Graph-based Spatial Transformer with Memory Replay for Multi-future Pedestrian Trajectory Prediction

Lihuan Li[1] Maurice Pagnucco[1] Yang Song[1]
[1]University of New South Wales, Australia
lihuanli80@gmail.com {morri,yang.song1}@unsw.edu.au

Abstract

Pedestrian trajectory prediction is an essential and challenging task for a variety of real-life applications such as autonomous driving and robotic motion planning. Besides generating a single future path, predicting multiple plausible future paths is becoming popular in some recent work on trajectory prediction. However, existing methods typically emphasize spatial interactions between pedestrians and surrounding areas but ignore the smoothness and temporal consistency of predictions. Our model aims to forecast multiple paths based on a historical trajectory by modeling multi-scale graph-based spatial transformers combined with a trajectory smoothing algorithm named "Memory Replay" utilizing a memory graph. Our method can comprehensively exploit the spatial information as well as correct the temporally inconsistent trajectories (e.g., sharp turns). We also propose a new evaluation metric named "Percentage of Trajectory Usage" to evaluate the comprehensiveness of diverse multi-future predictions. Our extensive experiments show that the proposed model achieves state-of-the-art performance on multi-future prediction and competitive results for single-future prediction. Code released at https://github.com/Jacobieee/ST-MR.

1. Introduction

Trajectory prediction is an indispensable part of social behavior analysis for a variety of applications including autonomous driving [5, 49], motion tracking [28, 34] and robotic systems [33]. Such tasks require a high-level understanding of videos and human social behaviors to precisely forecast the future locations of pedestrians based on the observed trajectories and scenes.

Trajectory prediction requires simultaneous processing of spatial and temporal information. While walking paths naturally exhibit a temporal consistency, it is also important to model spatial interactions among pedestrians such as talking, grouping and avoiding collisions. Other objects in a scene also affect the future paths, as pedestrians tend to avoid obstacles (e.g., street lamps, trees, vehicles) or unnecessary change of paths (e.g., walk from the pavement to the middle of the road). However, reactions to spatial interactions may also undermine the original intentions based on temporal information. Even if both of them are properly processed, it is still a conundrum to predict spatially reasonable trajectories while conforming to temporal consistency.

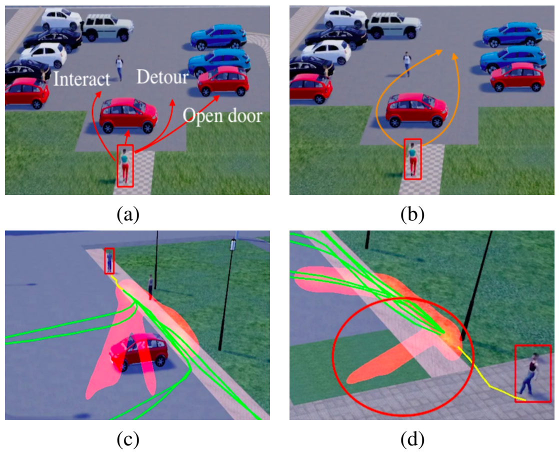


Figure 1. Illustration of multi-future trajectory prediction and existing issues. The yellow and green lines are the observed and ground truth trajectories. (a) Multiple future trajectories (red arrows) influenced by different intentions and destinations. (b) Multiple options of paths (orange arrows) heading to the same destination. (c) An imperfect prediction (with heatmap) in Multiverse [24] that goes through a vehicle. (d) An imperfect prediction (with heatmap in the red circle) in Multiverse [24] that violates temporal consistency.

Real-world datasets [2, 19, 30] have enabled the research on trajectory prediction and current approaches [11, 25] have made great progress on single-future trajectory prediction, where the predicted trajectories are evaluated against the ground truth trajectories recorded in the videos. However, the human mind is capricious and realistic situations are complicated. Given an observed trajectory, there can be multiple different destinations and multiple plausible future trajectories. Fig. 1(a) demonstrates that multiple intentions and destinations can drive the pedestrian at the bottom to walk in different paths. Fig. 1(b) shows the pedestrian can select different paths to the same destination.

To evaluate models that generate multiple possible trajectories, Liang *et al*. [24] recently proposed a simulated dataset named "Forking Paths", which provides multiple ground truth trajectories for the same historical trajectory. And in the same work, a two-stage end-to-end probabilistic model named "Multiverse" is designed for multi-future trajectory prediction. However, there are still some issues with this model. For example, Fig. 1(c) shows a path through the vehicle; Fig. 1(d) is an example of temporal inconsistent prediction which violates the normal pattern of human motion with a sharp turn.

In this paper, we propose an encoder-decoder network to address the aforementioned issues. To effectively process the spatial information, we first construct a multi-scale graph to represent scene segmentation and trajectory features. Then, we design a graph-based spatial transformer to learn the interactions between a pedestrian and other pedestrians as well as scene objects. Moreover, to integrate global temporal information, we develop a "Memory Replay" algorithm, which utilizes a memory graph to accumulate temporal information and "replay" it to the transformer at each time step to ensure the smoothness of trajectories. In addition, we propose a new evaluation metric "Percentage of Trajectory Usage" to evaluate the comprehensiveness of multi-future prediction, to complement the existing minADE$_K$ and minFDE$_K$ metrics in [24]. We show that our model achieves state-of-the-art performance on multi-future prediction on the Forking Paths dataset; and our results on single-future prediction are comparable to the current state-of-the-art models on the VIRAT/ActEV [2] dataset. We summarize our main contributions as follows:

1. We propose a graph-based spatial transformer for spatial interactions of pedestrians. By integrating the attention mechanism and graph structure, the spatial transformer can comprehensively generate and aggregate spatial features.
2. We design a novel trajectory smoothing algorithm, Memory Replay, for improving the temporal consistency of predicted trajectories and minimizing the conflicts between spatial and temporal information.
3. We define a new evaluation metric, Percentage of Trajectory Usage (PTU), to evaluate the comprehensiveness of multi-future prediction.

2. Related Work

Pedestrian trajectory prediction. There have been various methods that aim to forecast multiple possible future trajectories. Recent approaches [11, 18, 35] apply Generative Adversarial Networks (GANs) to generate a distribution of trajectories. Inverse Reinforcement Learning (IRL) [7, 17, 27] is also becoming popular on multi-future trajectory prediction tasks. Besides, predicting multiple trajectories is emerging in vehicle trajectory prediction [5, 22, 37, 49]. These approaches, however, have been evaluated using single-future trajectories, as the ground truth contains a single path for each pedestrian. Currently, the *Multiverse* model [24] achieves the state-of-the-art performance on the new 3D simulated dataset, the *Forking Paths*, which is the first public benchmark designed specifically for evaluating the generation of multi-future trajectories. Our model outperforms *Multiverse* on the *Forking Paths* dataset for multi-future trajectory prediction.

GNN-based models. In recent years, Graph Neural Networks (GNNs) have become popular. Traditional GNN models such as Graph Convolutional Network (GCN) [16], GraphSAGE [13] and Graph Attention Network (GAT) [39] are widely used in computer vision tasks such as pose estimation [44, 51], panoptic segmentation [42], point cloud analysis [50], etc. For pedestrian trajectory prediction, Sun *et al*. [36] construct a GCN-based recursive social behavior graph (RSBG) given the annotations by sociologists. STGAT [14] models a spatial-temporal graph attention network to encode the pedestrian interaction. Other works [12, 15, 29, 48] also implement improvements on GNNs to contribute to pedestrian trajectory prediction. We construct a multi-scale graph to model the interactions between pedestrians and multiple scales of surrounding areas.

Transformer-based methods. Transformer-based methods [38] have been a trend in deep learning tasks. It was first used in Natural Language Processing [8, 31, 40], then flourishes in computer vision [4, 9, 10, 53]. Modeling both spatial and temporal transformer [45, 52] can compete or even outperform the traditional sequence-to-sequence models in trajectory prediction, demonstrating their effectiveness in complex spatio-temporal feature processing. Other methods [3, 20, 47] have also inserted transformer-based modules and achieved high performance on both pedestrian and vehicle trajectory prediction. We design a novel graph-based spatial transformer containing an attention-based message generation and a GAT-based aggregation method to effectively collect and process spatial information.

3. Methods

3.1. Overview

Given a series of scene semantic segmentation maps $S = S_1, S_2, \ldots, S_{T_{obs}}$ and positions $X = (x_1, y_1), (x_2, y_2), \ldots, (x_{T_{obs}}, y_{T_{obs}})$ of a pedestrian for time $1 : T_{obs}$, our model aims to predict multiple possible future trajectories where the ith prediction of a pedestrian is denoted as $\hat{Y}^i = (\hat{x}^i_t, \hat{y}^i_t)$ for time $t = T_{obs+1} : T_{pred}$ by learning and inferring $P(\hat{Y}|S, X)$.

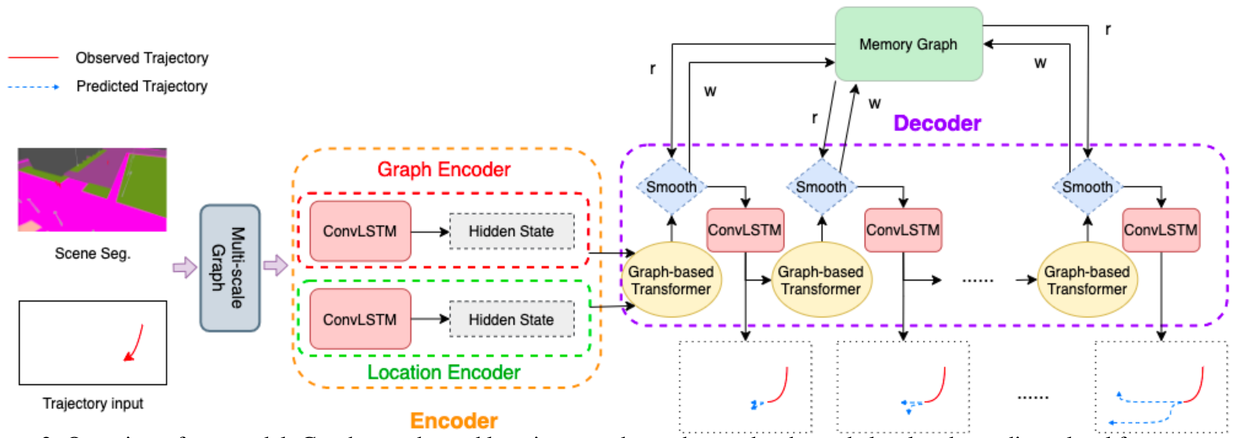


Figure 2. Overview of our model. Graph encoder and location encoder each encodes the node level and coordinate level features processed by the multi-scale graph. At each time step in decoding, our proposed graph-based spatial transformers infer the possible neighboring locations at the next step. Then, our trajectory smoothing algorithm smooths and corrects the locations that violate the temporal consistency based on a memory graph that stores the temporal information of the earlier trajectory.

Fig. 2 shows the overall structure of our proposed model. Our model takes the observed trajectory X and scene segmentation S for video frames in the observation period as inputs. A multi-scale graph is constructed, which is a two-dimensional grid where the areas of the grid cells change with different scales. Each grid cell contains a sub-area of the scene segmentation and trajectory information. The inputs processed by the multi-scale graph are passed into an encoder-decoder network to generate future trajectories. The encoder encodes the motion pattern of pedestrians as well as the scene feature over time. The decoder consists of two main components: a multi-scale graph-based spatial transformer and a trajectory smoothing algorithm which is named “Memory Replay”. The spatial transformer processes the information and makes prediction of the next step. Memory Replay smooths the prediction by reading and writing in a memory graph that contains the overall temporal information of the decoded trajectory. At each time step, the decoder generates a probability distribution of locations at the next time step using these two components, followed by a convolutional LSTM cell [43]. The most probable locations that we select at each time is determined by the diverse beam search [21].

3.2. Multi-scale Graph Generation

We formulate a video frame as a graph $\mathcal{G}(V, E)$ with a set of nodes V and edges E. Specifically, we use a 2D regular grid with $|V|$ grid cells to split a frame into multiple areas, where each area can be considered a node $v \in V$ that connects to adjacent nodes with an undirected edge $e \in E$. Each grid cell can establish connections to the horizontal, vertical and diagonal neighbors. Inspired by the idea of feature pyramid [24,26], we design the graph in different scales to process spatial information in multiple levels. There are two scales of grids which are the same as those in [24], and subsequently, the number of nodes can be 36×18 and 18×9.

Our implementation differs from feature pyramid where we change the amount of features included in a node instead of resizing the whole image (video frame). Nodes in the larger scale have less but finer features, and nodes in the smaller scale have more but coarser features. By learning on the multi-scale graph, our model can be more adaptive to different levels of information and make comprehensive decisions based on surrounding areas of a pedestrian.

3.3. Spatio-temporal Encoder

Inspired by recent studies [24,32], we propose two types of trajectory encoder: graph encoder and location encoder. In each graph scale, the graph encoder encodes the node level feature which is the index of the grid cell where the current location belongs to, and the location encoder record the specific coordinate which is the offset from the center of the area covered by the node. These two encoded hidden states are passed into the decoder separately.

In contrast to recent approaches [1,11,36] that model the motion of all pedestrians in a scene to enrich the spatial features, we utilize convolutional LSTM [43] to encode both spatial and temporal features simultaneously:

$$H(g)_t^{\mathcal{G}} = \text{ConvLSTM}(g_t^{\mathcal{G}}, H(g)_{t-1}^{\mathcal{G}}) \tag{1}$$
$$H(l)_t^{\mathcal{G}} = \text{ConvLSTM}(l_t^{\mathcal{G}}, H(l)_{t-1}^{\mathcal{G}}) \tag{2}$$

which denote the hidden states of graph encoder and location encoder at the graph scale $\mathcal{G}$ at time t, respectively. Since the subsequent process in the encoder and decoder all operates in the same way on these two hidden states, we denote both hidden states collectively as $H_t^{\mathcal{G}}$. To embed the pedestrian location (x_t, y_t) for the graph encoder in graph scale $\mathcal{G}$ at time t, we adapt one-hot encoding multiplied by the scene segmentation:

$$g_t^{\mathcal{G}} = \text{one-hot}(\text{idx}(x_t, y_t)^{\mathcal{G}}) \odot S_t^{\mathcal{G}} \tag{3}$$

where the idx() function converts the coordinate to the index of the grid cell in $\mathcal{G}$. Then, the one-hot function projects the indexed cell onto the corresponding position on the graph. For the location encoder, we further calculate the offset from the center of the indexed cell:

$$l_t^{\mathcal{G}} = x_t'^{\mathcal{G}}, y_t'^{\mathcal{G}} = (x_t, y_t) - C(\text{idx}(x_t, y_t)^{\mathcal{G}}) \tag{4}$$

where $C()$ function obtains the center coordinate of the indexed cell. The offset $(x_t'^{\mathcal{G}}, y_t'^{\mathcal{G}})$ can be calculated by subtracting such center coordinate from the real coordinate (x_t, y_t). We denote this offset coordinate as $l_t^{\mathcal{G}}$.

We keep the hidden state $H_{T_{obs}}^{\mathcal{G}}$ at time T_{obs} after finishing encoding. And following [24], we also calculate an average of semantic segmentation map $\bar{S^{\mathcal{G}}} = \frac{1}{T_{obs}} \sum_{t=1}^{T_{obs}} S_t^{\mathcal{G}}$ and construct the hidden state to be passed into the decoder under graph scale $\mathcal{G}$ as:

$$H_{T_{obs}}^{\mathcal{G}} = (H_{T_{obs}}^{\mathcal{G}} || \bar{S^{\mathcal{G}}}) \tag{5}$$

where $||$ is concatenation. The scene segmentation map provides the pedestrians with awareness of the content and location for each object in the scene, which facilitates modeling of human-to-scene interactions.

3.4. Graph-based Spatial Transformer

Although RNN models are commonly used for addressing sequence prediction tasks [14], they have limitations in collecting nearby information of a person. Recently, [20,52] have made some progress on trajectory prediction by exploiting attention mechanism spatially and temporally. [14] has employed GAT to model human-to-human relationship. However, compared to the crowded scenes in the ETH [30] and UCY [19] datasets, most scenes in the VIRAT/ActEV and Forking Paths datasets [2,24] have much fewer pedestrians, whereas the spatial interaction with scene objects is also important.

Therefore, we design a graph-based spatial transformer to effectively model the relationship between both human-to-scene and human-to-human with the help of scene semantic segmentation features. The transformer takes the graph structured encoded hidden states $H_{T_{obs}}^{\mathcal{G}}$ from Eq. (5) as input node states. We use an attention mechanism to generate messages for all node pairs and aggregate them by the graph structure. The transformer will finally produce a group of updated node states that indicates the possible locations for the next time step.

Attention-based message generation. We generate two types of messages from node v_j to node v_i: attention message and global message.

To extract the interactions between the pedestrian and neighboring areas, we first generate an attention message:

$$M_{Attn[i \leftarrow j]}^{\mathcal{G}} = f_{V[i]}^{\mathcal{G}} \odot (f_{Q[i]}^{\mathcal{G}} || f_{K[j]}^{\mathcal{G}^T}) + \vec{b}^{\mathcal{G}} \tag{6}$$

where $M_{Attn}^{\mathcal{G}}$ is a matrix containing messages between all pairs of nodes in the graph scale $\mathcal{G}$. We learn a query matrix $f_Q^{\mathcal{G}}$, key matrix $f_K^{\mathcal{G}}$ and value matrix $f_V^{\mathcal{G}}$ from the graph structured hidden state $H_{T_{obs}}^{\mathcal{G}}$. Inspired by GAT, we concatenate the query matrix and the transpose of key matrix to create an entry for calculating attention value of each node pair v_i and v_j. Then, we assign an importance value to each entry by conducting an element-wise multiplication between the concatenated matrix and the value matrix and add bias $\vec{b^{\mathcal{G}}}$. We denote the message from node v_j to node v_i as $M_{Attn[i \leftarrow j]}^{\mathcal{G}}$.

The spatial transformer adapts both advantages of self-attention mechanism and graph structure so that a pedestrian can attach different importance on the neighboring areas. However, distant objects (e.g., people, vehicles, obstacles) also provide essential spatial context in trajectory planning. Therefore, we also calculate a similarity score between each node pair $v_i^{\mathcal{G}}$ and $v_j^{\mathcal{G}}$ as a global message:

$$M_{global[i \leftarrow j]}^{\mathcal{G}} = \vec{h}_i^{\mathcal{G}} \vec{h}_j^{\mathcal{G}^T} \tag{7}$$

where $M_{global[i \leftarrow j]}^{\mathcal{G}}$ estimates the distance between features of node v_i and v_j in the hidden space. Finally, we obtain the total message passed from node v_j to node v_i by summing these two types of messages together:

$$M_{[i \leftarrow j]}^{\mathcal{G}} = M_{Attn[i \leftarrow j]}^{\mathcal{G}} \oplus M_{global[i \leftarrow j]}^{\mathcal{G}} \tag{8}$$

where $\oplus$ is element-wise addition. The total message includes both unique information from neighboring nodes and similarity estimation from a global view.

Update of node states. To update node states, we first calculate the edge weights based on the total message in Eq. (8):

$$e_{[i \leftarrow j]}^{\mathcal{G}} = \frac{\exp(M_{[i \leftarrow j]}^{\mathcal{G}})}{\sum_{k \in \mathcal{N}_i^{\mathcal{G}}} \exp(M_{[i \leftarrow k]}^{\mathcal{G}})} \tag{9}$$

where calculated edge weight $e_{[i \leftarrow j]}^{\mathcal{G}}$ is normalized by the softmax function and node k belongs to the neighbors of node i. To update the new node states, we apply a simple dot product between calculated edge weight and the node state in the previous time step:

$$\widetilde{H}_t^{\mathcal{G}}(i) = e_{[i \leftarrow j]}^{\mathcal{G}} h_i^{\mathcal{G}} \tag{10}$$

where $\widetilde{H}_t^{\mathcal{G}}$ is the calculated node state for all nodes. The output at time t and the new hidden state at time $t+1$ are generated by:

$$\hat{P}_t^{\mathcal{G}} = \sigma(\delta 1(H_t^{\mathcal{G}})) \tag{11}$$

$$H_{t+1}^{\mathcal{G}} = \text{ConvLSTM}(\widetilde{H}_t^{\mathcal{G}}, \delta_2(\hat{P}_t^{\mathcal{G}})) \tag{12}$$

where $\hat{P}_t^{\mathcal{G}}$ is the prediction at time t. For each node v_i, $\hat{P}_t^{\mathcal{G}}(i)$ can be considered a probability if the input is from the graph encoder, or a coordinate value offset from the center of node v_i if the input is from the location encoder. The output at time t will be the input at $t+1$ and be passed into the convolutional LSTM cell with the updated node states

$\widetilde{H}_t^{\mathcal{G}}$. δ_1 and $\delta 2$ are two different linear layers. The hidden state at time $t+1$ is represented as $H_{t+1}^{\mathcal{G}}$.

3.5. Memory Replay

Our spatial transformer can encourage the model to focus more on the most probable areas but neglects to take into account the temporal consistency. In the decoder, decoded hidden states from the transformer at time step t is heavily based on the decoded states at $t-1$. However, the locations at the current time is also influenced by the hidden states at all previous time steps. In other words, if we only consider the computations based on the most recent time step, our predictions sometimes deviate from the originally intended destinations implied by the hidden states further before. To address this issue, we propose a trajectory smoothing algorithm "Memory Replay" that utilizes a memory graph $G(V)$ to dynamically record the decoded temporal information of a trajectory, where $|V|$ is the same as the number of nodes in the hidden state graph scale $\mathcal{G}$. Memory Replay operates on the edge weights calculated by the transformer. At each time step, the memory graph G carries the smoothed edge weights of all node pairs (including a node to itself) in the past decoding time steps and decreases the weights of edges that point to the temporally inconsistent locations.

The processing steps are shown in Algorithm 1. Before decoding, we initialize the memory graph to all zeros. During each time step in decoding, we first pass the hidden state at the previous time into our spatial transformer to calculate the edge weights $e^{\mathcal{G}}$ for each node by Eq. (9) (Line 5). Then, we smooth the value of $e^{\mathcal{G}}$ by element-wise addition with G (Line 6). G is populated with the smoothed $e^{\mathcal{G}}$ at every time step (Line 7) to ensure that it includes the most recent decoded states. The hidden state at the current time is calculated by combining the updated node states based on the smoothed edge weights and the hidden state at the previous time in Eq. (10), as well as the output at the previous time (observed trajectory when the time is T_{obs}) in Eq. (12) (Line 9). Finally, we generate the output at the current time based on the new hidden state by Eq. (11) (Line 10). Therefore, the memory graph can record the newest edge weights at each time step, where the edge weights are smoothed by the memory graph at the previous time step. Memory Replay effects in such a recursive manner.

3.6. Loss

Following [24], we split our training as a classification task (graph encoder stream) and regression task (location encoder stream). We consider the ground truth output for each graph scale $\mathcal{G}$ at each time t as $P_i^{\mathcal{G}}(t)$ and the duration for loss calculation as $T_{1:loss}$. We use cross-entropy loss for

Algorithm 1: Memory Replay.

```
input : Encoded last hidden state $H^{\mathcal{G}}_{T_{obs}}$ and the last
        observed trajectory $\hat{P}^{\mathcal{G}}_{T_{obs}}$ in Graph scale $\mathcal{G}$
        at time $T_{obs}$
1  for $\mathcal{G} \in \{(18,32),(9,16)\}$ do
2      $G \leftarrow zeros \times \mathcal{G}$
3      $H^{\mathcal{G}}_{T_{prev}}, \hat{P}^{\mathcal{G}}_{T_{prev}} \leftarrow H^{\mathcal{G}}_{T_{obs}}, \hat{P}^{\mathcal{G}}_{T_{obs}}$
4      for $T_{curr} \longleftarrow T_{obs+1}$ to $T_{pred}$ do
5          $e^{\mathcal{G}} \leftarrow$ calculate edge weights by the spatial
             transformer in Eq. (9)
6          $e^{\mathcal{G}} \leftarrow \sigma(e^{\mathcal{G}} \oplus G)$
7          $G \leftarrow e^{\mathcal{G}}$
8          $T_{prev} \leftarrow T_{curr}$
9          $H^{\mathcal{G}}_{T_{prev}} \leftarrow$ prepare for hidden state in the
             next time step by Eq. (10) and Eq. (12)
10         $\hat{P}^{\mathcal{G}}_{T_{prev}} \leftarrow$ generate output with $H^{\mathcal{G}}_{T_{prev}}$ by
             Eq. (11)
11     end
12 end
```

the graph encoder stream:

$$L_c^{\mathcal{G}} = -\frac{1}{T_{loss}} \sum_{t=T_1}^{T_{loss}} \sum_{i \in \mathcal{G}} P_i^{\mathcal{G}}(t) \log(\hat{P}_i^{\mathcal{G}}(t)) \quad (13)$$

In addition, inspired by [36], we propose exponential smooth $L1$ loss for the location encoder stream:

$$L_r^{\mathcal{G}} = \frac{1}{T_{loss}} \sum_{t=T_1}^{T_{loss}} \sum_{i \in \mathcal{G}} \text{Smooth}_{L1}(P_i^{\mathcal{G}}(t), \hat{P}_i^{\mathcal{G}}(t)) \times e^{\frac{T_{loss}-t+1}{\mu}} \quad (14)$$

where we define a penalty term $e^{\frac{T_{loss}-t+1}{\mu}}$ to guide the model focus more on the predictions at earlier time steps as the quality of earlier trajectories can greatly affect the later ones. The hyperparameter μ is used to control the strength of the penalty term.

To benefit from the multi-scale graph, we refer to multi-scale discriminators [41] and sum the loss calculated in both scales $Scales \in [36 \times 18, 18 \times 9]$:

$$L = \sum_{\mathcal{G} \in Scales} \alpha L_c^{\mathcal{G}} + \beta L_r^{\mathcal{G}} \quad (15)$$

and L is used to optimize the training in both scales.

3.7. Generation of Multiple Trajectories

We refer to [21, 24] utilizing diverse beam search to generate multiple trajectories in the graph encoder stream. At time $t-1$ in the graph scale $\mathcal{G}$, we obtain a set of K decoded trajectories with their conditional logarithmic probabilities denoted as $C_1^{\mathcal{G},k}, C_2^{\mathcal{G},k} \ldots C_{t-1}^{\mathcal{G},k}$ where $k \in [1, K]$ and K is the beam size. Given the probability $\hat{P}_t^{\mathcal{G},k}$ inferred by the model at time t, we calculate the new logarithmic probability of the graph node i in the beam k as:

$$C_t^{\mathcal{G},k}(i) = C_{t-1}^{\mathcal{G},k} + \log(\hat{P}_t^{\mathcal{G},k}(i)) - \gamma(i) \tag{16}$$

where $i \in \mathcal{G}$ and $k \in [1, K]$. $\gamma(i)$ is the diversity rate. In total, we need to calculate $|V| \times K$ such probabilities for all nodes and beams, where $|V|$ is the number of nodes. Finally, we select top K of them as the predictions. For the location encoder stream, we apply the offset values to the predicted nodes to obtain the precise coordinates.

4. Experiments

4.1. Evaluation Metrics

Single Future Evaluation. Same as previous studies [1, 11, 25], we use the following two common evaluation metrics: 1) Average Displacement Error (ADE): the average $L2$ distance between the ground truth locations and predicted locations over all time steps. 2) Final Displacement Error (FDE): the $L2$ distance between the ground truth locations and predicted locations at the final time step.

Multi-future Evaluation. We assume that for each data sample, there are J ground truth trajectories and the model makes K predictions. Following the recent public benchmark [24] on the Forking Paths dataset, we use: 1) Minimum Average Displacement Error Given K Predictions (minADE$_K$); and 2) Minimum Final Displacement Error Given K Predictions (minFDE$_K$). For each ground truth $j \in J$ in the data sample $i \in N$, we choose one of the K predictions with the smallest overall distance to j to calculate the average displacement and the one with the smallest final distance to calculate the final displacement.

Percentage of Trajectory Usage. We propose a new method of evaluation named "Percentage of Trajectory Usage" (PTU) to evaluate the comprehensiveness of the performance of multi-future prediction.

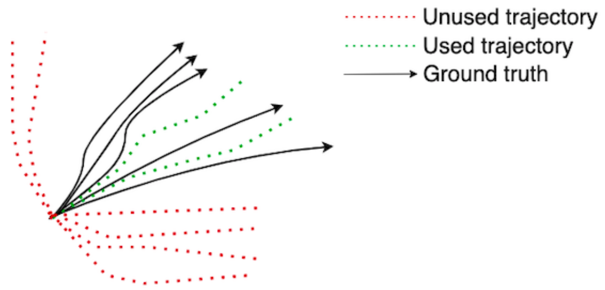


Figure 3. An illustration of low percentage of predicted trajectories usage by minADE$_K$ evaluation. There are 5 ground truth trajectories (black arrows) and 8 predictions (dotted lines). But only 2 predictions (green dotted lines) are used while evaluating with minADE$_K$ and others (red dotted lines) are unused.

Although minADE$_K$ and minFDE$_K$ can evaluate the displacement between prediction and ground truth, they ignore the diversity of prediction distribution. As illustrated in Fig. 3, there are 5 ground truth trajectories and 8 predicted trajectories. However, according to the definitions of minADE$_K$ and minFDE$_K$, only 2 predictions are included in the evaluation, which are the closest prediction for each ground truth. We consider this an incomprehensive prediction, where several ground truth trajectories share the same predicted trajectory. Ideally, we expect to have 5 distinct predictions corresponding to 5 ground truth trajectories. In addition, Yuan *et al*. [46] developed *Average Self Distance* (ASD) and *Final Self Distance* (FSD) to evaluate the diversity of prediction distribution by calculating the $L2$ distance between each prediction with its nearest prediction. However, diversity of prediction that calculated by ASD and FSD fails to consider the number of predicted trajectories that lie in the distribution of ground truth.

To evaluate the comprehensiveness of prediction distribution, we define PTU as:

$$\text{PTU} = \frac{\sum_{i=1}^{N} |\hat{p}_i| / |Y_i|}{N} \tag{17}$$

where $|\hat{p}_i|$ denotes the number of predictions used while evaluating with minADE$_K$ and minFDE$_K$ and $|Y_i|$ denotes the number of ground truth trajectories in a data sample. We sum such percentage for all N data samples then average it. Under the same result of minADE$_K$ and minFDE$_K$, a larger PTU represents a more comprehensive prediction.

4.2. Implementation Details

We use the same data processing method as [11] and, following [24], we apply the pre-trained scene segmentation model [6] to obtain scene segmentation features. We utilize a single convolutional LSTM layer for both the encoder and decoder; and aggregate features of one-hop neighbors in our graph-based transformer. We set the learning rate as 0.3 with decay value of 0.95 and weight decay of 0.001, which are the same as [24]. We embed the graph features with an embedding size of 32 and the hidden state sizes for both the encoder and decoder are 256. For hyperparameters α and β in total loss, we set $\alpha = 1.0$ and $\beta = 0.2$; and μ in our exponential smooth $L1$ loss is 10. We only apply such exponential loss on single-future prediction as it will affect the diversity of multi-future prediction through experiments. In order to align with [24], we also generate $K = 20$ most possible predictions for each data sample in multi-future prediction.

4.3. Multi-future Prediction

The Forking Paths Dataset. The Forking Paths dataset [24] is a simulated dataset specifically designed for multi-future prediction. This dataset was constructed by 5 scenes in VIRAT/ActEV and 4 scenes in ETH/UCY. There are 127 scenarios, each of which is rendered in three 45-degree views and one top-down view. There is one controlled agent in each scenario which has on average 5.9 future trajecto-

Method	minADE$_{20}$ ↓				minFDE$_{20}$ ↓			
	45-degree	top-down	all	PTU ↑	45-degree	top-down	all	PTU ↑
LSTM	201.0	183.7	196.7	N/A	381.5	355.0	374.9	N/A
Social-GAN(PV)	191.2	176.5	187.5	44.70%	351.9	335.0	347.7	42.82%
Social-GAN(V)	187.1	172.7	183.5	43.00%	342.1	326.7	338.3	41.85%
Next	186.6	166.9	181.7	N/A	360.0	326.6	351.7	N/A
Multiverse	168.9	157.7	166.1	47.45%	333.8	316.5	329.5	44.35%
Ours	**165.5**	**154.5**	**162.8**	**48.65%**	**318.9**	**302.5**	**314.8**	**50.83%**

Table 1. Quantitative evaluation of multi-future trajectory prediction. minADE$_K$ and minFDE$_K$ results are presented on 45-degree, top-down and all views. PTU results are only evaluated for multi-future prediction models. All models are trained on the VIRAT/ActEV dataset and tested on the Forking Paths dataset.

ries. We aim to predict multiple trajectories for each controlled agent. The length of observation time is $T_{obs} = 8$ frames and prediction time is $T_{pred-obs} = 12$ frames.

Baselines. We compare our model with 4 baseline models. ***LSTM***: A simple LSTM implementation which only models the trajectory inputs. ***Social GAN*** [11]: A recent GAN-based model that generates multimodal prediction distributions. We report two configurations: the model with only variety loss (Social-GAN(V)) and with both variety loss and global pooling (Social-GAN(PV)). ***Next*** [25]: the state-of-the-art model on the VIRAT/ActEV for single-future prediction. Since the model utilizes rich visual features, we compare our model with Next without activity prediction module. ***Multiverse*** [24]: the recent state-of-the-art probabilistic model for multi-future prediction on the Forking Paths dataset.

Quantitative Evaluation. Table 1 shows the comparisons of multi-future prediction between basline models and our model in minADE$_{20}$, minFDE$_{20}$ and PTU metrics. We can see that our model outperforms all baseline methods. Compared to the current state-of-the-art model Multiverse, the average minADE$_{20}$ reduces by 3 points and the average minFDE$_{20}$ reduces by 15 points on all views. The PTU value is higher than Multiverse under minADE$_K$ by 1.2% and under minFDE$_K$ by 6.5%, which demonstrates that our model generates more comprehensive predictions.

Qualitative Evaluation. Fig. 4(a) are the results of Multiverse and Fig. 4(b) are those of ours. From the three sets of comparisons on the left, we can see that the predictions of Multiverse go through the vehicles, whereas ours can make predictions that lie in the ground truth distribution without colliding with other objects. These cases demonstrate that our spatial transformer can detect objects and make reasonable decisions accordingly. Furthermore, the three sets on the right in Fig. 4(a) show the temporal inconsistent cases of Multiverse, whereas ours can make smooth predictions, which reflects that our Memory Replay is effective in retaining the temporal consistency of predictions.

Method	ADE↓	FDE↓
LSTM	23.98	44.97
Social-GAN(V)	30.40	61.93
Social-GAN(PV)	30.42	60.70
Next	19.78	42.43
Multiverse	**18.51**	**35.84**
Ours	18.58	36.08

Table 2. Quantitative evaluation of single-future trajectory prediction on VIRAT/ActEV dataset in ADE and FDE metrics.

Method	ADE↓	FDE↓
Single-scale graph	19.71	37.32
No Location Encoder	41.18	61.23
No Memory Replay	19.34	37.05
No exponential loss	19.39	37.09
Full Model	**18.58**	**36.08**

Table 3. Ablation study for key components in our model on single-future prediction.

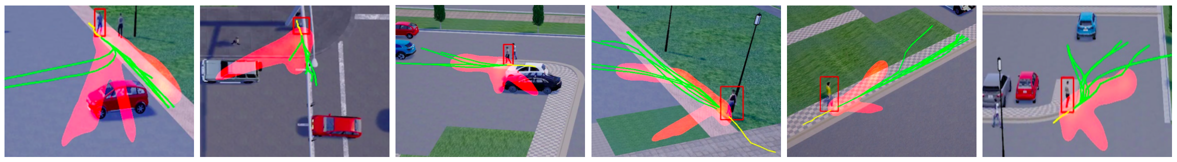
4.4. Single Future Prediction

VIRAT/ActEV Dataset. Following [24], we use VIRAT/ActEV [2] as the dataset for single-future trajectory prediction. This dataset is designed to evaluate tasks such as activity detection and object tracking. We use the same training, validation and testing split as [23–25] to make fair comparisons. Observation length is 3.2 seconds (8 frames) and prediction length is 4.8 seconds (12 frames), which are the same as the previous works [1, 11, 23–25].

Quantitative Evaluation. The results of single-future prediction are shown in Table 2. The result of our model is the second best and very close to that of Multiverse, exhibiting large improvement over GAN-based models. This implies that our model performs effectively for both simulated multi-future and real-world single-future scenarios.

4.5. Ablation Study

Ablations of key components. For single-future prediction, we verify four components: without multi-scale graph and only keep a scale of 36 × 18, no Memory Replay module, no Location Encoder, and no Exponential Smooth $L1$


(a) Imperfect/Error cases of Multiverse

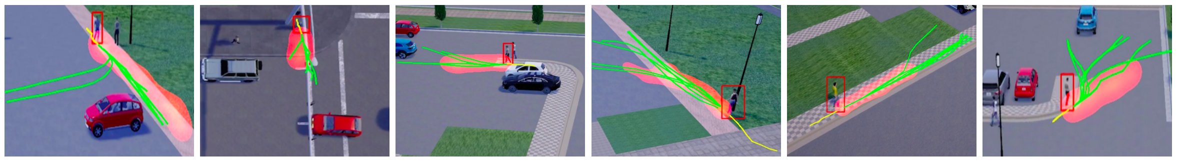

(b) Improved cases of our model

Figure 4. Qualitative comparisons between the Multiverse and our model. The yellow and green lines are observed and ground truth trajectories. The heatmaps are prediction distributions.

Method	minADE$_{20}$ ↓				minFDE$_{20}$ ↓			
	45-degree	top-down	all	PTU ↑	45-degree	top-down	all	PTU ↑
Single-scale graph	170.9	160.3	168.3	44.89%	337.9	315.8	332.4	39.96%
No Location Encoder	245.4	237.3	243.4	31.14%	463.0	441.7	457.7	28.56%
No Memory Replay	167.3	**153.2**	163.7	46.53%	330.1	306.2	324.1	42.43%
Ours (Full Model)	**165.5**	154.5	**162.8**	**48.65%**	**318.9**	**302.5**	**314.8**	**50.83%**

Table 4. Ablation study for key components in our model on multi-future prediction.

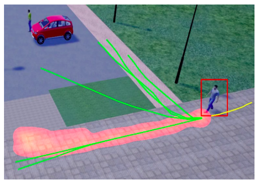
Value of μ	ADE↓	FDE↓
$+\infty$	19.39	37.09
μ=20	19.19	36.83
μ=10	18.58	**36.08**
μ=5	**18.56**	36.24

Table 5. Ablation study for different values of μ in our Exponential Smooth $L1$ loss on single-future prediction.

loss. For multi-future prediction, we only test on the first three components as the exponential loss is only designed for promoting performance of single-future prediction in training. As indicated in Tables 3 and 4, without any of these key components, our model shows performance drop in varying degrees. Additionally, we apply PTU on all key components on multi-future prediction. From Table 4, we can see that our model achieves the highest PTU under both minADE$_K$ and minFDE$_K$.

Exponential smooth $L1$ loss. Multiplying the penalty term can lead to small improvement, as it guides the model to focus more on the earlier data in a sequence, which can affect the overall performance of predictions in the whole sequence. We select the values of μ as $+\infty$, 20, 10 and 5 ($+\infty$ means only using smooth $L1$ loss) for comparisons. Table 5 shows that when $\mu = 10$, our model achieves the overall best performance, which reduces the ADE by 3.6% and FDE by 2.4% on average.

Limitations. We show some imperfect cases of our model as limitations. Fig. 5(a) is a case that our model only predicts trajectories that go straight ahead, whereas there are some ground truth trajectories turning right in diverse degrees. It might be improved if diversity control is applied on loss functions or during selecting final predictions. Fig. 5(b) shows our model sometimes is not aware of the walking speed and makes prediction apparently longer than the ground truth. Data augmentation would help reduce similar occurrences. These imperfect cases and possible ideas would inspire our future research.


(a)

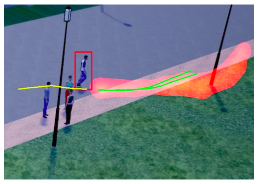

(b)

Figure 5. Example of limitations of our model.

5. Conclusion

This paper focuses on multi-future pedestrian trajectory prediction when there are multiple plausible future trajectories for each pedestrian in the ground truth. We model the spatial interactions by a graph-based spatial transformer, which utilizes an improved attention-based message generation and aggregation method as well as adopting a multi-scale graph structure. We also introduce the Memory Replay algorithm to generate smooth trajectories by coordinating with the transformer. Moreover, we propose Percentage of Trajectory Usage to evaluate the comprehensiveness of multi-future prediction. Our proposed model achieves state-of-the-art performance for multi-future prediction on the Forking Paths dataset and our single-future prediction results can compete with those by current state-of-the-art models on the VIRAT/ActEV dataset.

References

[1] 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*, pages 961–971, 2016. 3, 6, 7

[2] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzad Godil, David Joy, Andrew Delgado, Alan Smeaton, Yvette Graham, et al. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. In *Proceedings of TRECVID 2018*, 2018. 1, 2, 4, 7

[3] Alessia Bertugli, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. AC-VRNN: Attentive conditional-VRNN for multi-future trajectory prediction. *Computer Vision and Image Understanding*, 210:103245, 2021. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 1, 2

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 6

[7] Dooseop Choi, Kyoungwook Min, and Jeongdan Choi. Regularising neural networks for future trajectory prediction via inverse reinforcement learning framework. *IET Computer Vision*, 14(5):192–200, 2020. 2

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[10] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2

[11] 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*, pages 2255–2264, 2018. 1, 2, 3, 6, 7

[12] Sirin Haddad and Siew Kei Lam. Graph2kernel Grid-LSTM: A multi-cued model for pedestrian trajectory prediction by learning adaptive neighborhoods. *arXiv preprint arXiv:2007.01915*, 2020. 2

[13] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. 2

[14] Yingfan Huang, HuiKun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. STGAT: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6272–6281, 2019. 2, 4

[15] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019. 2

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[17] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European conference on computer vision*, pages 201–214. Springer, 2012. 2

[18] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks. *arXiv preprint arXiv:1907.03395*, 2019. 2

[19] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007. 1, 4

[20] Jiachen Li, Hengbo Ma, Zhihao Zhang, and Masayoshi Tomizuka. Social-WaGDAT: Interaction-aware trajectory prediction via wasserstein graph double-attention network. *arXiv preprint arXiv:2002.06241*, 2020. 2, 4

[21] Jiwei Li, Will Monroe, and Dan Jurafsky. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*, 2016. 3, 5

[22] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving. *arXiv preprint arXiv:1907.07792*, 2019. 2

[23] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Simaug: Learning robust representations from simulation for trajectory prediction. In *European Conference on Computer Vision*, pages 275–292. Springer, 2020. 7

[24] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The Garden of Forking Paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020. 1, 2, 3, 4, 5, 6, 7

[25] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the Future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019. 1, 6, 7

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[27] Yuejiang Liu, Qi Yan, and Alexandre Alahi. Social NCE: Contrastive learning of socially-aware motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15118–15129, 2021. 2

[28] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 1

[29] 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*, pages 14424–14432, 2020. 2

[30] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European conference on computer vision*, pages 452–465. Springer, 2010. 1, 4

[31] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 2

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 3

[33] Christoph Rösmann, Malte Oeljeklaus, Frank Hoffmann, and Torsten Bertram. Online trajectory prediction and planning for social robot navigation. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1255–1260. IEEE, 2017. 1

[34] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the Untrackable: Learning to track multiple cues with long-term dependencies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 300–311, 2017. 1

[35] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 2

[36] Jianhua Sun, Qinhong Jiang, and Cewu Lu. Recursive social behavior graph for trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 660–669, 2020. 2, 3, 5

[37] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32:15424–15434, 2019. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 2

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 2

[41] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 5

[42] Yangxin Wu, Gengwei Zhang, Yiming Gao, Xiajun Deng, Ke Gong, Xiaodan Liang, and Liang Lin. Bidirectional graph reasoning network for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9080–9089, 2020. 2

[43] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 3

[44] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 2

[45] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523. Springer, 2020. 2

[46] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. *arXiv preprint arXiv:1907.04967*, 2019. 6

[47] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting. *arXiv preprint arXiv:2103.14023*, 2021. 2

[48] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. SR-LSTM: State refinement for LSTM towards pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019. 2

[49] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. TNT: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 1, 2

[50] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019. 2

[51] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019. 2

[52] Xiaodong Zhao, Yaran Chen, Jin Guo, and Dongbin Zhao. A spatial-temporal attention model for human trajectory prediction. *IEEE CAA J. Autom. Sinica*, 7(4):965–974, 2020. 2, 4

[53] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 2