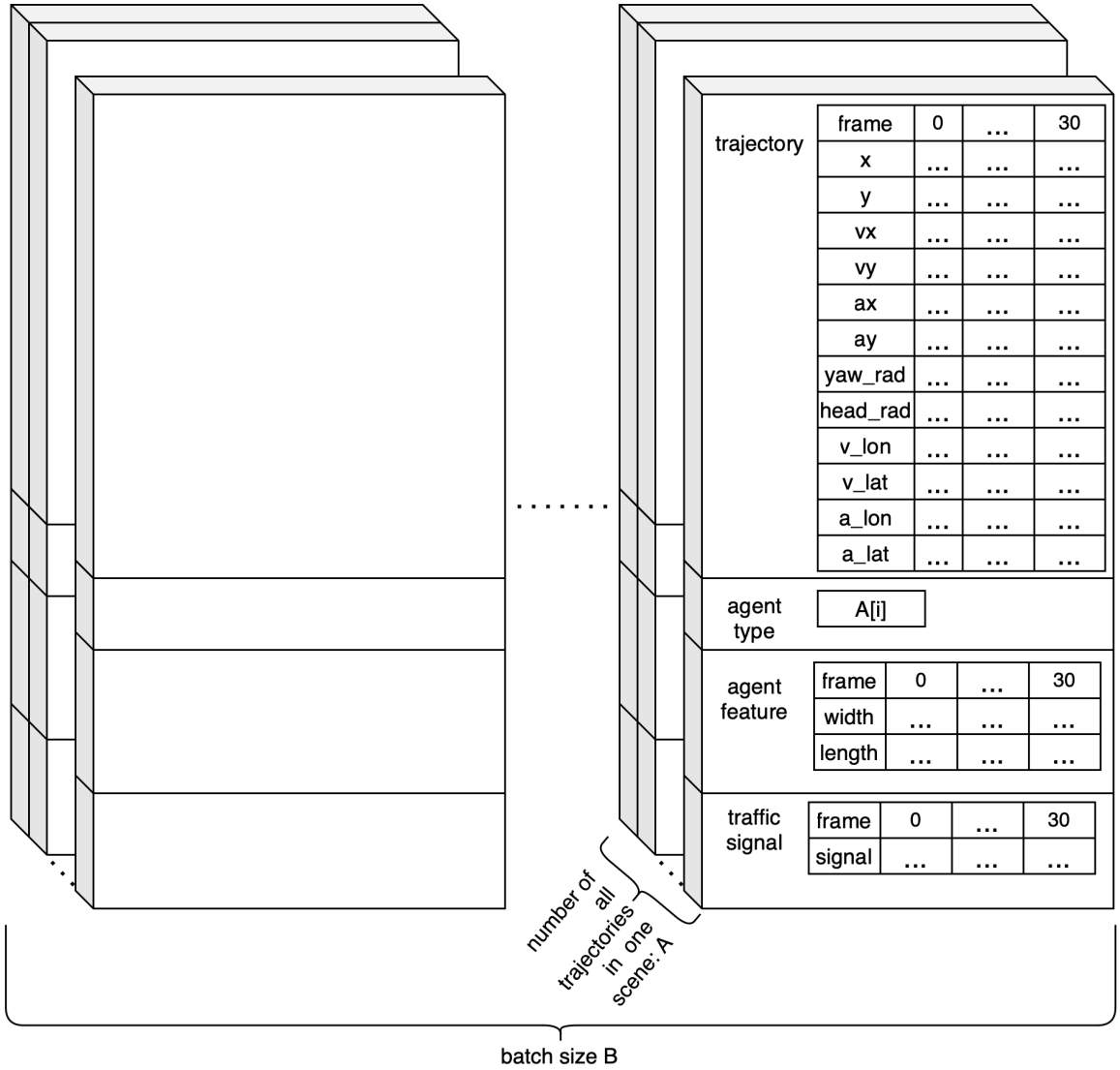



Figure 13: dNRI Data Input. The trajectory tensor is of shape(B, A, t, dim). In our settings, B(Batch size)=64, t(frames)=30, A(number of agents in the scene) dim(trajectory feature dimension)=12. The agent type tensor is of shape(B, A). The agent feature tensor is of shape(B, A, t, 2), representing agent width and length. And traffic signal is of shape(B, t). Since some padding trajectories exist, a trajectory mask tensor of shape(B, A, t) is also provided to indicate whether a trajectory is valid or not.