ASTRA: A Scene-aware TRAnsformer-based model for trajectory prediction

Anonymous authors
Paper under double-blind review

Abstract

We present ASTRA (**A S**cene-aware **TRA**nsformer-based model for trajectory prediction), a light-weight pedestrian trajectory forecasting model that integrates the scene context, spatial dynamics, social inter-agent interactions and temporal progressions for precise forecasting. We utilised a U-Net-based feature extractor, via its latent vector representation, to capture scene representations and a graph-aware transformer encoder for capturing social interactions. These components are integrated to learn an agent-scene aware embedding, enabling the model to learn spatial dynamics and forecast the future trajectory of pedestrians. The model is designed to produce both deterministic and stochastic outcomes, with the stochastic predictions being generated by incorporating a Conditional Variational Auto-Encoder (CVAE). ASTRA also proposes a simple yet effective weighted penalty loss function, which helps to yield predictions that outperform a wide array of state-of-the-art deterministic and generative models. ASTRA demonstrates an average improvement of 27%/10% in deterministic/stochastic settings on the ETH-UCY dataset, and 26% improvement on the PIE dataset, respectively, along with seven times fewer parameters than the existing state-of-the-art model (see Figure 1). Additionally, the model's versatility allows it to generalize across different perspectives, such as Bird's Eye View (BEV) and Ego-Vehicle View (EVV).

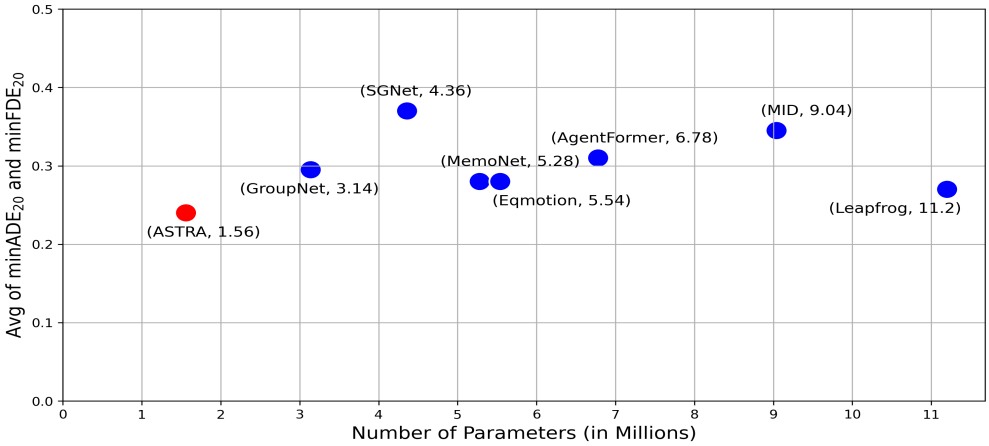
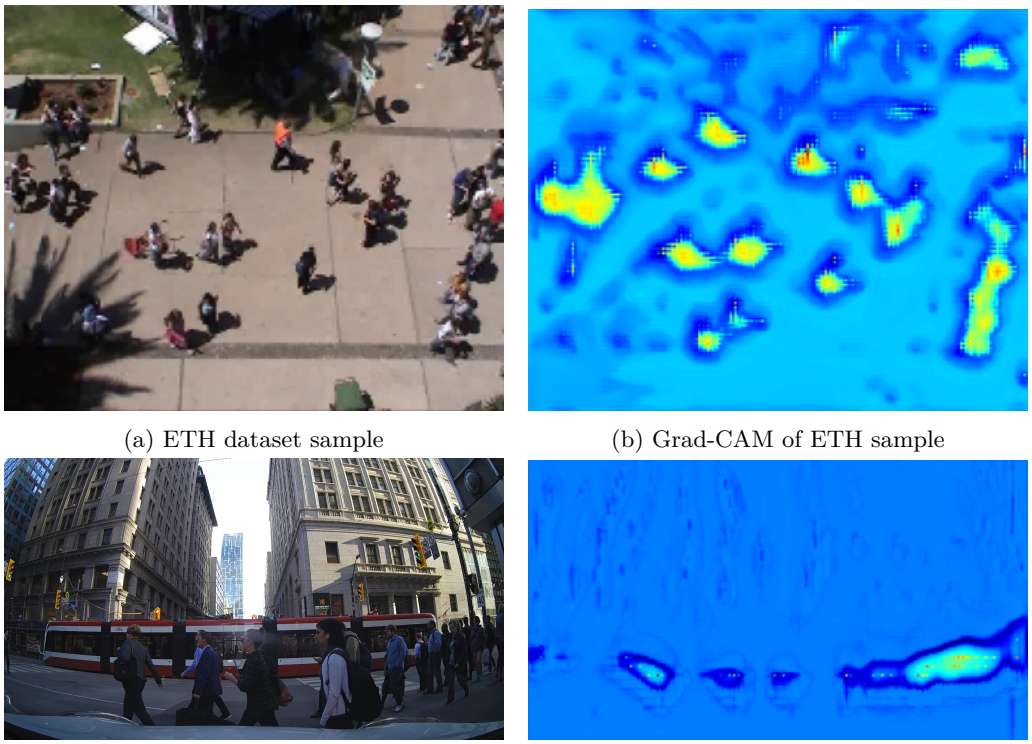


Figure 1: Comparison of average (minADE_{20}/minFDE_{20}) against the number of parameters for various models on the ETH-UCY dataset. Each point represents a different model, with the model name and number of parameters in millions indicated. Our model, ASTRA, achieves state-of-the-art results with the least number of parameters, demonstrating its efficiency and effectiveness in pedestrian trajectory forecasting.

1 Introduction

The pursuit of forecasting human trajectories is central, acting as a cornerstone for devising secure and interactive autonomous systems across various sectors. This endeavour is crucial in a wide array of applica-

tions, encompassing autonomous vehicles, drones, surveillance, human-robot interaction, and social robotics. Furthermore, it is crucial for predictive models to strike a balance between accuracy, dependability, and computational efficiency, given the imperative for these models to function on in-vehicle processing units with limited capabilities. The challenge of trajectory prediction involves estimating the future locations of agents within a scene, given its past trajectory. This estimation task can be tackled either through Bird's Eye View (BEV) (Figure 2a) or Ego-Vehicle View (EVV) (Figure 2c) perspectives. This demands a comprehensive understanding of the scene, in addition to spatial, temporal, and social aspects that govern human movement and interaction.



(a) ETH dataset sample (b) Grad-CAM of ETH sample

(c) PIE dataset sample (d) Grad-CAM of PIE sample

Figure 2: Sample images from the BEV dataset (ETH), and EVV dataset (PIE), along with their Grad-CAM representation from U-Net.

To solve the prediction problem, various building blocks, including RNNs, 3D-CNNs, and transformers, have been employed to address the temporal dimension, with transformers demonstrating superior efficacy Giuliari et al. (2021a); Rasouli & Kotseruba (2023). However, temporal modelling alone is unaware of the social behaviour of the agents within the scene, i.e. how agents interact with one another. In addressing the social dimension, methods such as Social Pooling Pang et al. (2021) and Graph Neural Networks Kipf & Welling (2016) (GNNs) have been explored, with GNN emerging as the most effective Xu et al. (2023). Some researchers have integrated both transformers and GNN, either sequentially or in parallel, to refine the prediction paradigm Li et al. (2022); Chen et al. (2023); Jia et al. (2023); Liu et al. (2022). However, these approaches entail heightened computational burdens due to the resource-intensive nature of both GNNs and transformers. Furthermore, transformers, by their inherent design, may pose potential challenges in preserving information, as they do not inherently accommodate the graph structure in their input. On the other hand, scene dimension, or scene embedding, delves into the interaction between an agent and its surroundings. Rasouli et al. (2021) and Mangalam et al. (2021) utilised semantic segmentation maps which enhanced the model's grasp of environmental context. Another aspect across all surveyed papers, however, is their tendency to focus exclusively on either BEV or EVV, rarely considering both like Yao et al. (2021). This narrow focus becomes particularly problematic in, e.g., unstructured environments where a BEV might not be available, limiting the applicability of these methods.

In light of these challenges, this paper introduces a lightweight model, coined ASTRA (**A S**cene-aware **TRA**nsformer-based model for trajectory prediction). By integrating a U-Net-based key-point extractor Ribera et al. (2019), ASTRA captures essential scene features without relying on explicit segmentation map annotations and alleviates the data requirements and preprocessing efforts highlighted earlier. This method also synergises the strengths of GNNs in representing the social dimension of the problem and of transformers in encoding its temporal dimension. Crucially, our approach processes spatial, temporal, and social dimensions concurrently, by embedding the graph structure into the token's sequence prior to the attention mechanism, rendering the transformer graph-aware. The model does so while keeping the complexity of the model minimal. To refine the model's ability to accurately learn trajectories, we implemented a modified version of the trajectory prediction loss, incorporating a penalty component. This is in contrast to Yuan et al. (2021) which does not build a graph and does not preserve the social structure; they distinguish between self-agent and all other agents, then they treat all other agents the same without encoding the positional or structural encodings.

Furthermore, distinct from the vast majority of models in this domain, our model demonstrates generalisability by being applicable to both types of trajectory prediction datasets, BEV and EVV.

Our methodology underwent evaluation using renowned benchmark trajectory prediction datasets (Figure 2): ETH Pellegrini et al. (2009a), UCY Lerner et al. (2007), and the PIE dataset Rasouli et al. (2019). The empirical findings highlight ASTRA's outperforming the latest state-of-the-art methodologies. Notably, our method showcased significant improvements of 27% on the deterministic and 10% on the stochastic settings of the ETH and UCY datasets and 26% on PIE.

While maintaining high accuracy, ASTRA also features a significant reduction in the number of model's parameters (Figure 1) - seven times fewer than the existing competing state-of-the-art model Mao et al. (2023).

The paper's highlights are as follows:

1. A lightweight model architecture that is seven times lighter than the existing state-of-the-art model, tailored for deployment on devices with limited processing capabilities while maintaining state-of-the-art predictive performance.

2. A loss-penalization strategy that enhances trajectory prediction, featuring a weighting trajectory loss function that dynamically adjusts penalty progression in response to prediction challenges.

3. Utilisation of the Scene-aware embeddings with a U-Net-based feature extractor to encode scene representations from frames, addressing a critical aspect often overlooked in recent works.

4. A graph-aware transformer encoder that contributes significantly to generating Agent-Scene aware embedding for improved prediction accuracy, ensuring informed inter-agent interaction capture.

2 Related Work

2.1 Trajectory Prediction

The trajectory prediction problem is usually approached in two ways: stochastic (multi-modal) predictions Mao et al. (2023); Yuan et al. (2021); Xu et al. (2022b); Yao et al. (2021) and deterministic (uni-modal) predictions Helbing & Molnar (1995); Pellegrini et al. (2009b); Alahi et al. (2016). The model produces only one prediction (most probable) per input motion in a deterministic setting. In contrast, a stochastic setting involves the model generating multiple predictions for each input motion. Stochastic approaches utilize generative techniques like Conditional Variational Auto-Encoders (CVAEs) Yuan et al. (2021); Yao et al. (2021), Generative Adversarial Networks (GANs) Huang et al. (2021), Normalizing Flows Bhattacharyya et al. (2020), or Denoising Diffusion Probabilistic Models Mao et al. (2023) to introduce randomness into the prediction process, thereby generating diverse future trajectories with varying qualities for each pedestrian, aiming to elucidate the distribution of potential future coordinates of pedestrian trajectories. ASTRA also

offers both deterministic and stochastic predictions like some of the previous works Xu et al. (2023); Yao et al. (2021); Salzmann et al. (2020).

2.2 Social and Scene Dimension

The social dimension focuses on capturing agent-agent interactions, emphasizing how individuals or objects influence each other's movements within a shared space. Notably, some methodologies incorporate social pooling, concurrently with attention mechanisms Pang et al. (2021). Algorithms in this domain predominantly leverage various forms of Graph Neural Networks (GNNs) to encapsulate the social dynamics among agents. Some methods employ a fully connected undirected graph, encompassing all scene agents Xu et al. (2023); Gilles et al. (2021); Kosaraju et al. (2019). This approach, albeit comprehensive, escalates exponentially with the number of agents (nodes). Conversely, other methods opt for sparsely connected graphs, establishing connections solely among agents within a proximal range, thereby reducing the linkage count substantially Fang et al. (2021); Girase et al. (2021); Salzmann et al. (2020); Weng et al. (2021). In a similar vein, Yuan et al. (2021); Salzmann et al. (2020) proposes sparse, directional graphs, predicated on the premise that different agent types possess varying perceptual ranges. Regarding the optimal depth of GNN layers, Addanki et al. (2021) advocate for deeper graphs to enhance performance. This stands in contrast to the findings of Weng et al. (2021) and Liu et al. (2020), who posit that two layers are optimal. Nevertheless, this depth increases computational demands, particularly when agent nodes are numerous, posing challenges for autonomous vehicle applications reliant on edge devices for processing.

The scene dimension, extracted from the video frames, includes the low-level representation of the physical environment, obstacles, and any elements that could affect the agent's path, ensuring a comprehensive understanding of both social and environmental factors in predicting movement trajectories. Rasouli et al. (2021) and Mangalam et al. (2021) capture scene dimension with the help of semantic segmentation to delineate visual attributes of varied classes, subsequently elucidating their interrelations via attention. However, obtaining a panoptic segmentation mask, might not be always feasible. Also, this approach introduces a considerable dependency on the availability of additional segmentation maps, presenting a challenge in terms of data requirements and preprocessing efforts.

2.3 Temporal Dimension

Understanding the trajectory history of an agent significantly augments the predictive accuracy regarding its potential future path. Predominantly, ego-camera-based models are tailored to shorter temporal horizons and employ 3D Convolutional Neural Networks (3D-CNNs) Fang et al. (2021); Kotseruba et al. (2021). Some research, instead, adopts Hidden Markov Models (HMMs) for temporal analysis Cai et al. (2020). For scenarios necessitating extended time horizon considerations, more intricate structures are proposed, including Transformers Yuan et al. (2021); Xu et al. (2023); Giuliari et al. (2021b); Chen et al. (2021) and various forms of Recurrent Neural Networks (RNNs) Girase et al. (2021), including Long Short-Term Memory networks (LSTMs) Rasouli & Kotseruba (2023); Bhattacharyya et al. (2021); Fang et al. (2021) and Gated Recurrent Units (GRUs) Gilles et al. (2021). Both Transformer and RNN-based models have exhibited superior performance, often achieving state-of-the-art results in this domain. However, some of these models tend to address the temporal dimension in isolation from the social context. This segregated approach potentially results in information loss and contributes to an increased computational load, necessitated by the addition of separate components to handle the social dimension. Consequently, there emerges a pressing demand for integrated models capable of concurrently processing both temporal and social dimensions. A promising direction in this regard is the development of graph-aware transformers, which encapsulate the essence of both temporal dynamics and social interactions within a unified framework.

2.4 Graph-aware Transformers

Graph-aware transformers aim to compound the benefits of graphs (with their associated social embeddings) and of transformers, with their acclaimed attention mechanism and temporal embeddings. Notably, these advancements have predominantly catered to graph-centric datasets like ACTOR Tang et al. (2009) and CHAMELEON SQUIRREL Rozemberczki et al. (2021). Direct application of graph-aware transformers

remains untouched in pedestrian trajectory forecasting, with prevalent methodologies leaning towards transformers processing embeddings emanating from graphs Li et al. (2022); Chen et al. (2023); Jia et al. (2023). There has been a discernible preference for using GNN and transformer blocks, rather than fully-integrated graph-aware transformers.

A comprehensive evaluation of numerous contemporary graph-transformer models across three graph-centric datasets is conducted in Müller et al. (2023). The analysis reveals a consistent pattern: models employing Random Walk for structural encoding exhibit superior performance across all tested datasets. Building on this empirical evidence, our approach utilizes Random Walk to encode the pedestrian graph, which is then seamlessly integrated into the transformer architecture. This integration is designed to yield a graph-aware transformer, thereby enhancing the model's capability to effectively capture and interpret complex pedestrian dynamics within various environments. To the best of the authors' knowledge, this is the first work towards utilising a graph-aware transformer to solve the trajectory prediction problem, opposing many methods which use graphs along with transformers.

3 Methodology

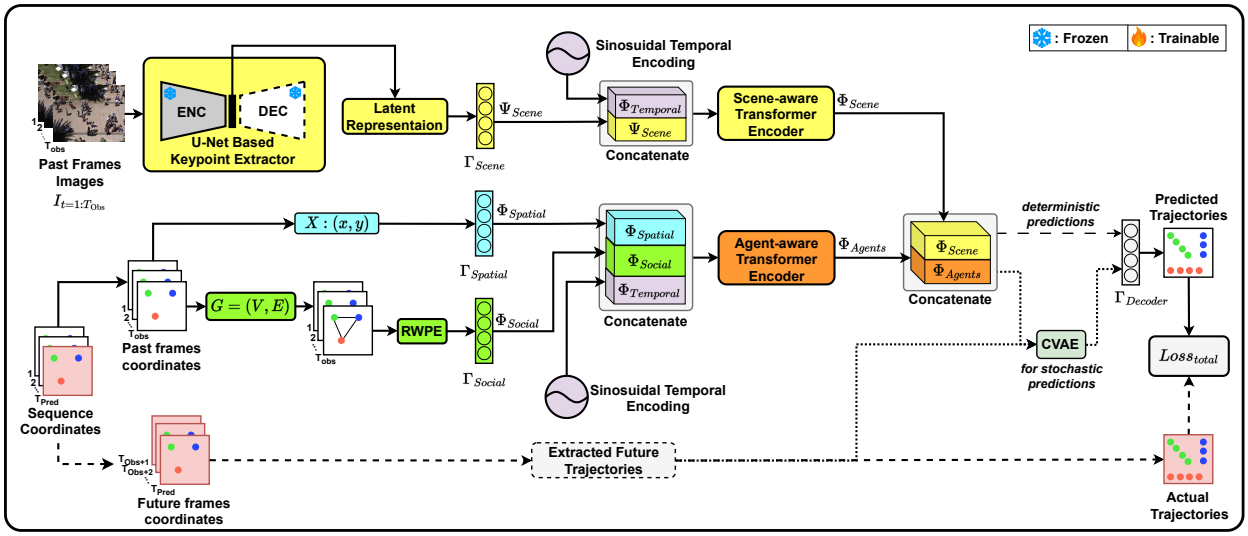


Figure 3: **Model Architecture.** Overview of ASTRA model architecture for pedestrian trajectory forecasting.

Problem Formulation. The objective of pedestrian trajectory prediction is to forecast the future position of a pedestrian based on the observed historical sequence of the pedestrian's positions. For this, the historical sequence of the pedestrians is provided as a sequence of coordinates $\boldsymbol{X} = \{X_t^a \mid t \in (1, 2, \ldots, T_{obs})\ ; a \in (1, 2, \ldots, A)\}$ related to A target agents extracted over the previous T_{obs} time instants. Here X_t^a is a pair of 2D coordinates $\{x_t^a, y_t^a\}$ for BEV datasets (see Figure 2a), and a set of bounding box coordinates, $\{x_{1,t}^a, y_{1,t}^a, x_{2,t}^a, y_{2,t}^a\}$, for EVV datasets (see Figure 2c). In addition to the sequence coordinates $\boldsymbol{X}$, T_{obs} input frames/images are also available, denoted as $\boldsymbol{I} = \{I_t \mid t \in (1, 2, \ldots, T_{obs})\}$. The goal of ASTRA is to output deterministic or multi-modal trajectories of the pedestrian. In the deterministic setting, the problem consists of predicting the output prediction coordinates of the A agents in the subsequent T_{pred} future frames, formally $\hat{\boldsymbol{Y}} = \{\hat{Y}_t^a \mid t \in (1, 2, \ldots, T_{pred});\ a \in (1, 2, \ldots, A)\}$ where $\hat{Y}_t^a$ denotes the coordinates of the agent a at a future time t, and the corresponding ground truth being $\boldsymbol{Y} = \{Y_t^a \mid t \in (1, 2, \ldots, T_{pred});\ a \in (1, 2, \ldots, A)\}$. To output multimodal trajectories we need to learn a generative model, denoted by $P_\theta(\mathcal{Y}|\boldsymbol{X}, \boldsymbol{I})$ which is parameterized by θ and given $\boldsymbol{X}$ and $\boldsymbol{I}$, outputs K predicted trajectories denoted by $\mathcal{Y} = \{\hat{\boldsymbol{Y}}^{(1)}, \hat{\boldsymbol{Y}}^{(2)}, \ldots, \hat{\boldsymbol{Y}}^{(K)}\}$.

3.1 Components

The encoder part of our model consists of two main components: A scene-aware component and an agent-aware component. While the former is dedicated to encoding the scene, and encapsulating the contextual details, the latter focuses on encoding the spatial, temporal, and social dimensions of the agents, as shown in Figure 3. The output from these two components is aggregated before being decoded to generate single or multiple predicted future trajectories for deterministic/multi-modal predictions, respectively. In order to learn essential information about the scene's spatial layout and the positional dynamics of agents within it, the U-Net Ronneberger et al. (2015) is pre-trained using the method detailed in Ribera et al. (2019) which utilizes a specialized loss function, Weighted Hausdorff Distance to learn a latent representation of the scene context (Figure 4). More sophisticated schemes to generate the scene representation, like transformer-based architectures, and fusing social representations via gated cross-attention can also be considered but we leave exploring possibly more effective and sophisticated architecture designs as future work. The Grad-CAM visualizations (Figures 2b and 2d) highlight this capability, showing that the pre-trained model pays attention to regions with a high likelihood of pedestrian activity. The U-Net-based keypoint extractor is frozen after the pretraining when used in ASTRA model architecture.

3.1.1 Temporal Encodings

To guide the network to model the temporal dimension, we add temporal encodings to the agents and scene to capture the temporal dependencies within the sequence of pedestrian trajectories from past frames, adopting a time encoder akin to the positional encoding found in the original Transformer architecture Vaswani et al. (2017).

$$\Phi_{\text{Temporal}}(t, i) = \begin{cases} \sin\left(\frac{t}{10000^{2i/d}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{t}{10000^{2i/d}}\right) & \text{if } i \text{ is odd} \end{cases} \tag{1}$$

where t is the time step, i is the dimension, and d is the dimensionality of the model.

3.1.2 Scene-aware embeddings.

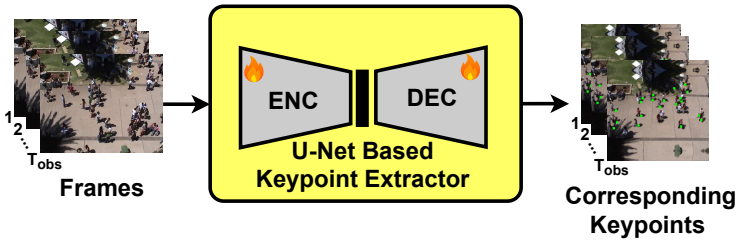
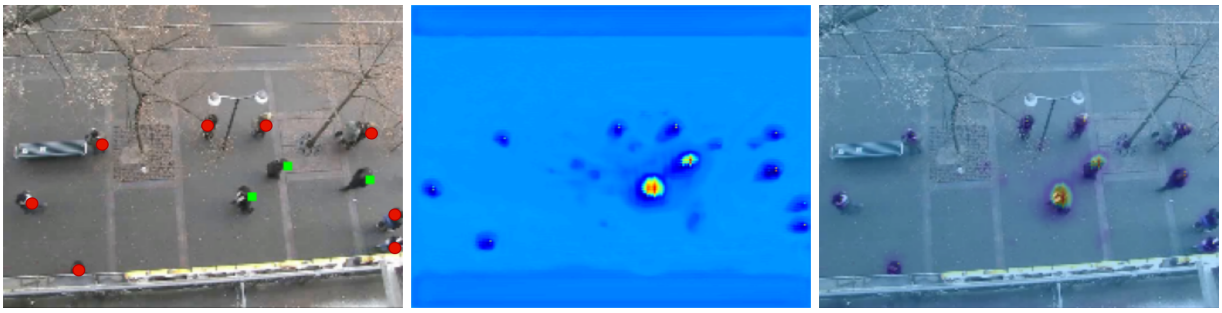


Figure 4: Pretraining U-Net based keypoint extractor.

A latent representation of pedestrian characteristics is obtained using a pre-trained U-Net encoder (Figure 3); this latent vector can include some crucial characteristics like spatial groupings and interactions with the environment. This step is crucial as the U-Net extractor possesses the proficiency to discern both labelled and unlabelled pedestrians, depicted in green and red, respectively in Figure 5a.

3.1.3 Scene-aware transformer encoder.

The latent representation of all frames (Ψ_{Scene}) is treated as input tokens to the scene-aware single-layer transformer encoder ($T_{\text{Scene-aware}}$), which in turn generates scene-aware embeddings (Φ_{Scene}) for each frame. The single-layer transformer encoder architectural choice significantly contributes to the lightweight nature of our model. The resulting scene embedding is:



(a) Original Image (b) Grad-CAM Image (c) Overlayed Grad-CAM

Figure 5: **Grad-CAM visualizations**: In (a), the red circle indicates unlabelled pedestrians, while the green square highlights labelled pedestrians. In (b), the U-Net-based keypoint extractor focuses on unlabelled pedestrians as well, thereby capturing scene context from them too.

$$\Psi_{\text{Scene}} = \Gamma_{\text{Scene}}\left(\Upsilon_{\text{Encoder}}(\boldsymbol{I})\right) \tag{2}$$

$$\Phi_{\text{Scene}} = T_{\text{Scene-aware}}\left([\Psi_{\text{Scene}}; \Phi_{\text{Temporal}}]\right) \tag{3}$$

where past input frame images projected using a Multi-Layer Perceptron (MLP) layer (Γ_{Scene}), and $\Upsilon_{\text{Encoder}}(.)$ denotes the encoder part of the U-Net.

The temporal encoding Φ_{Temporal}, crucial for capturing the temporal dynamics within the observed frames, adopts the design of the traditional positional encoding Vaswani et al. (2017) and follows the work of Yuan et al. (2021).

3.1.4 Agent-aware embeddings.

The second component is dedicated to encoding the different dimensions of each agent for all agents in the scene.

The spatial coordinates $\mathbf{X}$ of each agent are linearly projected to a latent space using an MLP layer (Γ_{Spatial}) to get spatial encoding (Φ_{Spatial})

$$\Phi_{\text{Spatial}} = \Gamma_{\text{Spatial}}(\boldsymbol{X}) \tag{4}$$

Spatial embeddings are not enough to capture the full information about the agents. If two agents in two frames have the same location in the image, their spatial embedding will be the same. Hence, temporal encoding (equation in Supplementary material) is also included to distinguish them.

Having the spatial and temporal dimensions of the agents is still not enough to understand their interaction in the scene. To capture the social dimension in this multi-agent environment, we generate a fully connected undirected graph between agents, in which the nodes are the agents' locations, and the edges between agents are the reciprocal of the distance. Consequently, the closer the agents are to each other, the stronger the link between them. Formally, we represent the social dimension using a graph $G = (V, E)$, where V is the set of agents and E is the collection of edges, with weights

$$e_{ij} = \frac{1}{d(v_i, v_j)} \tag{5}$$

where $d(v_i, v_j)$ is the distance between agents v_i and v_j.

Subsequently, Random Walk Positional Encodings (RWPEs) Dwivedi et al. (2021) is used to capture the structural relationships between nodes in the graph, such as their proximity to each other and the number

of paths between them. These RWPEs are further projected to latent space using a separate MLP (Γ_{Social}) to get social encodings (Φ_{Social}), mathematically:

$$\Phi_{\text{Social}} = \Gamma_{\text{Social}}\left(\text{RWPE}(\mathbf{G})\right) \tag{6}$$

This preserves the graph structure of the agents while making the transformer encoder graph-aware. Furthermore, our method empowers the network to determine the significance of each agent relative to others autonomously.

While we do not claim to be the first to use transformers or graphs, we claim that we are the first to integrate RWPE directly into transformer tokens, creating a graph-aware transformer in the context of trajectory prediction, which has proven successful, outperforming the state-of-the-art (SOTA).

3.1.5 Agent-aware transformer encoder.

After calculating the spatial, temporal and social representations for each agent, our model concatenates them. This concatenated vector is then fed into an agent-aware single-layer transformer encoder ($T_{\text{Agent-aware}}$) that generates agent-aware embedding (Φ_{Agents}).

$$\Phi_{\text{Agents}} = T_{\text{Agent-aware}}([\Phi_{\text{Spatial}}; \Phi_{\text{Temporal}}; \Phi_{\text{Social}}]) \tag{7}$$

3.1.6 Decoder.

To generate multi-modal trajectories, we need to learn a generative model $P_\theta(\mathcal{Y}|\boldsymbol{X}, \boldsymbol{I})$ for which we adopted CVAE to learn the inherent distribution of future target trajectories conditioned on observed past trajectories through the utilization of a stochastic latent variable. During inference, K samples are drawn from the learned distribution and decoded to future trajectories. (More details in supplementary material) For deterministic predictions, CVAE is skipped and the outputs of both the scene transformer (Φ_{Scene}) and the agents' transformer (Φ_{Agents}) are concatenated and directly passed through an MLP decoder (Γ_{Decoder}) to produce future trajectories ($\hat{Y}$) of the agents in the future frames as shown in Figure 3, namely:

$$\hat{\mathbf{Y}} = \Gamma_{\text{Decoder}}([\Phi_{\text{Scene}}; \Phi_{\text{Agents}}]) \tag{8}$$

3.1.7 Weighted trajectory loss function.

We introduce a weighted-penalty strategy that can be applied to common loss functions used in trajectory prediction such as MSE and Smooth L1 Loss. The application of this strategy is through a dynamic penalty function $w(t)$, designed to escalate or de-escalate the significance of prediction errors as one moves further into the future. The definition of the weighted loss function is given by:

$$L_{\text{weighted}}(\hat{Y}, Y) = \sum_{t=1}^{T_{pred}} w(t) \cdot L(\hat{Y}_t, Y_t), \tag{9}$$

where $\hat{Y}$ and Y are the predicted and actual trajectories respectively, T_{pred} denotes the number of prediction timesteps, $w(t)$ represents the dynamic weighting function at time t, and $L(\hat{Y}_t, Y_t)$ is the predefined loss function (e.g., MSE or SmoothL1 Loss) applied to the predicted and true positions at each time step t.

The weight function $w(t)$ is designed to be versatile, accommodating a spectrum of mathematical formulations that align with the specific needs of the predictive model. It is generically defined as:

$$w(t) = f(t, T_{\text{pred}}, \alpha, \beta), \tag{10}$$

where α and β are parameters that establish the bounds of the weighting function, and f is an adaptable function that governs the progression of weights at each timestep t.

In particular, the function $w(t)$ may be selected from various mathematical forms, such as linear, parabolic, or quadratic, which are discussed in greater detail within the supplementary materials. The choice of function

enables the model to adjust the penalty progression in alignment with the anticipated prediction challenge at each timestep.

To generate multi-modal trajectories, the entire ASTRA model is optimized using Equation 11. The KL divergence term ensures that the prior network implicitly learns the dependency between $\boldsymbol{Y}$ and $\boldsymbol{X}$.

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) + D_{KL}(\mathcal{N}(\mu_{z_q}, \sigma_{z_q})||\mathcal{N}(\mu_{z_p}, \sigma_{z_p})) \tag{11}$$

In this context, μ_{z_p} and σ_{z_p} represent the mean and variance of the latent distribution $p_\theta(z_p|x)$, parameterized by θ. These parameters are learned during training and are used to approximate the true distribution of the latent variables, enabling the generation of multiple plausible trajectories. The pair $(\mu_{z_p}, \sigma_{z_p})$ is contrasted with $(\mu_{z_q}, \sigma_{z_q})$, which parameterizes the ground truth distribution $q_\Phi(z_q|x, y)$, helping to guide the model towards accurate predictions through the KL divergence term. For deterministic trajectories loss, KL divergence term is dropped from the equation 11. More details are discussed in the supplementary.

4 Experiments

4.1 Datasets and Evaluation Protocol

For a comprehensive evaluation, we benchmarked our model on three trajectory prediction datasets; namely, ETH Pellegrini et al. (2009a), UCY Lerner et al. (2007), and PIE dataset Rasouli et al. (2019). **ETH** and **UCY** offer a bird's-eye view of pedestrian dynamics in urban settings, including five datasets with 1,536 pedestrians across four scenes. For evaluation, we used their standard protocol; leave-one-out strategy, observing eight time steps (3.2s) and predicting the following 12 steps (4.8s).

In contrast, the **PIE dataset** provides an Ego-vehicle perspective, containing over 6 hours of ego-centric driving footage, along with bounding box annotations for traffic objects, action labels for pedestrians, and ego-vehicle sensor information Rasouli et al. (2019). A total of 1,842 pedestrian samples are considered with the following split: Training(50%), Validation(40%) and Testing(10%)Rasouli et al. (2019). Model performance is evaluated based on a shorter observational window of 0.5 seconds and a prediction window of 1 second, providing insights into the model's capability in rapidly evolving traffic scenariosRasouli & Kotseruba (2023).

4.2 Evaluation Metrics

We used the standard evaluation metrics of ADE and FDE for ETH-UCY deterministic settings and minADE and minFDE for their stochastic settings. ADE, FDE, CADE, CFDE, ARB and FRB for PIE dataset, the supplementary material explains these metrics.

4.3 Setting up the experiments

We trained the model on the ETH-UCY and PIE datasets using the AdamW optimizer with a weight decay of 5×10^{-4} for 200 epochs. The initial learning rate was set to 1×10^{-3}, and a cosine annealing scheduler was employed. Training was conducted on a NVIDIA DGX A100 machine, equipped with 8 GPUs, each having 80 GB of memory.

4.4 Discussing the Results

4.4.1 Quantitative Results.

For ETH-UCY, we compared our model results against several baselines. These comparisons are presented in Table 1 and Table 2, which contains results primarily sourced from the EqMotion (CVPR 2023) Xu et al. (2023) for deterministic predictions and LeapFrog (CVPR 2023) Mao et al. (2023) for stochastic predictions respectively. It is important to note that to provide a thorough comparative framework, we independently computed and included additional models Wang et al. (2022); Yao et al. (2021) to their respective tables as

Table 1: **Deterministic Results**: ADE/FDE results for ETH-UCY baselines. Best in **bold**, second best underlined.

Model	ETH	Hotel	Univ	Zara1	Zara2	Average
Linear	1.33/2.94	0.39/0.72	0.82/1.59	0.62/1.21	0.77/1.48	0.79/1.59
S-LSTMAlahi et al. (2016)	1.09/2.35	0.79/1.76	0.67/1.40	0.47/1.00	0.56/1.17	0.72/1.54
S-AttentionVemula et al. (2018)	1.39/2.39	2.51/2.91	1.25/2.54	1.01/2.17	0.88/1.75	1.41/2.35
SGAN-indGupta et al. (2018)	1.13/2.21	1.01/2.18	0.60/1.28	0.42/0.91	0.52/1.11	0.74/1.54
Traj++Salzmann et al. (2020)	1.02/2.00	0.33/0.62	0.53/1.19	0.44/0.99	0.32/0.73	0.53/1.11
TransFGiuliari et al. (2021b)	1.03/2.10	0.36/0.71	0.53/1.32	0.44/1.00	0.34/0.76	0.54/1.17
MemoNetXu et al. (2022b)	1.00/2.08	0.35/0.67	0.55/1.19	0.46/1.00	0.37/0.82	0.55/1.15
SGNetWang et al. (2022)	0.81/1.60	0.41/0.87	0.58/1.24	0.37/0.79	0.31/0.68	0.56/1.04
EqMotionXu et al. (2023)	0.96/1.92	0.30/0.58	**0.50**/1.10	0.39/0.86	0.30/0.68	0.49/1.03
ASTRA (Ours)	**0.47**/**0.82**	**0.29**/**0.56**	0.55/**1.00**	**0.34**/**0.71**	**0.24**/**0.41**	**0.38**/**0.70**

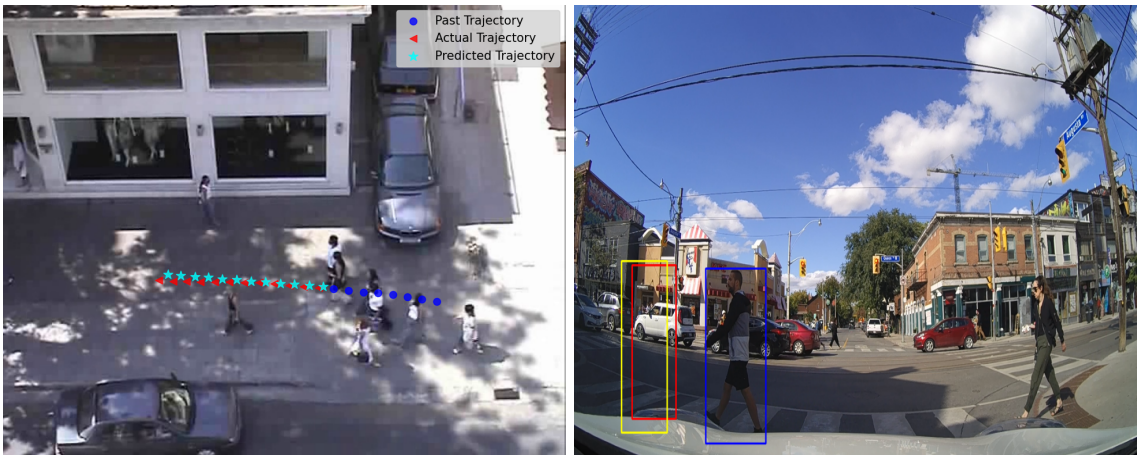
Table 2: **Stochastic Results**: minADE_{20} and minFDE_{20} results for ETH-UCY baselines. Best in **bold**, second best underlined. NP- means unpenalised.

Model	ETH	Hotel	Univ	Zara1	Zara2	Average
Social-GAN Gupta et al. (2018)	0.87/1.62	0.67/1.37	0.76/1.52	0.35/0.68	0.42/0.84	0.61/1.21
NMMP Hu et al. (2020)	0.61/1.08	0.33/0.63	0.52/1.11	0.32/0.66	0.43/0.85	0.41/0.82
STAR Yu et al. (2020a)	0.36/0.65	0.17/0.36	0.31/0.62	0.29/0.52	0.22/0.46	0.26/0.53
PECNet Mangalam et al. (2020)	0.54/0.87	0.18/0.24	0.35/0.60	0.22/0.39	0.17/0.30	0.29/0.48
Trajectron++ Salzmann et al. (2020)	0.61/1.02	0.19/0.28	0.30/0.54	0.24/0.42	0.18/0.32	0.30/0.51
BiTrap-NP Yao et al. (2021)	0.55/0.95	0.17/0.28	0.25/0.47	0.22/0.44	0.16/0.33	0.27/0.49
MemoNet Xu et al. (2022b)	0.40/0.61	**0.11**/**0.17**	0.24/0.43	0.18/0.32	0.14/0.24	0.21/0.35
GroupNet Xu et al. (2022a)	0.40/0.76	0.12/0.18	**0.22**/**0.41**	0.17/0.31	**0.12**/0.24	0.21/0.38
SGNet Wang et al. (2022)	0.47/0.77	0.20/0.38	0.33/0.62	0.18/0.32	0.15/0.28	0.27/0.47
MID Gu et al. (2022)	0.46/0.73	0.15/0.25	0.26/0.49	0.21/0.39	0.17/0.33	0.25/0.44
Agentformer Yuan et al. (2021)	0.45/0.75	0.14/0.22	0.25/0.45	0.18/0.30	0.14/0.24	0.23/0.39
EqMotion Xu et al. (2023)	0.40/0.61	0.12/0.18	0.23/0.43	0.18/0.32	0.13/0.23	0.21/0.35
Leapfrog Mao et al. (2023)	0.39/0.58	**0.11**/**0.17**	0.26/0.43	0.18/0.26	0.13/0.22	0.21/0.33
ASTRA(Non Penalised)	0.37/0.49	0.24/0.34	0.37/0.52	0.23/0.32	0.16/0.23	0.27/0.38
ASTRA(Without Frame Encoding)	0.29/0.39	0.18/ 0.29	0.29/ 0.43	0.17/0.26	0.14/0.2	0.21/ 0.31
ASTRA(With Frame Encoding)	**0.27**/**0.36**	0.17/0.25	0.28/**0.41**	**0.15**/**0.23**	0.13/**0.16**	**0.20**/**0.28**

they were not originally part of the EqMotion or LeapFrog analysis. Our model significantly advances the state-of-the-art on ETH-UCY, outperforming the EqMotion Xu et al. (2023) by improving predictive accuracy by approximately 27% on average for deterministic predictions as shown in Table 1 and approximately 10% on average improvement over LeapFrogMao et al. (2023) for stochastic predictions as shown in Table 2, highlighting the efficacy of our approach in diverse scenarios. We also highlight the effectiveness of utilizing frame encodings from U-Net, as demonstrated in Table 2.

Similarly, we benchmarked our model against various established models for the PIE dataset. The comparative analysis is summarized in Table 4, with the reference results taken from the PedFormer Rasouli & Kotseruba (2023), demonstrating an average improvement of 26%.

Trainable Parameters. Regarding the computational side, ASTRA has seven times fewer trainable parameters than the existing SOTA model LeapFrog Mao et al. (2023) as shown in Figure 1. It is important to note that the parameter count reported for ASTRA in Figure 1 includes the parameters from the U-Net, which is otherwise actually frozen during the trajectory prediction phase. For a fair comparison, the parameter count reported for other models (like Yuan et al. (2021) is considered by taking their respective pre-trained parameters into account.



(a) BEV (b) EVV

Figure 6: Sample images of the deterministic prediction from BEV datasets (a.) (ETH and UCY) and EVV dataset (b.) (PIE). The Red and Yellow bounding box indicates the ground-truth and predicted final position respectively and the Blue bounding box indicates the start position.

4.4.2 Qualitative results.

We can clearly see the results of our prediction from Figure 6 for deterministic predictions and Figure 8 for stochastic prediction. Figure 7 exemplifies the proximity of our model's results to the ground truth, it also shows how using the weighted penalty strategy has yielded better results than the unpenalised one, highlighting the improved effectiveness of our strategy.

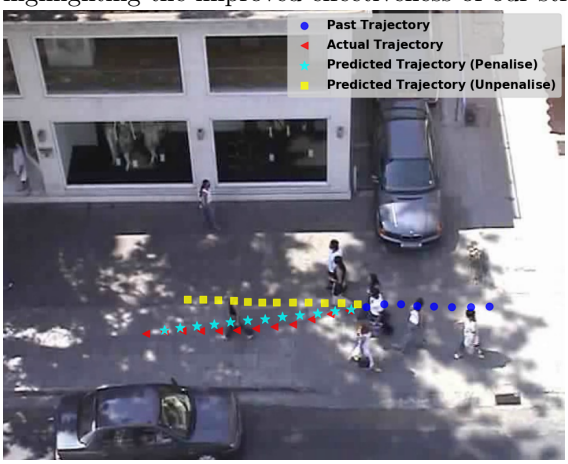


Figure 7: Visual comparison of penalised and unpenalised loss on ETH-UCY, showing the enhanced performance of the former.

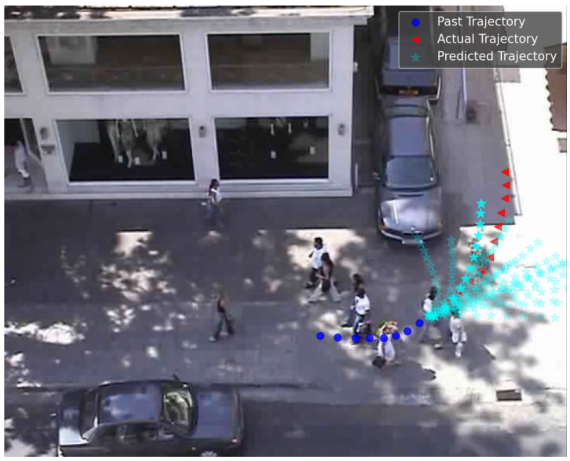


Figure 8: Sample images of the stochastic predictions from ETH-UCY Dataset.

4.4.3 Ablation studies.

In the ablation study presented in Table 3, we evaluated the contribution of each component in our trajectory prediction model to ascertain their individual and collective impact on the performance metrics on the ETH-UCY (UNIV) dataset. Initially, the model incorporated only spatial information, which served as a baseline for subsequent enhancements. The sequential integration of temporal and social components yielded successive improvements, demonstrating their respective significance in capturing the dynamics of agent movement. The addition of the data augmentation technique (detailed in the supplementary mate-

Table 3: **Ablation:** ADE/FDE & $\text{minADE}_{20}/\text{minFDE}_{20}$ with variations in ASTRA's model components on UNIV dataset (where ✓: Component enabled, ×: Component disabled)

Spatial	**Temporal**	**Augmentation**	**Social**	**U-Net Features**	**ADE/FDE**	**minADE_{20}/ minFDE_{20}**
✓	×	×	×	×	1.05/1.66	0.43/0.63
✓	✓	×	×	×	0.86/1.47	0.39/0.54
✓	✓	✓	×	×	0.67/1.17	0.31/0.48
✓	✓	✓	✓	×	0.66/1.12	0.29/0.43
✓	✓	✓	✓	✓	0.55/1.00	0.28/0.41

rial) further refined the model's performance, illustrating the value of varied training samples in enhancing generalization capabilities. Moreover, the incorporation of U-Net features contributed to a substantial leap forward, highlighting the importance of context-aware embeddings in accurately forecasting agent trajectories. This progression emphasizes the synergistic effect of combining heterogeneous data representations to capture the nuanced patterns of movement within a scene.

The ablation study also extended to the evaluation of loss functions, comparing the effects of penalised versus unpenalised approaches. Penalized loss functions, designed to focus the model's attention on more critical prediction horizons, proved to be more effective in refining the predictive accuracy, as outlined in Table 5 and in Table 2 (ASTRA(NP)) for deterministic and stochastic setting respectively and the same can be observed in Figure 7.

Table 4: Results for PIE dataset.

Model	**CADE**	**CFDE**	**ARB**	**FRB**
FOLYao et al. (2019)	73.87	164.53	78.16	143.69
FPLYagi et al. (2018)	56.66	132.23	-	-
B-LSTMBhattacharyya et al. (2018)	27.09	66.74	37.41	75.87
PIE_{traj}Rasouli et al. (2019)	21.82	53.63	27.16	55.39
PIE_{full}Rasouli et al. (2019)	19.50	45.27	24.40	49.09
BiPedRasouli et al. (2020)	15.21	35.03	19.62	39.12
PedFormerRasouli & Kotseruba (2023)	13.08	30.35	**15.27**	32.79
ASTRA(Ours)	**9.91**	**22.42**	18.32	**17.07**

Table 5: **Ablation:** ADE/FDE for penalised vs. unpenalised loss functions on UNIV dataset using SOTA ASTRA's configuration

Loss	**Normal**	**Penalised**
MSE	0.58/1.13	0.57/1.00
SmoothL1	0.57/1.15	0.55/1.00

5 Conclusion

We presented ASTRA, a model in the domain of pedestrian trajectory prediction, that outperforms the existing state-of-the-art models. This advancement renders ASTRA particularly suitable for deployment on devices with limited processing capabilities, thereby broadening the applicability of high-accuracy trajectory prediction technologies. ASTRA's adeptness in handling both BEV and EVV modalities further solidifies its applicability in diverse operational contexts. With the ability to produce deterministic and stochastic results, it enhances the predictive robustness and situational awareness of autonomous systems. Moving forward, we aim to extend the capabilities of the ASTRA model beyond pedestrian trajectory prediction to encompass a broader range of non-human agents. This expansion will involve adapting the model to understand and predict the movements of various entities within shared environments using more sophisticated architectural design choices to encode the scene and its fusion with social dimension. By broadening our focus, we hope to contribute to the development of truly comprehensive and adaptive systems capable of navigating the complexities of real-world interactions among a wide array of agents.

References

Ravichandra Addanki, Peter W Battaglia, David Budden, Andreea Deac, Jonathan Godwin, Thomas Keck, Wai Lok Sibon Li, Alvaro Sanchez-Gonzalez, Jacklynn Stott, Shantanu Thakoor, et al. Large-scale graph representation learning with very deep gnns and self-supervision. *arXiv preprint arXiv:2107.09422*, 2021.

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

Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty, 2018.

Apratim Bhattacharyya, Christoph-Nikolas Straehle, Mario Fritz, and Bernt Schiele. Haar wavelet based block autoregressive flows for trajectories. *CoRR*, abs/2009.09878, 2020. URL `https://arxiv.org/abs/2009.09878`.

Apratim Bhattacharyya, Daniel Olmeda Reino, Mario Fritz, and Bernt Schiele. Euro-pvi: Pedestrian vehicle interactions in dense urban centers. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6404–6413, 2021.

Wenqi Cai, Ganglei He, Jianlong Hu, Haiyan Zhao, Yuhai Wang, and Bingzhao Gao. A comprehensive intention prediction method considering vehicle interaction. *2020 4th CAA International Conference on Vehicular Control and Intelligence, CVCI 2020*, pp. 204–209, 12 2020. doi: 10.1109/CVCI51460.2020.9338520.

Weihuang Chen, Fangfang Wang, and Hongbin Sun. S2tnet: Spatio-temporal transformer networks for trajectory prediction in autonomous driving. In *Asian Conference on Machine Learning*, pp. 454–469. PMLR, 2021.

Xiaobo Chen, Huanjia Zhang, Yu Hu, Jun Liang, and Hai Wang. Vnagt: Variational non-autoregressive graph transformer network for multi-agent trajectory prediction. *IEEE Transactions on Vehicular Technology*, pp. 1–12, 2023. doi: 10.1109/TVT.2023.3273230.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Jianwu Fang, Dingxin Yan, Jiahuan Qiao, Jianru Xue, and Hongkai Yu. Dada: Driver attention prediction in driving accident scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 2021. ISSN 15580016. doi: 10.1109/TITS.2020.3044678.

Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. *arXiv preprint arXiv:2109.01827*, 2021.

Harshayu Girase, Haiming Gang, Srikanth Malla, Jiachen Li, Akira Kanehara, Karttikeya Mangalam, and Chiho Choi. Loki: Long term and key intentions for trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9803–9812, 2021.

Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pp. 10335–10342. IEEE, 2021a.

Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pp. 10335–10342. IEEE, 2021b.

Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17113–17122, June 2022.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2255–2264, 2018.

Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

Yue Hu, Siheng Chen, Ya Zhang, and Xiao Gu. Collaborative motion prediction via neural motion message passing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6319–6328, 2020.

Lei Huang, Jihui Zhuang, Xiaoming Cheng, Riming Xu, and Hongjie Ma. Sti-gan: Multimodal pedestrian trajectory prediction using spatiotemporal interactions and a generative adversarial network. *IEEE Access*, 9:50846–50856, 2021.

Xiaosong Jia, Penghao Wu, Li Chen, Yu Liu, Hongyang Li, and Junchi Yan. Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian D. Reid, Seyed Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *NeurIPS*, 2019.

Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. Benchmark for evaluating pedestrian action prediction. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1257–1267, 2021. doi: 10.1109/WACV48630.2021.00130.

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007. doi: https://doi.org/10.1111/j.1467-8659.2007.01089.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x`.

Lihuan Li, Maurice Pagnucco, and Yang Song. Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2231–2241, 2022.

Bingbin Liu, Ehsan Adeli, Zhangjie Cao, Kuan-Hui Lee, Abhijeet Shenoi, Adrien Gaidon, and Juan Carlos Niebles. Spatiotemporal relationship reasoning for pedestrian intent prediction. *IEEE Robotics and Automation Letters*, 5(2):3485–3492, 2020.

Yao Liu, Lina Yao, Binghao Li, Xianzhi Wang, and Claude Sammut. Social graph transformer networks for pedestrian trajectory prediction in complex social scenarios. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, pp. 1339–1349, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557455. URL `https://doi.org/10.1145/3511808.3557455`.

Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 759–776. Springer, 2020.

Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15233–15242, 2021.

Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5517–5526, June 2023.

Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14424–14432, 2020.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*, 2023.

Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11809–11819, 2021. doi: 10.1109/CVPR46437.2021.01164.

S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, 2009a. doi: 10.1109/ICCV.2009.5459260.

Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pp. 261–268. IEEE, 2009b.

Amir Rasouli and Iuliia Kotseruba. Pedformer: Pedestrian behavior prediction via cross-modal attention modulation and gated multitask learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9844–9851. IEEE, 2023.

Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6261–6270, 2019. doi: 10.1109/ICCV.2019.00636.

Amir Rasouli, Mohsen Rohani, and Jun Luo. Pedestrian behavior prediction via multitask learning and categorical interaction modeling. *CoRR*, abs/2012.03298, 2020. URL `https://arxiv.org/abs/2012.03298`.

Amir Rasouli, Mohsen Rohani, and Jun Luo. Bifold and semantic reasoning for pedestrian behavior prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15600–15610, 2021.

Javier Ribera, David Guera, Yuhao Chen, and Edward J Delp. Locating objects without bounding boxes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6479–6489, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 683–700. Springer, 2020.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 807–816, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557108. URL https://doi.org/10.1145/1557019.1557108.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pp. 4601–4607. IEEE, 2018.

Kira Vinogradova, Alexandr Dibrov, and Gene Myers. Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract). *Proceedings of the AAAI*, 34(10): 13943–13944, April 2020. ISSN 2159-5399. doi: 10.1609/aaai.v34i10.7244. URL http://dx.doi.org/10.1609/aaai.v34i10.7244.

Chuhua Wang, Yuchen Wang, Mingze Xu, and David J Crandall. Stepwise goal-driven networks for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, 2022.

Xinshuo Weng, Ye Yuan, and Kris Kitani. Ptp: Parallelized tracking and prediction with graph neural networks and diversity sampling. *IEEE Robotics and Automation Letters*, 6:4640–4647, 7 2021. ISSN 23773766. doi: 10.1109/LRA.2021.3068925.

Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6498–6507, June 2022a.

Chenxin Xu, Weibo Mao, Wenjun Zhang, and Siheng Chen. Remember intentions: Retrospective-memory-based trajectory prediction, June 2022b.

Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1410–1420, 2023.

Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos, 2018.

Yu Yao, Mingze Xu, Chiho Choi, David J Crandall, Ella M Atkins, and Behzad Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9711–9717. IEEE, 2019.

Yu Yao, Ella Atkins, Matthew Johnson-Roberson, Ram Vasudevan, and Xiaoxiao Du. Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation. *IEEE Robotics and Automation Letters*, 6(2):1463–1470, 2021.

Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 507–523. Springer, 2020a.

Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 507–523. Springer, 2020b.

Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Simone Zamboni, Zekarias Tilahun Kefato, Sarunas Girdzijauskas, Christoffer Norén, and Laura Dal Col. Pedestrian trajectory prediction with convolutional neural networks. *Pattern Recognition*, 121:108252, 2022.

A Conditional Variational Auto-Encoder Preliminaries

ASTRA employs a Conditional Variational Autoencoder (CVAE) Kingma & Welling (2013) framework to address the inherent stochasticity of the prediction task, enabling the generation of K distinct trajectories for each agent under consideration. Throughout the training phase, it aims to approximate the latent distribution Z by deducing its mean and variance by utilising Multilayer Perceptrons (MLPs). Subsequently, by employing the divergence loss function (second term in Equation 13), the model pushes the learned distribution $p_\theta(z_p|x)$, parameterized by θ, to be as close as possible to the ground truth distribution $q_\Phi(z_q|x, y)$, parameterized by Φ. We use the reparameterization trick to present z_p and z_q through the mean and variance pairs of $(\mu_{z_p}, \sigma_{z_p})$ and $(\mu_{z_q}, \sigma_{z_q})$, respectively. After training, K samples are drawn from the Z distribution and decoded to form the final trajectories.

B Loss Function Formulation

The loss for multi-modal trajectories is given in equations equation 12 and equation 13.
$L_{\text{weighted}}(\mathcal{Y}_k, \boldsymbol{Y})$ is calculated similar to Equation 14.

$$L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) = \min_{k=1,\ldots,K} L_{\text{weighted}}(\mathcal{Y}_k, \boldsymbol{Y}) \tag{12}$$

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) + D_{KL}(\mathcal{N}(\mu_{z_q}, \sigma_{z_q}) \mid\mid \mathcal{N}(\mu_{z_p}, \sigma_{z_p})) \tag{13}$$

For deterministic predictions, the final loss is the same as the weighted loss:

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{Y}, Y) = \sum_{t=1}^{T_{pred}} w(t) \cdot L(\hat{Y}_t, Y_t), \tag{14}$$

where $w(t)$ represent the weighted penalty function (section C) and $L(\hat{Y}_t, Y_t)$ is the predefined loss function: MSE or Smooth L1 loss (discussed below).

Mean square error (MSE)

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{15}$$

where y_i and $\hat{y}_i$ represent, the actual and predicted coordinates, respectively. MSE penalises larger trajectory prediction errors more heavily, ensuring model accuracy in critical scenarios.

Smooth L1 loss (SL1)

$$\text{SL1}(y_i, \hat{y}_i) = \begin{cases} 0.5 \times (y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| < 1 \\ |y_i - \hat{y}_i| - 0.5 & \text{otherwise.} \end{cases} \tag{16}$$

Unlike MSE, SL1 effectively balances the treatment of small and large errors. This loss is also less sensitive to outliers, due to its combination of L1 and L2 loss properties.

C Weighted Penalty Functions

In trajectory prediction, particularly for dynamic entities like pedestrians, vehicles, or other agents, the accuracy of predictions is paramount. The inherent challenge lies in managing the variability and uncertainty that escalates with longer prediction horizons. To address this, we implement different penalization strategies that adjust the model's emphasis to enhance reliability over extended prediction horizons. Our ablation focuses on three distinct penalty strategies: Linear, Quadratic, and Parabolic. Table 6 presents a quantitative analysis comparing the three penalty strategies as applied to the ETH-UCY (UNIV) dataset using SL1 loss.

It can be observed that the Parabolic penalty gives better results compared to other penalization strategies. Figure 9 compares the three weighted penalty strategies for a prediction window of 12 frames. Subsequent sections provide a detailed explanation for each of the penalty strategies.

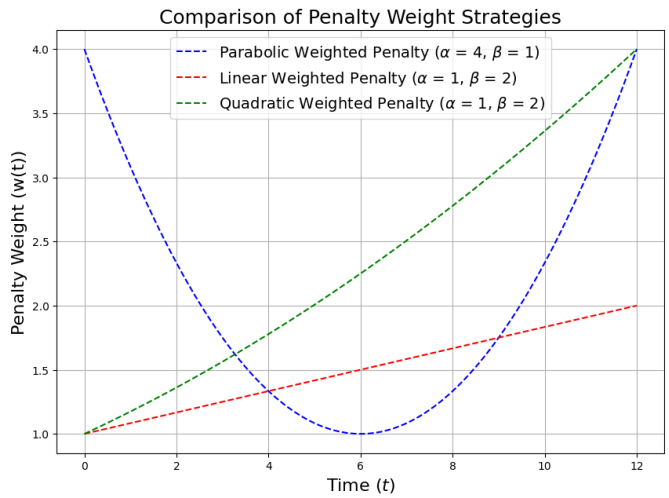


Figure 9: Comparison of various weighted penalty strategies

C.1 Linear Weighted Penalty

The Linear Weighted Penalty employs a weight function, $w(t)$, that linearly increases from a start weight (α) to an end weight (β), over the prediction period. This approach aims to progressively increase the penalty for prediction inaccuracies, particularly toward the latter part of the prediction horizon.

The weight function $w(t)$ is defined as:

$$w(t) = \alpha + \frac{t}{T_{pred}} \cdot (\beta - \alpha), \tag{17}$$

where α and β are the weights assigned to the initial and final predicted time steps, respectively.

C.2 Quadratic Weighted Penalty

The quadratic weighted penalty strategy intensifies the penalty in a quadratic manner as the difference between the prediction time and the past frames increases. This approach is more aggressive than the linear strategy, applying an exponentially increasing weight to errors in later prediction frames. The weight function $w(t)$ in this case is defined as:

$$w(t) = \left(\alpha + \frac{t}{T_{\text{pred}}} \cdot (\beta - \alpha)\right)^2 \tag{18}$$

C.3 Parabolic Weighted Penalty

The Parabolic Weighted Penalty assigns the maximum weight, α, to both the initial and final predicted time steps, highlighting their significance. Meanwhile, the minimum weight, β ($\beta < \alpha$), is allocated to the midpoint of the prediction interval. This distribution forms a parabolic trajectory (shown in Figure 9) of weights across the prediction period, as defined by:

$$w(t) = (\alpha - \beta) \cdot \left(2 \cdot \frac{t}{T_{pred}} - 1\right)^2 + \beta, \tag{19}$$

(a) Univ (No Penalty)

(b) Univ (Penalty)

(c) Zara01 (No Penalty)

(d) Zara01 (Penalty)

Figure 10: Qualitative comparison of unpenalised vs. penalised trajectories on ETH-UCY dataset in stochastic setting.

(a) Univ (No Penalty)

(b) Univ (Penalty)

(c) Zara01 (No Penalty)

(d) Zara01 (Penalty)

Figure 11: Qualitative comparison of unpenalised vs. penalised trajectories on ETH-UCY dataset in stochastic setting.

Table 6: **Ablation:** Comparing penalization strategies with SL1 loss on ETH-UCY (UNIV) dataset using ASTRA's SOTA configuration

Loss	minADE$_{20}$/minFDE$_{20}$
Unpenalised	0.37/0.52
Linear	0.33/0.47
Quadratic	0.30/0.46
Parabolic	**0.28/0.41**

C.4 Augmentation

To enhance the robustness and generalization of our trajectory prediction model, we implement a data augmentation strategy. This strategy randomly applies rotation and translation transformations to the trajectory sequences Zamboni et al. (2022). By applying random rotations and translations, our model is trained to be orientation-agnostic and adept at handling positional shifts in agents. These augmentations effectively increase the diversity of the training data, enabling the model to learn more generalized representations of agent movements. This, in turn, enhances the model's predictive accuracy and robustness, particularly in complex and unpredictable scenarios where pedestrian trajectories can vary significantly due to factors like group dynamics, obstacles, and varying crowd densities.

D Evaluation Metrics

ETH-UCY. To evaluate our model on ETH-UCY, we used commonly employed evaluation metrics Xu et al. (2023); Mohamed et al. (2020); Chen et al. (2023); Yu et al. (2020b): ADE/FDE and minADE$_K$/minFDE$_K$. Average Displacement Error (ADE) computes the average Euclidean distance between the predicted trajectory and the true trajectory across all prediction time steps for each agent. minADE$_K$ refers to the minimum ADE out of K randomly generated trajectories and ground truth future trajectories. We also used the Final Displacement Error (FDE), which focuses on the prediction accuracy at the final time step. It computes the Euclidean distance between the predicted and actual positions of each agent at the last prediction time step. minFDE$_K$ refers to the minimum FDE out of K randomly generated trajectories and ground truth future trajectories. For multimodal trajectory prediction, minADE$_K$ and minFDE$_K$ metrics are used for evaluation.

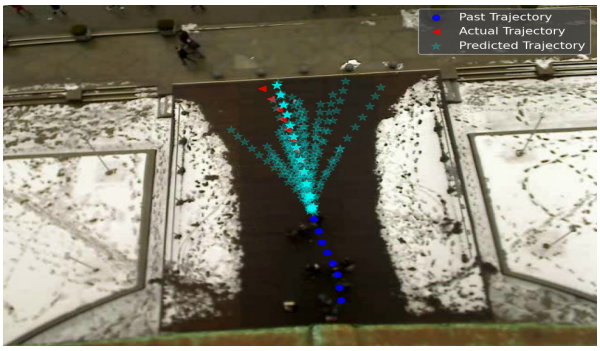
$$\text{ADE} = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \|Y_t^a - \hat{Y}_t^a\|_2. \tag{20}$$

$$\text{minADE}_K = \min_k \left(\frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \|Y_t^a - \hat{Y}_{t,k}^a\|_2 \right) \tag{21}$$

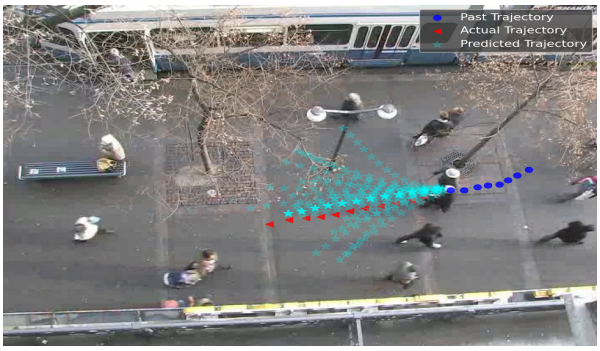
$$\text{FDE} = \|Y_{T_{pred}}^a - \hat{Y}_{T_{pred}}^a\|_2 \tag{22}$$

$$\text{minFDE}_K = \min_k \left(\|Y_{T_{pred}}^a - \hat{Y}_{T_{pred},k}^a\|_2 \right) \tag{23}$$

PIE. For the PIE Dataset, the ADE and FDE metrics are calculated based on the centroid of the bounding box Rasouli et al. (2019); Rasouli & Kotseruba (2023), denoted as Centre average displacement error for the bounding box (CADE) and Centre final displacement error for the bounding box (CFDE). In addition, we reported the average and final Root Mean Square Error (RMSE) of bounding box coordinates, denoted as ARB and FRB, respectively Rasouli et al. (2021).

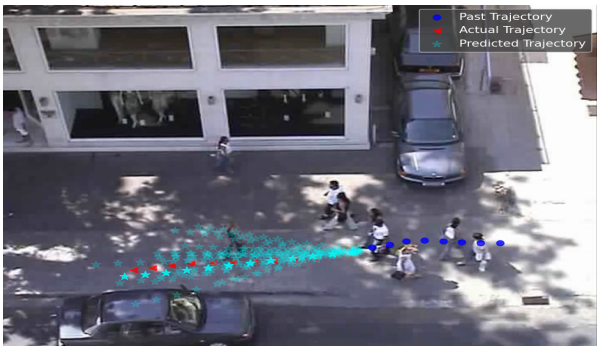


(a) ETH



(b) Hotel

(c) Univ



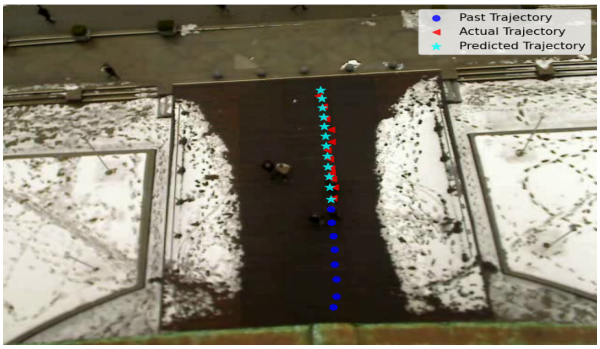
(d) Zara1

(e) Zara2

Figure 12: Multi-modal trajectory visualizations on ETH-UCY dataset (BEV)

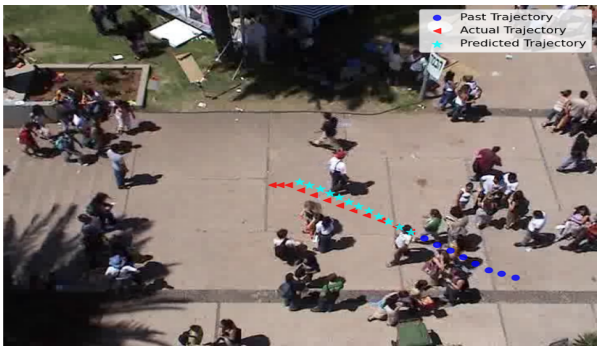
E Grad-CAM visualizations

Grad-CAM images were obtained by generating heatmaps overlaid onto the original image to aid in validating the relevance of highlighted regions. To obtain the Grad-CAM visualization, a single-channel output segmentation map was obtained from the pre-trained U-Net network, representing the probability of each pixel location being a keypoint Ribera et al. (2019). Probabilities were aggregated across all pixels, by comparing them with true keypoints and gradients of activation for the initial layer were extracted, similar to the approach taken by Vinogradova et al. (2020). Utilizing these gradients, a weighted average of the activation maps of the initial layer was computed to reconstruct the heatmap, similar to the method described in Selvaraju et al. (2017), for the Grad-CAM visualization. Overlaying this heatmap onto the original image highlights the regions that contribute significantly to the keypoint predictions made by the model.

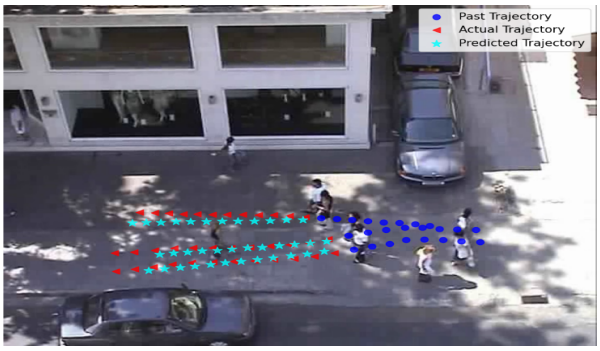


(a) ETH

(b) Hotel



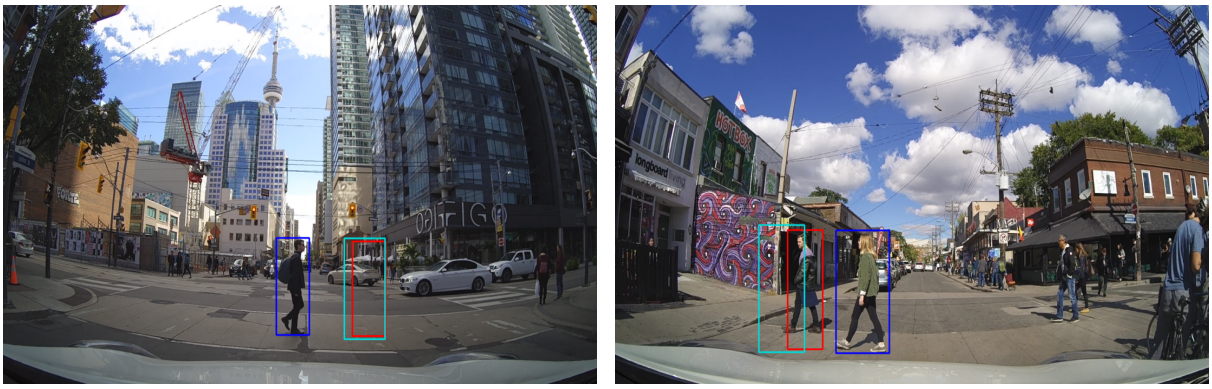
(c) Univ



(d) Zara1

(e) Zara2

Figure 13: Deterministic trajectory visualizations on ETH-UCY dataset (BEV)



(a) (b)

Figure 14: Trajectory Visualizations on PIE Dataset (EVV) where the red and cyan bounding box indicates the ground-truth and predicted final position respectively and the blue bounding box indicates the start position.

To ease reading the paper, Table 7 and 8 list the abbreviations and the mathematical symbols mentioned in the paper, respectively.

Table 7: Table of Abbreviations Used

Abbreviation/Term	**Description**
ASTRA	Agent-Scene aware model for pedestrian trajectory forecasting
BEV	Bird's Eye View
EVV	Ego-Vehicle View
AV	Autonomous Vehicle
MLP	Multi-Layer Perceptron
CVAE	Conditional Variational Auto-Encoder
GNN	Graph Neural Network
RWPE	Random Walk Positional Encoding
MSE	Mean Square Error (Loss Function)
SL1	Smooth L1 Loss (Loss Function)
ADE	Average Displacement Error
FDE	Final Displacement Error
CADE	Centre average displacement error for the bounding box
CFDE	Centre final displacement error for the bounding box
ARB	Average Root Mean Square Error for the bounding box
FRB	Final Root Mean Square Error for the bounding box

Table 8: Table of Mathematical Symbols Used

Symbols	**Description**
N	Total number of predictions in MSE calculation
$\boldsymbol{X}$	Observed trajectories of agents
$\boldsymbol{Y}$	Groundtruth future trajectories of agents
$\hat{\boldsymbol{Y}}$	Predicted trajectories of agents
T_{obs}	Number of past time instants for observation
T_{pred}	Number of future time instants for prediction
$\boldsymbol{I_{t=1:T_{\text{Obs}}}}$	Sequence of past input frame images
X_t^a	Observed coordinates for agent a at time t
$\hat{Y}_t^a$	Predicted coordinates for agent a at time t
A	Number of target agents
e_{ij}	Edge weight in graph G between nodes i and j
$d(v_i, v_j)$	Distance between agents v_i and v_j
$w(t)$	Weight function in weighted-penalty strategy
w_{start}	Start weight in weighted-penalty strategy
w_{end}	End weight in weighted-penalty strategy
$\boldsymbol{\Psi}_{\text{Scene}}$	Latent representation of scene(past frame) obtained from U-Net encoder
$\boldsymbol{\Phi}_{\text{Scene}}$	Scene-aware embeddings
$\boldsymbol{T}_{\text{Scene-aware}}$	Scene-aware Transformer encoder
$\boldsymbol{\Upsilon}_{\text{Encoder}}$	U-Net Encoder
$\boldsymbol{\Gamma}_{\text{Scene}}$	Multi-layer Perceptron layer for Scene embeddings
$\boldsymbol{\Phi}_{\text{Temporal}}$	Temporal encoding
$\boldsymbol{\Gamma}_{\text{Spatial}}$	Multi-layer Perceptron layer for Spatial embeddings
$\boldsymbol{\Phi}_{\text{Spatial}}$	Spatial embeddings
$\boldsymbol{\Gamma}_{\text{Social}}$	Multi-layer Perceptron layer for Social embeddings
$\boldsymbol{\Phi}_{\text{Social}}$	Social Embeddings
$\boldsymbol{T}_{\text{Agent-aware}}$	Agent-aware Transformer encoder
$\boldsymbol{\Phi}_{\text{Agents}}$	Agent-aware embeddings
$L_{\text{weighted}}(\hat{Y}, Y)$	Weighted-penalty Loss Function