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).

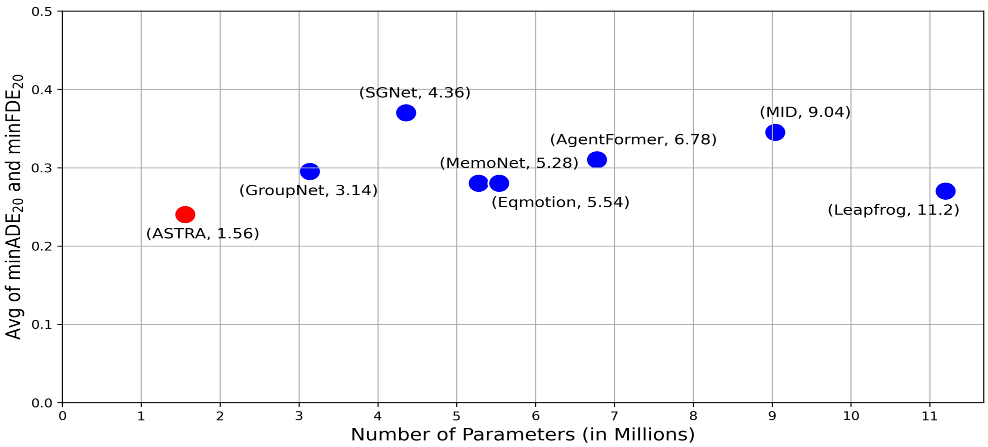


Figure 1: Comparison of average (minADE_{20}/minFDE_{20}) against the number of parameters for various models on the ETH-UCY dataset. Each point represents a different model, with the model name and number of parameters in millions indicated. Our model, ASTRA, achieves state-of-the-art results with the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

1 Introduction

Motivation & Importance 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 applications, 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) or Ego-Vehicle View (EVV) perspectives. This demands a comprehensive understanding of the scene, in addition to spatial, temporal, and social aspects that govern human movement and interaction.

Limitations in Existing Models 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 (1; 2). 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 (3) and Graph Neural Networks (4) (GNNs) have been explored, with GNN emerging as the most effective (5). Some researchers have integrated both transformers and GNN, either sequentially or in parallel, to refine the prediction paradigm (6; 7; 8; 9). 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. (10) and (11) 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 (12). This narrow focus becomes particularly problematic in, e.g., unstructured environments where a BEV might not be available, limiting the applicability of these methods.

Contributions of ASTRA 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 (13), 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 (discussed in detail in section 4.5). This is in contrast to (14) 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 ETH (15), UCY (16), and the PIE dataset (17). 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), FLOPs (Floating Point Operations), inference running time & MACs (Multiply-Accumulate Operations) than the existing competing state-of-the-art model (18; 14).

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 to generating Agent-Scene aware embedding for improved prediction accuracy, ensuring informed inter-agent interaction capture.

Paper Organisation The paper is structured as follows: Section 2 reviews related work in trajectory prediction. Sections 3 and 4 defines the problem and details ASTRA's architecture and key innovations, respectively. Section 5 describes the experimental setup. Section 6 presents empirical findings and performance comparisons. Finally, Section 7 summarizes contributions, limitations, and future directions.

2 Related Work

2.1 Stochastic vs. Deterministic Approaches in Trajectory Prediction

The trajectory prediction problem is usually addressed in two ways: stochastic (multi-modal) predictions (18; 14; 19; 12) and deterministic (uni-modal) predictions (20; 21; 22). A deterministic approach assumes a fixed future and produces only a single, most probable trajectory per input motion, making it suitable for scenarios where uncertainty is minimal or a single best estimate is required. In contrast, a stochastic approach acknowledges uncertainty in future motion and generates multiple possible trajectories for the same input, capturing the inherent variability in human movement. Stochastic approaches utilizes generative techniques like Conditional Variational Auto-Encoders (CVAEs) (14; 12), Generative Adversarial Networks (GANs) (23), Normalizing Flows (24), or Denoising Diffusion Probabilistic Models (18) to introduce randomness into the prediction process, thereby generating diverse future trajectories that better represent the distribution of possible pedestrian movements. ASTRA, like some prior works (5; 12; 25), supports both deterministic and stochastic predictions.

2.2 Social & Scene-aware Modelling

Approaches to Agent-Agent Interaction Modelling 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 (3). 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 (5; 26; 27). 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 (28; 29; 25; 30). In a similar vein, (14; 25) proposes sparse, directional graphs, predicated on the premise that different agent types possess varying perceptual ranges. Regarding the optimal depth of GNN layers, (31) advocate for deeper graphs to enhance performance. This stands in contrast to the findings of (30) and (32), 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.

Scene Embeddings 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. (10) and (11) 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) (28; 33). Some research, instead, adopts Hidden Markov Models (HMMs) for temporal analysis (34). For scenarios necessitating extended time horizon considerations, more intricate structures are proposed, including Transformers (14; 5; 35; 36) and various forms of Recurrent Neural Networks (RNNs) (29), including Long Short-Term Memory networks (LSTMs) (2; 37; 28) and Gated Recurrent Units (GRUs) (26). 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 (38) and CHAMELEON SQUIRREL (39). Direct application of graph-aware transformers remains untouched in pedestrian trajectory forecasting, with prevalent methodologies leaning towards transformers processing embeddings emanating from graphs (6; 7; 8). 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 (40). 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 Problem Formulation

The objective of trajectory prediction is to forecast a pedestrian's future position based on their observed historical trajectory. Formally, the historical trajectory of A target agents is given as a sequence of coordinates:

$$\boldsymbol{X} = \{X_t^a \mid t \in (1, 2, \ldots, T_{obs});\ a \in (1, 2, \ldots, A)\}, \tag{1}$$

where X_t^a represents the position of agent a at time t over the past T_{obs} time steps. For bird's-eye view (BEV) datasets , X_t^a consists of 2D coordinates $\{x_t^a, y_t^a\}$. For egocentric view (EVV) datasets , it consists of bounding box coordinates $\{x_{1,t}^a, y_{1,t}^a, x_{2,t}^a, y_{2,t}^a\}$. Additionally, the corresponding visual input frames/images are given as $\boldsymbol{I} = \{I_t \mid t \in (1, 2, \ldots, T_{obs})\}$.

Deterministic Setting The goal of ASTRA is to predict deterministic or multi-modal future trajectories for the pedestrians. In the deterministic setting, the model predicts the future trajectory coordinates for the A agents over the next T_{pred} time steps:

$$\hat{\boldsymbol{Y}} = \{\hat{Y}_t^a \mid t \in (1, 2, \ldots, T_{pred});\ a \in (1, 2, \ldots, A)\}, \tag{2}$$

where $\hat{Y}_t^a$ represents the predicted position of agent a at future time t, and the ground truth trajectory is denoted as $\boldsymbol{Y} = \{Y_t^a \mid t \in (1, 2, \ldots, T_{pred});\ a \in (1, 2, \ldots, A)\}$.

Stochastic Setting In the stochastic setting, the goal is to learn a generative model parameterized by θ as $p_\theta(\mathcal{Y}|\boldsymbol{X}, \boldsymbol{I})$, which, given the historical observations $\boldsymbol{X}$ and $\boldsymbol{I}$, generates K possible future trajectories:

$$\mathcal{Y} = \{\hat{\mathbf{Y}}^{(1)}, \hat{\mathbf{Y}}^{(2)}, \ldots, \hat{\mathbf{Y}}^{(K)}\}. \tag{3}$$

This enables the model to capture the inherent uncertainty in pedestrian motion by producing diverse trajectory hypotheses.

4 ASTRA Model

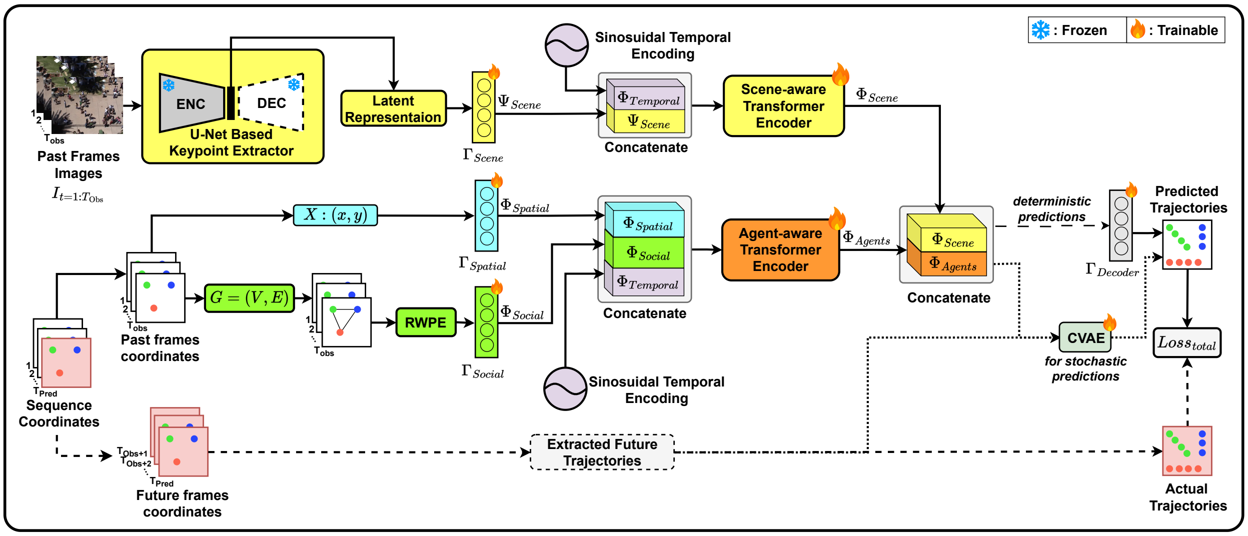


Figure 2: **Model Architecture.** Overview of ASTRA model architecture for pedestrian trajectory forecasting.

4.1 High-Level Overview

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 2. 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.

4.2 Scene-aware Transformer Encoder

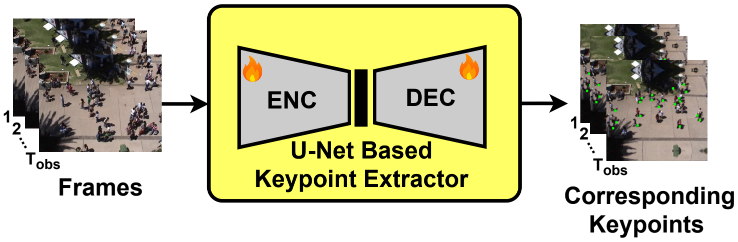
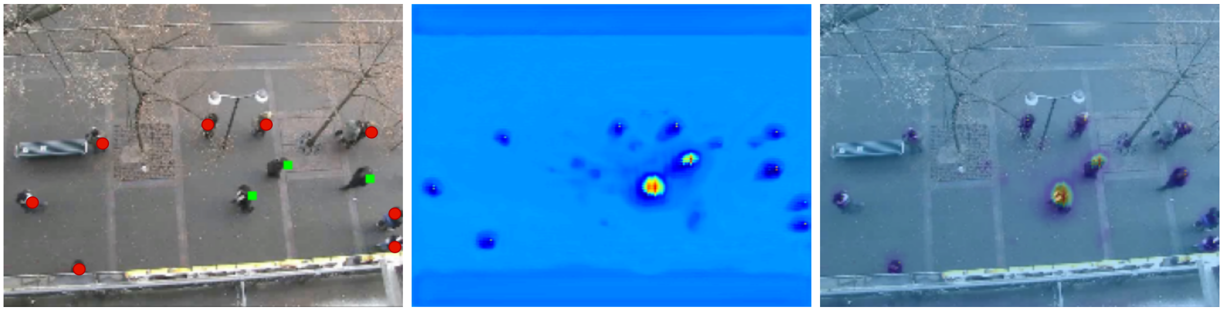


Figure 3: Pretraining U-Net based keypoint extractor.

In order to learn essential information about the scene's spatial layout and the positional dynamics of agents within it, the U-Net (41) is pre-trained to predict the location of pedestrians in the frame using the method detailed in (13), which utilizes a specialized loss function, Weighted Hausdorff Distance. This, in turn, helps the model learn a representation of the scene context (Figure 3), focusing on the pedestrians in the scene. This method of pre-training method allows the extraction of essential scene features involving pedestrians —since keypoints or locations of the pedestrians are always available in form of trajectories —without relying on explicit segmentation map annotations. This not only alleviates the need for labor-intensive annotation processes but also enables more efficient training on datasets where only pedestrian locations are provided.

A latent representation of pedestrian characteristics is obtained using a pre-trained U-Net encoder (Figure 2); this latent vector can include some crucial characteristics like spatial groupings and interactions with the environment. The Grad-CAM visualizations (Figure 4) highlight this capability, showing that the pre-trained model pays attention to regions with a high likelihood of pedestrian activity. 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 4a.

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 U-Net-based keypoint extractor is frozen after the pretraining (Figure 2) when used in ASTRA model architecture.



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

Figure 4: **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.

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:

$$\Psi_{\text{Scene}} = \Gamma_{\text{Scene}}\left(\Upsilon_{\text{Encoder}}(\boldsymbol{I})\right) \tag{4}$$

$$\Phi_{\text{Scene}} = T_{\text{Scene-aware}}\left([\Psi_{\text{Scene}}; \Phi_{\text{Temporal}}]\right) \tag{5}$$

where the latent representation of past input frame images (I) 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 (42) and follows the work of (14).

4.3 Agent-aware Transformer Encoder

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

Spatial Dimension The spatial coordinates $\mathbf{X}$ of each agent are linearly projected using an MLP layer (Γ_{Spatial}) to obtain the spatial encoding (Φ_{Spatial}). The spatial encoding belongs to $\mathbb{R}^{A \times D_{\text{S}}}$, where A is the number of agents and D_{Sp} is the spatial feature dimension.

$$\Phi_{\text{Spatial}} = \Gamma_{\text{Spatial}}(\boldsymbol{X}) \in \mathbb{R}^{A \times D_{\text{Sp}}} \tag{6}$$

Temporal Dimension Relying solely on spatial embeddings is insufficient, as agents occupying the same location in different frames would have identical spatial representations. To address this, we incorporate temporal encoding to distinguish agents across time.

To model temporal dependencies within pedestrian trajectories, we introduce temporal encodings for both agents and the scene. These encodings guide the network in capturing sequential information, similar to positional encodings in the Transformer architecture (42). The temporal encoding Φ_{Temporal} belongs to $\mathbb{R}^{A \times D_{\text{T}}}$, where A is the number of agents and D_{T} is the temporal feature dimension.

$$\Phi_{\text{Temporal}}(t, i) = \begin{cases} \sin\left(\frac{t}{10000^{2i/D_{\text{T}}}}\right) & \text{if } i \text{ is even,} \\ \cos\left(\frac{t}{10000^{2i/D_{\text{T}}}}\right) & \text{if } i \text{ is odd.} \end{cases} \in \mathbb{R}^{A \times D_{\text{T}}} \tag{7}$$

Social Dimension 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

$$e_{ij} = \frac{1}{d(v_i, v_j)} \tag{8}$$

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

Subsequently, Random Walk Positional Encodings (RWPEs) (43) are used to capture the structural relationships between nodes in the graph, such as their proximity and connectivity patterns. RWPE leverages random walks to encode positional information, allowing the model to incorporate relational dependencies between agents in a data-driven manner. By considering the number of paths and transition probabilities between nodes, RWPE provides a meaningful representation of spatial interactions in a dynamic environment.

These RWPEs are then projected using a separate MLP (Γ_{Social}) to obtain social encodings (Φ_{Social}). The social encoding belongs to $\mathbb{R}^{A \times D_{\text{So}}}$, where A is the number of agents and D_{So} is the social feature dimension:

$$\Phi_{\text{Social}} = \Gamma_{\text{Social}}\left(\text{RWPE}(\mathbf{G})\right) \in \mathbb{R}^{A \times D_{\text{So}}} \tag{9}$$

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.

Aggregating After computing the spatial, temporal, and social representations for each agent, our model concatenates them along the agent dimension. This ensures that each agent has a single feature vector consisting of a timestamp (temporal encoding), a social feature vector, and a spatial feature vector. The concatenated representation is then processed by an agent-aware single-layer transformer encoder ($T_{\text{Agent-aware}}$), which generates the final agent-aware embedding (Φ_{Agents}).

$$\Phi_{\text{Agents}} = T_{\text{Agent-aware}}\left([\Phi_{\text{Spatial}}; \Phi_{\text{Temporal}}; \Phi_{\text{Social}}]\right) \in \mathbb{R}^{A\times(D_{\text{Sp}}+D_{\text{T}}+D_{\text{So}})} \tag{10}$$

4.4 Decoder

Stochastic Decoding To generate multiple stochastic trajectories, we learn a generative model, $p_\theta(\mathcal{Y}|\boldsymbol{X}, \boldsymbol{I})$ for which we adopted CVAE (Conditional Variational Auto Encoder). We train CVAE, similar to (14),(12), to learn the inherent distribution of future target trajectories conditioned on observed past trajectories, by utilizing a latent variable Z. CVAE consists of three components - prior network ($p_\theta(Z|\tilde{X})$), recognition network (or posterior) ($q_\phi(Z|Y, X)$) and generation network ($g_\nu(\hat{Y}|Z)$), parameterized by θ, ϕ and ν respectively. Here $\tilde{X}$ is the latent representation of X and I, obtained after concatenating Φ_{Agents} and Φ_{Scene} and $\hat{Y}$ is the output of the generation network and are the predicted future trajectories.

For CVAE, prior distribution ($p_\theta(z|\tilde{X})$) is parameterized by $\mathcal{N}(\mu_z^p, (\sigma_z^p)^2)$. The approximate posterior distribution ($q_\phi(z|Y, X)$) is parameterized by $\mathcal{N}(\mu_z^q, (\sigma_z^q)^2)$, where μ_z^p and $(\sigma_z^p)^2$ represent the mean and variance of the prior distribution and μ_z^q and $(\sigma_z^q)^2$ represent the mean and variance of the posterior distribution. During Training, we sample latent variable (z) from the recognition network (posterior distribution) and fed it to the generation network (($g_\nu(\hat{Y}|Z)$), whereas during testing we sample z from the prior network (prior distribution). We use the reparameterization trick to sample z through the mean and variance pairs of ($\mu_z^p, (\sigma_z^p)^2$) and ($\mu_z^q, (\sigma_z^q)^2$), respectively. KL divergence Loss ($\mathcal{L}_{\text{KL}}$) help in minimizing the difference between the distribution of latent variable(z) of prior and recognition network.

K samples are drawn from the learned distribution and decoded using an MLP to obtain future trajectories. We optimize the parameters of the networks using the KL divergence which ensures that the prior network implicitly learns the dependency between future trajectories ($\boldsymbol{Y}$) and past trajectories ($\boldsymbol{X}$)

$$\mathcal{L}_{\text{KL}} = D_{KL}\big(q_\phi(Z \mid Y, X) \,\|\, p_\theta(Z \mid \tilde{X})\big) = D_{KL}\big(\mathcal{N}(\mu_{z_q}, (\sigma_z^q)^2) \,\|\, \mathcal{N}(\mu_{z_p}, (\sigma_{z_p})^2)\big). \tag{11}$$

Deterministic Decoding 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 2, namely:

$$\hat{\mathbf{Y}} = \Gamma_{\text{Decoder}}([\Phi_{\text{Scene}}; \Phi_{\text{Agents}}]) \tag{12}$$

4.5 Weighted 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:

$$L_{\text{weighted}}(\hat{Y}, Y) = \sum_{t=1}^{T_{pred}} w(t) \cdot L(\hat{Y}_t, Y_t), \tag{13}$$

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.

In time series data, as we move further into the future relative to the last observed data, the drift in predictions tends to increase, leading to higher errors. Motivated by this intuition, we initially penalized the predictions using linear and quadratic loss functions. These approaches showed improvements in overall prediction accuracy. However, upon closer analysis of the results, we observed that in some trajectories, there was a noticeable offset in the earlier parts of the predictions. To address this issue, we experimented with a parabolic weighting function for the penalty. Empirically, this approach outperformed the linear and quadratic strategies, yielding the most balanced and accurate predictions across the trajectories.

The weight function $w(t)$ (generally defined as $w(t) = f(t, T_{\text{pred}}, \alpha, \beta)$) is designed to be versatile, accommodating a spectrum of mathematical formulations that align with the specific needs of the predictive model. We used the parabolic weighted penalty, defined as:

$$w(t) = (\alpha - \beta) \cdot \left(2 \cdot \frac{t}{T_{pred}} - 1\right)^2 + \beta = 3\left(2 \cdot \frac{t}{T_{\text{pred}}} - 1\right)^2 + 1 \tag{14}$$

where α and β are parameters that establish the bounds of the weighting function (during our experimentations and hyperparameter tuning, we found that $\alpha = 4$ and $\beta = 1$, as the optimal values), 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 (discussed more in 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.

The final loss is the weighted and the stochastic, the latter is removed for the deterministic model.

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) + \mathcal{L}_{\text{KL}} \tag{15}$$

5 Experiments

5.1 Datasets & Evaluation Protocols

For a comprehensive evaluation, we benchmarked our model on three trajectory prediction datasets; namely, ETH (15), UCY (16), and PIE dataset (17).

ETH-UCY (Bird's Eye View) 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).

PIE (Ego-Vehicle View) 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 (17). A total of 1,842 pedestrian samples are considered with the following split: Training(50%), Validation(40%) and Testing(10%)(17). 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 scenarios(2).

5.2 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. CADE, CFDE, ARB and FRB for PIE dataset. The supplementary material explains these metrics.

5.3 Hyperparameter Tuning & Hardware Settings

The hyperparameters were initially tuned on the ETH subset of the ETH-UCY dataset and subsequently applied across all subsets. The key architectural hyperparameters used in our model are as follows: spatial embedding dimension ($\Phi_{\text{Spatial}} \in \mathbb{R}^{16}$), U-Net scene latent representation ($\Psi_{\text{Scene}} \in \mathbb{R}^{16}$), temporal embedding dimension ($\Phi_{\text{Temporal}} \in \mathbb{R}^{8}$), and random walk embedding ($\Phi_{\text{Social}} \in \mathbb{R}^{8}$). The transformer encoder consists of a single layer with two attention heads and a dropout rate of 0.2. To keep the model lightweight, MobileNet v2 (44) encoder layers are used for U-Net to reduce computational overhead while maintaining feature extraction efficiency.

For training, we employ the AdamW optimizer with a weight decay of 5×10^{-4} over 200 epochs. A cosine annealing scheduler is used, starting with an initial learning rate of 1×10^{-3}. All experiments were conducted on an NVIDIA DGX A100 system with 8 GPUs, each equipped with 80 GB of memory.

6 Results & Discussion

6.1 Quantitative Results

ETH-UCY For ETH-UCY, we compared our model results against several baselines. These comparisons are presented in Table 1 and Table 3, which contains results primarily sourced from the EqMotion (CVPR 2023) (5) for deterministic predictions and LeapFrog (CVPR 2023) (18) for stochastic predictions respectively. It is important to note that to provide a thorough comparative framework, we independently computed and included additional models (45; 12) to their respective tables as they were not originally part of the EqMotion or LeapFrog analysis. Our model advances the state-of-the-art on ETH-UCY, outperforming the EqMotion (5) by improving predictive accuracy by approximately 27% on average for deterministic predictions as shown in Table 1 and approximately 10% on average improvement over LeapFrog(18) for stochastic predictions as shown in Table 3, highlighting the efficacy of our approach in diverse scenarios. While ASTRA maintains superior performance across most benchmarks, there are, however, some exception cases, like in the Hotel and Univ scenes, where a notable proportion of pedestrians remain largely stationary throughout both the observation and prediction windows, resulting in slightly inferior performance in these scenarios. We also highlight the effectiveness of utilizing frame encodings from U-Net, as demonstrated in Table 3.

PIE Similarly, we benchmarked our model against various established models for the PIE dataset. The comparative analysis is summarized in Table 2, with the reference results taken from the PedFormer (2), demonstrating an average improvement of 26%.

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-LSTM(22)	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-Attention(46)	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-ind(47)	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++(25)	1.02/2.00	0.33/0.62	0.53/1.19	0.44/0.99	0.32/0.73	0.53/1.11
TransF(35)	1.03/2.10	0.36/0.71	0.53/1.32	0.44/1.00	0.34/0.76	0.54/1.17
MemoNet(19)	1.00/2.08	0.35/0.67	0.55/1.19	0.46/1.00	0.37/0.82	0.55/1.15
SGNet(45)	0.81/1.60	0.41/0.87	0.58/1.24	0.37/0.79	0.31/0.68	0.56/1.04
EqMotion(5)	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**

6.2 Efficiency

Regarding the computational side, ASTRA has seven times fewer trainable parameters than the existing SOTA model LeapFrog (18) 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. The complete ASTRA architecture, including all components, comprises 1.56M parameters, 652M FLOPs (floating-point operations), and 326M MACs (Multiply-Accumulate Operations). Even in configurations without U-Net pretraining, the deterministic variant has 13.52K parameters, 15.87K FLOPs, and 7.935K MACs, while the stochastic variant has 141.48K parameters, 165.416K FLOPs, and 82.708K MACs. This flexibility allows ASTRA to scale efficiently under strict resource constraints.

Table 2: ARB, FRB, CADE & CFDE are calculated for PIE dataset. Best in **bold**, second best underlined

Model	**CADE**	**CFDE**	**ARB**	**FRB**
FOL(48)	73.87	164.53	78.16	143.69
FPL(49)	56.66	132.23	-	-
B-LSTM(50)	27.09	66.74	37.41	75.87
PIE$_{\text{traj}}$(17)	21.82	53.63	27.16	55.39
PIE$_{\text{full}}$(17)	19.50	45.27	24.40	49.09
BiPed(51)	15.21	35.03	19.62	39.12
PedFormer(2)	13.08	30.35	**15.27**	32.79
ASTRA(Ours)	**9.91**	**22.42**	18.32	**17.07**

Table 3: **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 (47)	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 (52)	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 (53)	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 (54)	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++ (25)	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 (12)	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 (19)	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 (55)	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 (45)	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 (56)	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 (14)	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 (5)	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 (18)	0.39/0.58	**0.11**/**0.17**	0.26/0.43	0.18/0.26	0.13/0.22	0.21/0.33
TrajFine (57)	0.35/0.60	**0.11**/0.18	**0.22**/0.48	**0.15**/0.30	**0.12**/0.25	**0.19**/0.36
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 Image Input)	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 Image Input)	**0.27**/**0.36**	0.17/0.25	0.28/**0.41**	**0.15**/**0.23**	0.13/**0.16**	0.20/**0.28**

Additionally, we computed the model's running time, which takes 19.23ms to generate predictions for future trajectories. To further assess ASTRA's efficiency, we compare its running time, FLOPs, MACs, and number of parameters with three other models: AgentFormer, a transformer-based architecture similar to ours; Leapfrog, a state-of-the-art approach; and GroupNet, which has the fewest parameters among all compared models. As shown in the Table 4, ASTRA demonstrates strong efficiency compared to these alternatives.

Table 4: Model Parameters and Inference Time comparison. (**M** denotes Million and **ms** denotes Millisecond)

Model	**Parameters (M)**	**FLOPS (M)**	**MACs (M)**	**Inference Time (ms)**
GroupNet	3.14	806	403	105.78
Agentformer	6.78	3084	1542	521.34
Leapfrog	11.2	5695	2848	28.91
Astra (Ours)	1.56	652	326	19.23

6.3 Ablations

In the ablation study presented in Table 5, 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 material) 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 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.

We also investigate the effect of using a Single Transformer Encoder versus Dual Transformer Encoders (Table 5, last two rows). For the Single Encoder setup, we merge $T_{\text{Scene-aware}}$ and $T_{\text{Agent-aware}}$ into a unified transformer encoder. While this configuration captures a joint representation of scene features and agent dynamics, it inherently limits the model performance, as can be inferred from Table 5.

Table 5: **Ablation:** ADE/FDE & minADE$_{20}$/minFDE$_{20}$ with variations in ASTRA's model components on UNIV dataset (where ✓: Component enabled, ×: Component disabled).

Spatial	**Temporal**	**Augmentation**	**Social**	**U-Net Features**	**Transformer Encoder**	**ADE/FDE**	**minADE$_{20}$/ minFDE$_{20}$**
✓	×	×	×	×	Single	1.05/1.66	0.43/0.63
✓	✓	×	×	×	Single	0.86/1.47	0.39/0.54
✓	✓	✓	×	×	Single	0.67/1.17	0.31/0.48
✓	✓	✓	✓	×	Single	0.66/1.12	0.29/0.43
✓	✓	✓	✓	✓	**Single**	0.74/0.61	0.98/1.41
✓	✓	✓	✓	✓	**Dual**	0.55/1.00	0.28/0.41

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 6 and in Table 3 (ASTRA(NP)) for deterministic and stochastic setting respectively and the same can be observed in Figure 7. Additionally, we compared the CLIP encoder —with added linear layer for fine-tuning in an end-to-end fashion —with our U-Net-based encoder, as shown in Table 7. Our model consistently performed on par with or better than CLIP while using significantly fewer parameters, MACs, and FLOPs, highlighting its efficiency. This can be attributed to the pre-training, which enables the model to focus specifically on pedestrians.

Table 6: **Ablation:** ADE/FDE for penalised vs. unpenalised loss functions on UNIV dataset using SOTA ASTRA's configuration

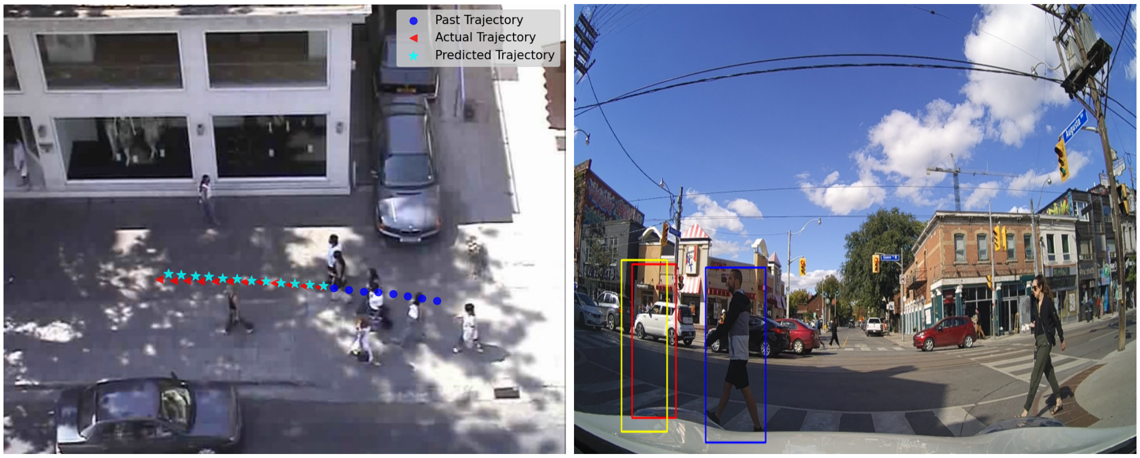
Loss	**Normal**	**Penalised**
MSE	0.58/1.13	0.57/1.00
SmoothL1	0.57/1.15	0.55/1.00

Table 7: Efficiency and quantitative (minADE$_{20}$ and minFDE$_{20}$) results for ETH-UCY baselines for different image encoders. (Million denotes **M**)

Encoder	**Parameter Count (M)**	**FLOPs (M)**	**MACs (M)**	**ETH**	**Hotel**	**Univ**	**Zara1**	**Zara2**	**Average**
CLIP (58)	149.77	22540	11270	0.26/0.37	0.18/0.28	0.34/0.48	0.18/0.29	0.15/0.21	0.22/0.324
UNET (Ours)	1.56	652	326	0.27/0.36	0.17/0.25	0.28/0.41	0.15/0.23	0.13/0.16	0.2/0.282

6.4 Qualitative Analysis

We can clearly see the results of our prediction from Figure 5 for deterministic predictions and Figure 6 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.



(a) BEV

(b) EVV

Figure 5: 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.

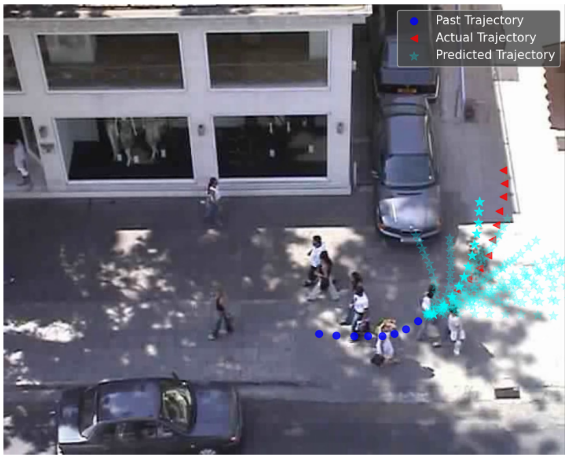


Figure 6: Sample images of the stochastic predictions from ETH-UCY Dataset.

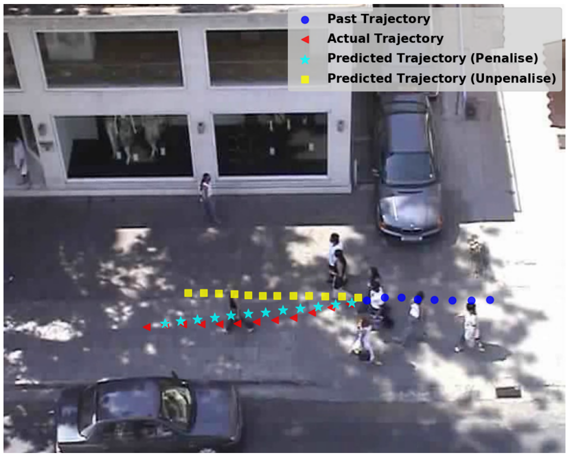

Past Trajectory
Actual Trajectory
Predicted Trajectory (Penalise)
Predicted Trajectory (Unpenalise)

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

7 Conclusion & Future Work

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

[1] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, “Transformer networks for trajectory forecasting,” in *2020 25th international conference on pattern recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[2] A. Rasouli and I. Kotseruba, “Pedformer: Pedestrian behavior prediction via cross-modal attention modulation and gated multitask learning,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9844–9851.

[3] B. Pang, T. Zhao, X. Xie, and Y. N. Wu, “Trajectory prediction with latent belief energy-based model,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11 809–11 819.

[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[5] C. Xu, R. T. Tan, Y. Tan, S. Chen, Y. G. Wang, X. Wang, and Y. Wang, “Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1410–1420.

[6] L. Li, M. Pagnucco, and Y. 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*, 2022, pp. 2231–2241.

[7] X. Chen, H. Zhang, Y. Hu, J. Liang, and H. Wang, “Vnagt: Variational non-autoregressive graph transformer network for multi-agent trajectory prediction,” *IEEE Transactions on Vehicular Technology*, pp. 1–12, 2023.

[8] X. Jia, P. Wu, L. Chen, Y. Liu, H. Li, and J. Yan, “Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding,” *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[9] Y. Liu, L. Yao, B. Li, X. Wang, and C. 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*, ser. CIKM ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1339–1349. [Online]. Available: https://doi.org/10.1145/3511808.3557455

[10] A. Rasouli, M. Rohani, and J. Luo, “Bifold and semantic reasoning for pedestrian behavior prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 600–15 610.

[11] K. Mangalam, Y. An, H. Girase, and J. Malik, “From goals, waypoints & paths to long term human trajectory forecasting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 233–15 242.

[12] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, “Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1463–1470, 2021.

[13] J. Ribera, D. Guera, Y. Chen, and E. J. Delp, “Locating objects without bounding boxes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6479–6489.

[14] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, “Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[15] 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*, 2009, pp. 261–268.

[16] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x

[17] A. Rasouli, I. Kotseruba, T. Kunic, and J. 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)*, 2019, pp. 6261–6270.

[18] W. Mao, C. Xu, Q. Zhu, S. Chen, and Y. Wang, "Leapfrog diffusion model for stochastic trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 5517–5526.

[19] C. Xu, W. Mao, W. Zhang, and S. Chen, "Remember intentions: Retrospective-memory-based trajectory prediction," pp. 6488–6497, June 2022.

[20] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[21] 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.* IEEE, 2009, pp. 261–268.

[22] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] L. Huang, J. Zhuang, X. Cheng, R. Xu, and H. Ma, "Sti-gan: Multimodal pedestrian trajectory prediction using spatiotemporal interactions and a generative adversarial network," *IEEE Access*, vol. 9, pp. 50 846–50 856, 2021.

[24] A. Bhattacharyya, C. Straehle, M. Fritz, and B. Schiele, "Haar wavelet based block autoregressive flows for trajectories," *CoRR*, vol. abs/2009.09878, 2020. [Online]. Available: https://arxiv.org/abs/2009.09878

[25] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. 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.* Springer, 2020, pp. 683–700.

[26] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," *arXiv preprint arXiv:2109.01827*, 2021.

[27] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. D. Reid, S. H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *NeurIPS*, 2019.

[28] J. Fang, D. Yan, J. Qiao, J. Xue, and H. Yu, "Dada: Driver attention prediction in driving accident scenarios," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[29] H. Girase, H. Gang, S. Malla, J. Li, A. Kanehara, K. Mangalam, and C. Choi, "Loki: Long term and key intentions for trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9803–9812.

[30] X. Weng, Y. Yuan, and K. Kitani, "Ptp: Parallelized tracking and prediction with graph neural networks and diversity sampling," *IEEE Robotics and Automation Letters*, vol. 6, pp. 4640–4647, 7 2021.

[31] R. Addanki, P. W. Battaglia, D. Budden, A. Deac, J. Godwin, T. Keck, W. L. S. Li, A. Sanchez-Gonzalez, J. Stott, S. Thakoor *et al.*, "Large-scale graph representation learning with very deep gnns and self-supervision," *arXiv preprint arXiv:2107.09422*, 2021.

[32] B. Liu, E. Adeli, Z. Cao, K.-H. Lee, A. Shenoi, A. Gaidon, and J. C. Niebles, “Spatiotemporal relationship reasoning for pedestrian intent prediction,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3485–3492, 2020.

[33] I. Kotseruba, A. Rasouli, and J. K. Tsotsos, “Benchmark for evaluating pedestrian action prediction,” in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1257–1267.

[34] W. Cai, G. He, J. Hu, H. Zhao, Y. Wang, and B. 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.

[35] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, “Transformer networks for trajectory forecasting,” in *2020 25th international conference on pattern recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[36] W. Chen, F. Wang, and H. Sun, “S2tnet: Spatio-temporal transformer networks for trajectory prediction in autonomous driving,” in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 454–469.

[37] A. Bhattacharyya, D. O. Reino, M. Fritz, and B. 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.

[38] J. Tang, J. Sun, C. Wang, and Z. Yang, “Social influence analysis in large-scale networks,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 807–816. [Online]. Available: https://doi.org/10.1145/1557019.1557108

[39] B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node embedding,” *Journal of Complex Networks*, vol. 9, no. 2, p. cnab014, 2021.

[40] L. Müller, M. Galkin, C. Morris, and L. Rampášek, “Attending to graph transformers,” *arXiv preprint arXiv:2302.04181*, 2023.

[41] O. Ronneberger, P. Fischer, and T. 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*. Springer, 2015, pp. 234–241.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[43] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, “Graph neural networks with learnable structural and positional representations,” *arXiv preprint arXiv:2110.07875*, 2021.

[44] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[45] C. Wang, Y. Wang, M. Xu, and D. J. Crandall, “Stepwise goal-driven networks for trajectory prediction,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2716–2723, 2022.

[46] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4601–4607.

[47] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[48] Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush, “Egocentric vision-based future vehicle localization for intelligent driving assistance systems,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9711–9717.

[49] T. Yagi, K. Mangalam, R. Yonetani, and Y. Sato, “Future person localization in first-person videos,” 2018.

[50] A. Bhattacharyya, M. Fritz, and B. Schiele, “Long-term on-board prediction of people in traffic scenes under uncertainty,” 2018.

[51] A. Rasouli, M. Rohani, and J. Luo, “Pedestrian behavior prediction via multitask learning and categorical interaction modeling,” *CoRR*, vol. abs/2012.03298, 2020. [Online]. Available: https://arxiv.org/abs/2012.03298

[52] Y. Hu, S. Chen, Y. Zhang, and X. Gu, “Collaborative motion prediction via neural motion message passing,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6319–6328.

[53] C. Yu, X. Ma, J. Ren, H. Zhao, and S. 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*. Springer, 2020, pp. 507–523.

[54] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. 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*. Springer, 2020, pp. 759–776.

[55] C. Xu, M. Li, Z. Ni, Y. Zhang, and S. 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)*, June 2022, pp. 6498–6507.

[56] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu, “Stochastic trajectory prediction via motion indeterminacy diffusion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 113–17 122.

[57] K.-L. Wang, L.-W. Tsao, J.-C. Wu, H.-H. Shuai, and W.-H. Cheng, “Trajfine: Predicted trajectory refinement for pedestrian trajectory forecasting,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024, pp. 4483–4492.

[58] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[59] S. Zamboni, Z. T. Kefato, S. Girdzijauskas, C. Norén, and L. Dal Col, “Pedestrian trajectory prediction with convolutional neural networks,” *Pattern Recognition*, vol. 121, p. 108252, 2022.

[60] A. Mohamed, K. Qian, M. Elhoseiny, and C. 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*, 2020, pp. 14 424–14 432.

[61] C. Yu, X. Ma, J. Ren, H. Zhao, and S. 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*. Springer, 2020, pp. 507–523.

[62] K. Vinogradova, A. Dibrov, and G. Myers, “Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract),” *Proceedings of the AAAI*, vol. 34, no. 10, p. 13943–13944, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1609/aaai.v34i10.7244

[63] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

A Weighted Loss Function Details

A.1 Loss Function Formulation

The loss for multi-modal trajectories is given in equations equation 16 and equation 17. $L_{\text{weighted}}(\mathcal{Y}_k, \boldsymbol{Y})$ is calculated similar to Equation 18.

$$L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) = \min_{k=1,\dots,K} L_{\text{weighted}}(\mathcal{Y}_k, \boldsymbol{Y}) \tag{16}$$

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) + D_{KL}(\mathcal{N}(\mu_{z_q}, \sigma_{z_q}) \,||\, \mathcal{N}(\mu_{z_p}, \sigma_{z_p})) \tag{17}$$

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

$$\mathcal{L}_{\text{final}} = L_{\text{weighted}}(\hat{Y}, Y) = \sum_{t=1}^{T_{pred}} w(t) \cdot L(\hat{Y}_t, Y_t), \tag{18}$$

where $w(t)$ represent the weighted penalty function (section A.2) and $L(\hat{Y}_t, Y_t)$ is the predefined loss function: MSE or Smooth L1 loss (discussed below).

Mean square error (MSE)

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{19}$$

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)

$$\text{SL1}(y_i, \hat{y}_i) = \begin{cases} 0.5 \times (y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| < 1 \\ |y_i - \hat{y}_i| - 0.5 & \text{otherwise.} \end{cases} \tag{20}$$

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.

A.2 Weighted Penalty Function

In many trajectory prediction tasks, errors at different time steps may have varying importance. For instance, predictions further into the future might be more uncertain, whereas errors in early predictions could be more critical for subsequent decisions. To address this, we introduce a weighted penalty function $w(t)$ that adjusts the loss function's sensitivity to prediction errors over time.

Our ablation focuses on three distinct penalty functions, $w(t, T_{\text{pred}}, \alpha, \beta)$: Linear, Quadratic, and Parabolic, which are parameterized by α and β, determined empirically as shown in Figure 8. Here, t represents the current time step, and T_{pred} denotes the total prediction horizon. Once the values of α and β are determined, $w(t, T_{\text{pred}}, \alpha, \beta)$ can be simply denoted as $w(t)$.

Table 8 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 yields better results compared to the other penalization strategies.

Figure 8 illustrates a comparison of the three weighted penalty strategies for a prediction window of 12 frames. The subsequent sections provide a detailed explanation of each of these penalty strategies.

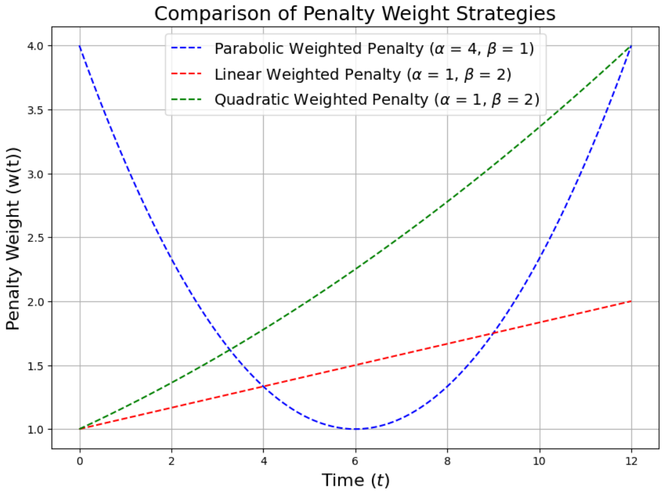


Figure 8: Comparison of various weighted penalty strategies

A.2.1 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:

$$w(t) = \alpha + \frac{t}{T_{pred}} \cdot (\beta - \alpha), \tag{21}$$

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

A.2.2 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:

$$w(t) = \left(\alpha + \frac{t}{T_{\text{pred}}} \cdot (\beta - \alpha)\right)^2 \tag{22}$$

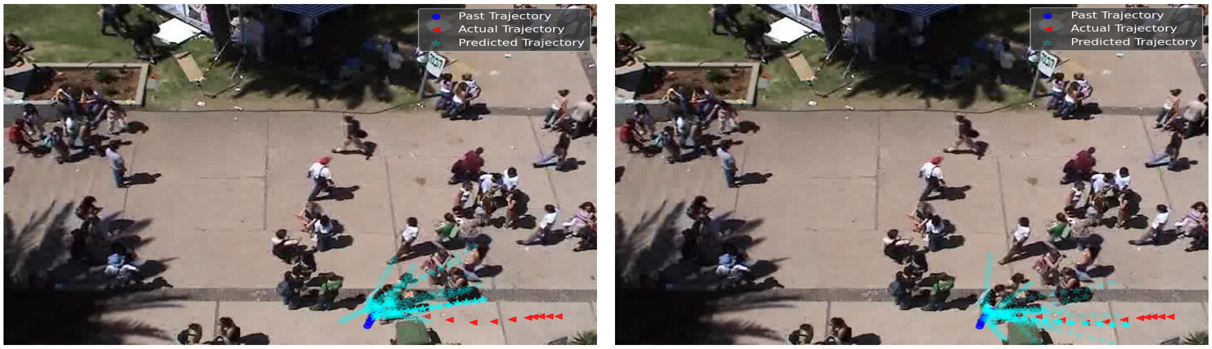
A.2.3 Parabolic Weighted Penalty

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 8) of weights across the prediction period, as defined by:

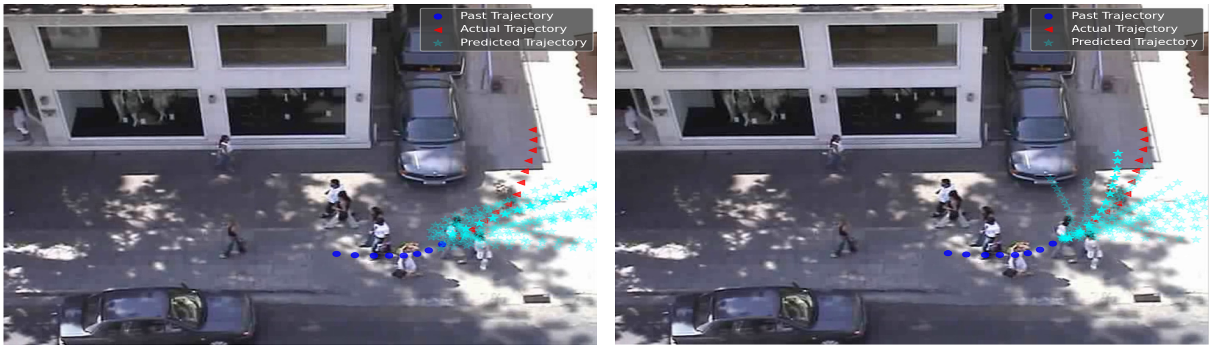
$$w(t) = (\alpha - \beta) \cdot \left(2 \cdot \frac{t}{T_{pred}} - 1\right)^2 + \beta, \tag{23}$$

A.2.4 Augmentation

To enhance the robustness and generalization of our trajectory prediction model, we implement a data augmentation strategy inspired by (59). This strategy applies random rotation and translation transformations



(a) Univ (No Penalty) (b) Univ (Penalty)



(c) Zara01 (No Penalty) (d) Zara01 (Penalty)

Figure 9: Qualitative comparison of unpenalised vs. penalised trajectories on ETH-UCY dataset in stochastic setting.

Table 8: **Ablation:** Comparing penalization strategies with SL1 loss on ETH-UCY (UNIV) dataset using ASTRA's SOTA configuration

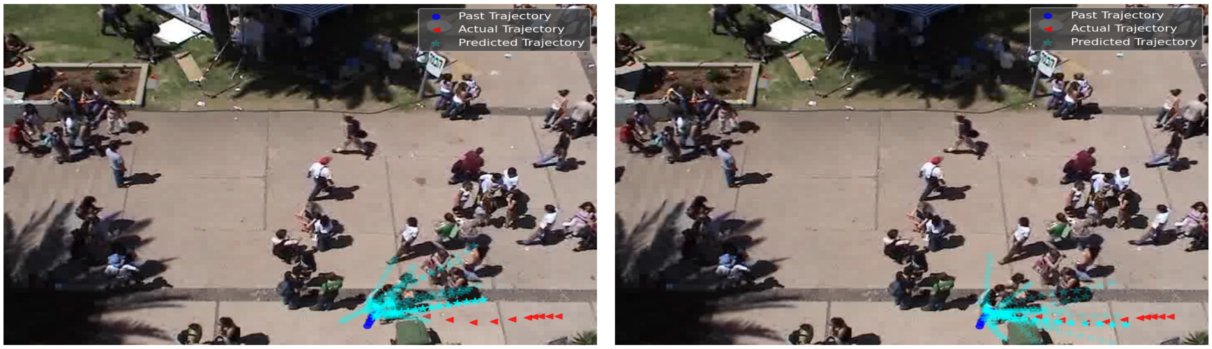
Loss	$\text{minADE}_{20}/\text{minFDE}_{20}$
Unpenalised	0.37/0.52
Linear	0.33/0.47
Quadratic	0.30/0.46
Parabolic	**0.28/0.41**

to the trajectory sequences with a probability of 0.4, as illustrated in Fig. 11. By introducing these augmentations, the model becomes orientation-agnostic and better equipped to handle positional shifts in agents. The increased diversity in training data enables the model to learn more generalized representations of agent movements, improving its adaptability to real-world scenarios.

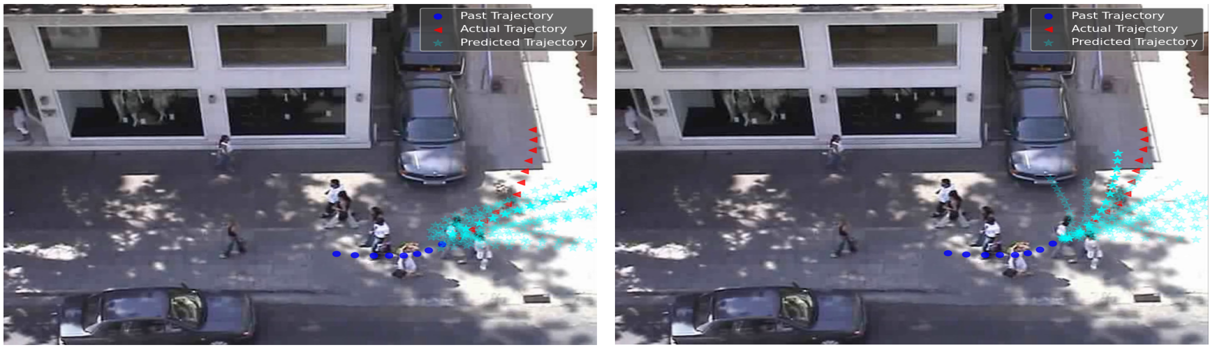
B Evaluation Metrics

B.1 ETH-UCY

To evaluate our model on ETH-UCY, we used commonly employed evaluation metrics (5; 60; 7; 61): ADE/FDE and $\text{minADE}_K/\text{minFDE}_K$. 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



(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.

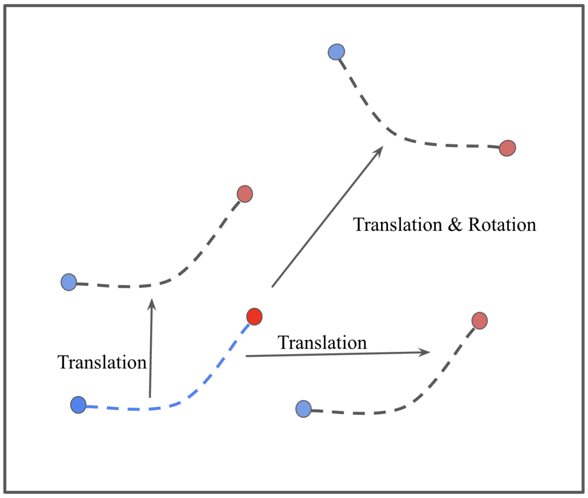


Figure 11: Illustration of data augmentation to trajectory sequences

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.

$$\text{ADE} = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \|Y_t^a - \hat{Y}_t^a\|_2. \tag{24}$$

$$\text{minADE}_K = \min_k \left(\frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \|Y_t^a - \hat{Y}_{t,k}^a\|_2 \right) \tag{25}$$

$$\text{FDE} = \|Y_{T_{pred}}^a - \hat{Y}_{T_{pred}}^a\|_2 \tag{26}$$

$$\text{minFDE}_K = \min_k \left(\|Y_{T_{pred}}^a - \hat{Y}_{T_{pred},k}^a\|_2 \right) \tag{27}$$

B.2 PIE

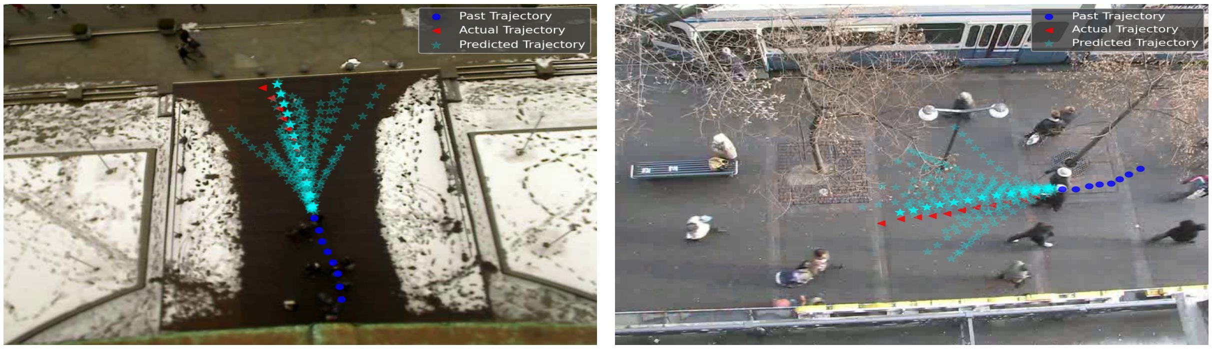
For the PIE Dataset, the ADE and FDE metrics are calculated based on the centroid of the bounding box (17; 2), denoted as Centre average displacement error for the bounding box (CADE) and Centre final displacement error for the bounding box (CFDE). In addition, we reported the average and final Root Mean Square Error (RMSE) of bounding box coordinates, denoted as ARB and FRB, respectively (10).

C Grad-CAM visualizations

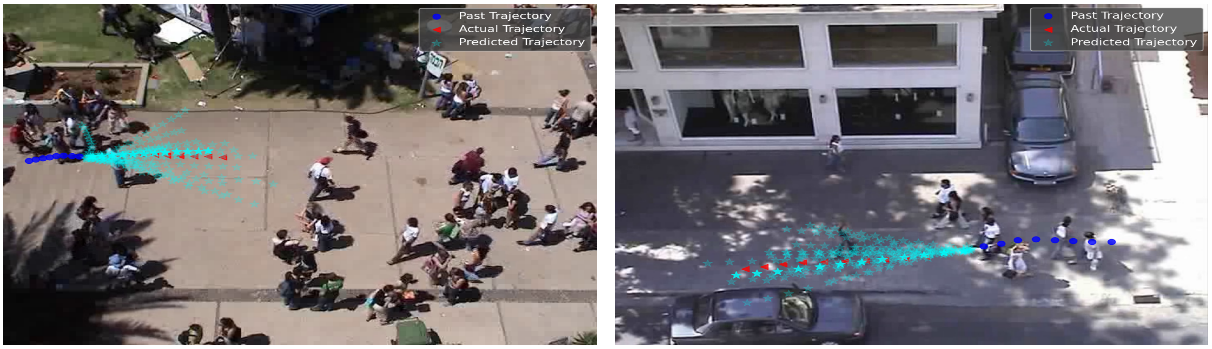
Grad-CAM images were obtained by generating heatmaps overlaid onto the original image to aid in validating the relevance of highlighted regions. To obtain the Grad-CAM visualization, a single-channel output segmentation map was obtained from the pre-trained U-Net network, representing the probability of each pixel location being a keypoint (13). 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 (62). 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 (63), 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.

D Abbreviations and Mathematical Notation

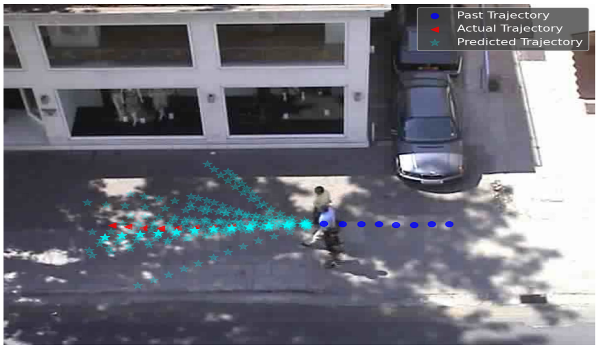
To ease reading the paper, Table 9 and 10 list the abbreviations and the mathematical symbols mentioned in the paper, respectively.



(a) ETH (b) Hotel

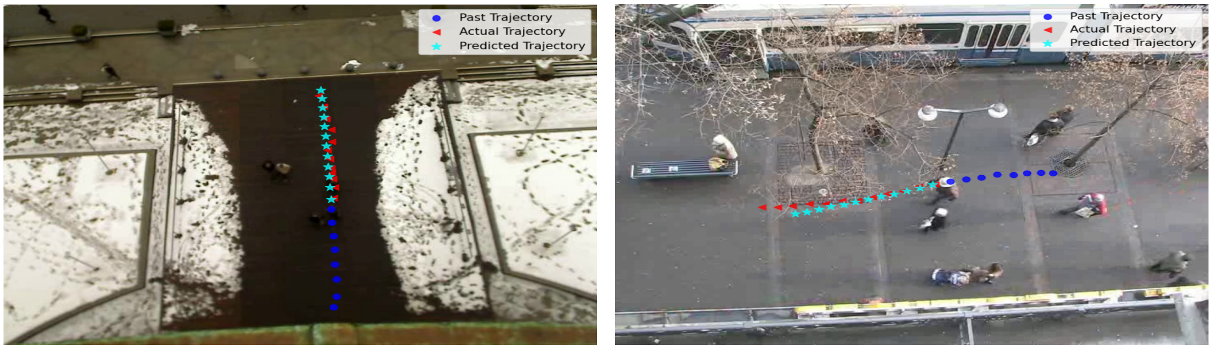


(c) Univ (d) Zara1

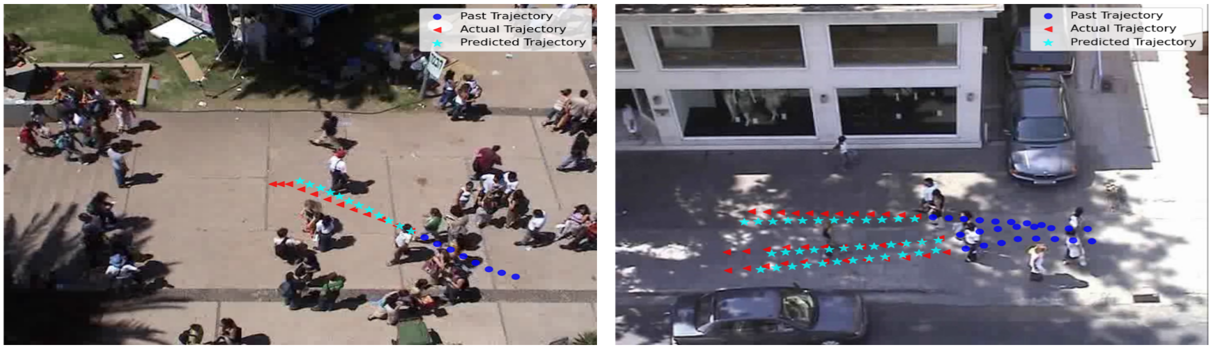


(e) Zara2

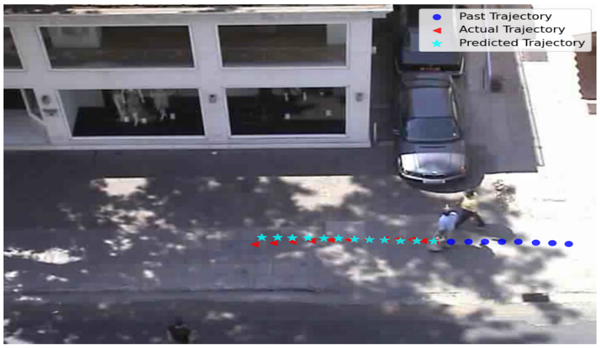
Figure 12: Multi-modal trajectory visualizations on ETH-UCY dataset (BEV)



(a) ETH (b) Hotel



(c) Univ (d) Zara1



(e) Zara2

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

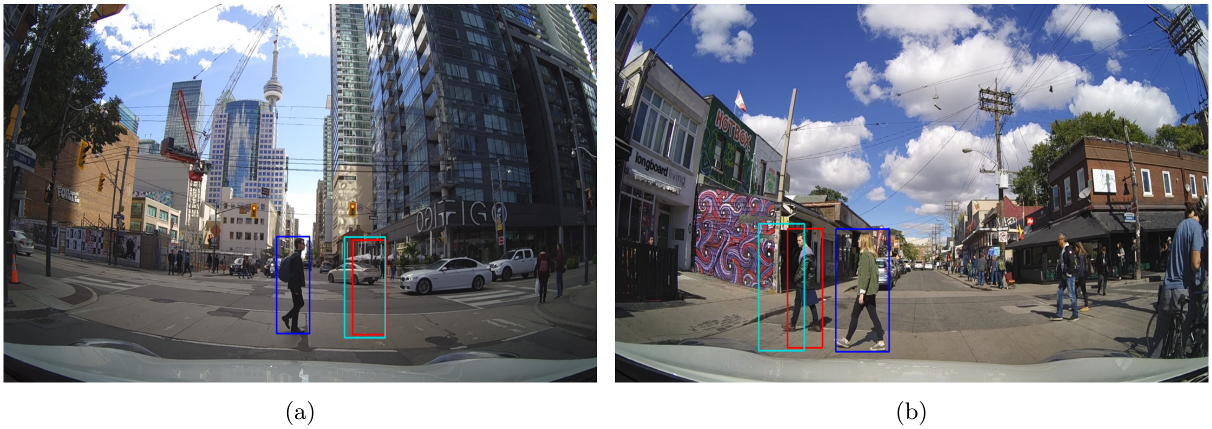

(a) (b)

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.

Table 9: 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 10: 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
$\boldsymbol{I}_{\boldsymbol{t=1:T}_{\text{Obs}}}$	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