2021 IEEE/CVF International Conference on Computer Vision (ICCV) | 978-1-6654-2812-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCV48922.2021.00967

AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting

Ye Yuan[1] Xinshuo Weng[1] Yanglan Ou[2] Kris Kitani[1]
[1]Carnegie Mellon University [2]Penn State University
https://www.ye-yuan.com/agentformer

Abstract

*Predicting accurate future trajectories of multiple agents is essential for autonomous systems but is challenging due to the complex interaction between agents and the uncertainty in each agent's future behavior. Forecasting multi-agent trajectories requires modeling two key dimensions: (1) **time dimension**, where we model the influence of past agent states over future states; (2) **social dimension**, where we model how the state of each agent affects others. Most prior methods model these two dimensions separately, e.g., first using a temporal model to summarize features over time for each agent independently and then modeling the interaction of the summarized features with a social model. This approach is suboptimal since independent feature encoding over either the time or social dimension can result in a loss of information. Instead, we would prefer a method that allows an agent's state at one time to **directly** affect another agent's state at a future time. To this end, we propose a new Transformer, termed AgentFormer, that simultaneously models the time and social dimensions. The model leverages a sequence representation of multi-agent trajectories by flattening trajectory features across time and agents. Since standard attention operations disregard the agent identity of each element in the sequence, AgentFormer uses a novel agent-aware attention mechanism that preserves agent identities by attending to elements of the same agent differently than elements of other agents. Based on AgentFormer, we propose a stochastic multi-agent trajectory prediction model that can attend to features of any agent at any previous timestep when inferring an agent's future position. The latent intent of all agents is also jointly modeled, allowing the stochasticity in one agent's behavior to affect other agents. Extensive experiments show that our method significantly improves the state of the art on well-established pedestrian and autonomous driving datasets.*

1. Introduction

The safe planning of autonomous systems such as self-driving vehicles requires forecasting accurate future trajectories of surrounding agents (*e.g.*, pedestrians, vehicles). However, multi-agent trajectory forecasting is challenging since the social interaction between agents, *i.e.*, behavioral influence of an agent on others, is a complex process. The problem is further complicated by the uncertainty of each agent's future behavior, *i.e.*, each agent has its latent intent unobserved by the system (*e.g.*, turning left or right) that governs its future trajectory and in turn affects other agents. Therefore, a good multi-agent trajectory forecasting method should effectively model (1) the complex social interaction between agents and (2) the latent intent of each agent's future behavior and its social influence on other agents.

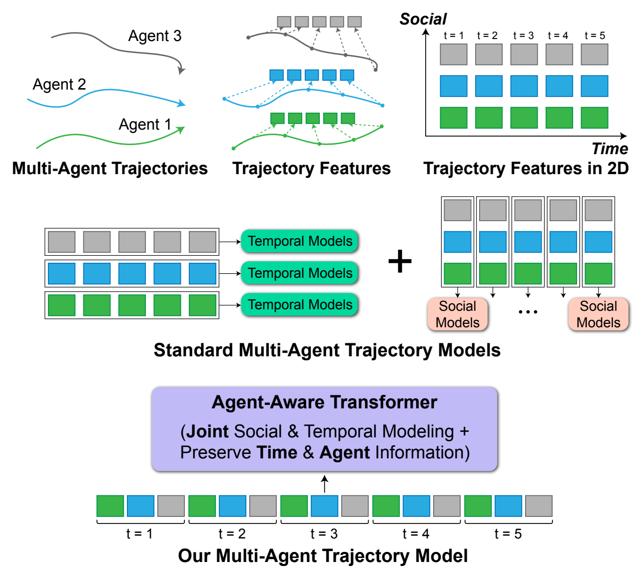

Figure 1. Different from standard approaches that model multi-agent trajectories in the time and social dimensions separately, our AgentFormer allows for joint modeling of the time and social dimensions while preserving time and agent information.

Multi-agent social interaction modeling involves two key dimensions as illustrated in Fig. 1 (Top): (1) ***time dimension***, where we model how past agent states (positions and velocities) influence future agent states; (2) ***social dimension***, where we model how each agent's state affects the

978-1-6654-2812-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCV48922.2021.00967

state of other agents. Most prior multi-agent trajectory forecasting methods model these two dimensions separately (see Fig. 1 (Middle)). Approaches like [25, 1, 15] first use temporal models (*e.g*., LSTMs [17] or Transformers [46]) to summarize trajectory features over time for each agent independently and then input the summarized temporal features to social models (*e.g*., graph neural networks [23]) to capture social interaction between agents. Alternatively, methods like [44, 18] first use social models to produce social features for each agent at each independent timestep and then apply temporal models over the social features. In this work, we argue that modeling the time and social dimensions separately can be suboptimal since the independent feature encoding over either the time or social dimension is not informed by features across the other dimension, and the encoded features may not contain the necessary information for modeling the other dimension.

To tackle this problem, we propose a new Transformer model, termed AgentFormer, that simultaneously learns representations from both the time and social dimensions. AgentFormer allows an agent's state at one time to affect another agent's state at a future time *directly* instead of through intermediate features encoded over one dimension. As Transformers require sequences as input, we leverage a sequence representation of multi-agent trajectories by flattening trajectory features across time and agents (see Fig. 1 (Bottom)). However, directly applying standard Transformers to these multi-agent sequences will result in a loss of *time* and *agent* information since standard attention operations discard the timestep and agent identity associated with each element in the sequence. We solve the loss of time information using a time encoder that appends a timestamp feature to each element. However, the loss of agent identity is a more complicated problem: unlike time, there is no innate ordering between agents, and assigning an agent index-based encoding will break the required permutation invariance of agents and create artificial dependencies on agent indices in the model. Instead, we propose a novel agent-aware attention mechanism to preserve agent information. Specifically, agent-aware attention generates two sets of keys and queries via different linear transformations; one set of keys and queries is used to compute inter-agent attention (agent to agent) while the other set is designated for intra-agent attention (agent to itself). This design allows agent-aware attention to attend to elements of the same agent differently than elements of other agents, thus keeping the notion of agent identity. Agent-aware attention can be implemented efficiently via masked operations. Furthermore, AgentFormer can also encode rule-based connectivity between agents (*e.g*., based on distance) by masking out the attention weights between unconnected agents.

Based on AgentFormer, which allows us to model social interaction effectively, we propose a multi-agent trajectory prediction framework that also models the social influence of each agent's future trajectory on other agents. The probabilistic formulation of the model follows the conditional variational autoencoder (CVAE [21]) where we model the generative future trajectory distribution conditioned on context (*e.g*., past trajectories, semantic maps). We introduce a latent code for each agent to represent its latent intent. To model the social influence of each agent's future behavior (governed by latent intent) on other agents, the latent codes of all agents are jointly inferred from the future trajectories of all agents during training, and they are also jointly used by a trajectory decoder to output socially-aware multi-agent future trajectories. Thanks to AgentFormer, the trajectory decoder can attend to features of any agent at any previous timestep when inferring an agent's future position. To improve the diversity of sampled trajectories and avoid similar samples caused by random sampling, we further adopt a multi-agent trajectory sampler that can generate diverse and plausible multi-agent trajectories by mapping context to various configurations of all agents' latent codes.

We evaluate our method on well-established pedestrian datasets, ETH [37] and UCY [28], and an autonomous driving dataset, nuScenes [3]. On ETH/UCY and nuScenes, we outperform state-of-the-art multi-agent prediction methods with substantial performance improvement (41% and 42%). We further conduct extensive ablation studies to show the superiority of AgentFormer over various combinations of social and temporal models. We also demonstrate the efficacy of agent-aware attention against agent encoding.

To summarize, the main contributions of this paper are: (1) We propose a new Transformer that simultaneously models the time and social dimensions of multi-agent trajectories with a sequence representation. (2) We propose a novel agent-aware attention mechanism that preserves the agent identity of each element in the multi-agent trajectory sequence. (3) We present a multi-agent forecasting framework that models the latent intent of all agents jointly to produce socially-plausible future trajectories. (4) Our approach significantly improves the state of the art on well-established pedestrian and autonomous driving datasets.

2. Related Work

Sequence Modeling. Sequences are an important representation of data such as video, audio, price, *etc*. Historically, RNNs (*e.g*., LSTMs [17], GRUs [7]) have achieved remarkable success in sequence modeling, with applications to speech recognition [51, 35], image captioning [52], machine translation [32], human pose estimation [55, 24], *etc*. In particular, RNNs have been the preferred temporal models for trajectory and motion forecasting. Many RNN-based methods model the trajectory pattern of pedestrians to predict their 2D future locations [1, 19, 60]. Prior work has also used RNNs to model the temporal dynamics of 3D human pose [11, 57, 59]. With the invention of Transformers and positional encoding [46], many works start to adopt Transformers for sequence modeling due to their strong ability to capture long-range dependencies. Transformers have first dominated the natural language processing (NLP) domain across various tasks [9, 26, 53]. Beyond NLP, numerous visual Transformers have been proposed to tackle vision tasks, such as image classification [10], object detection [4], and instance segmentation [49]. Recently, Transformers have also been used for trajectory forecasting. Transformer-TF [12] applies the standard Transformer to predict the future trajectories of each agent independently. STAR [54] uses separate temporal and spatial Transformers to forecast multi-agent trajectories. Interaction Transformer [30] combines RNNs and Transformers for multi-agent trajectory modeling. Different from prior work, Our AgentFormer leverages a sequence representation of multi-agent trajectories and a novel agent-aware attention mechanism to preserve time and agent information in the sequence.

Trajectory Prediction. Early work on trajectory prediction adopts a deterministic approach using models such as social forces [16], Gaussian process (GP) [48], and RNNs [1, 36, 47]. A thorough review of these deterministic methods is provided in [42]. As the future trajectory of an agent is uncertain and often multi-modal, recent trajectory prediction methods start to model the trajectory distribution with deep generative models [21, 13, 39] such as conditional variational autoencoders (CVAEs) [27, 56, 19, 45, 50, 44], generative adversarial networks (GANs) [15, 43, 25, 61], and normalizing flows (NFs) [40, 41, 14]. Most of these methods follow a seq2seq structure [2, 6] and predict future trajectories using intermediate features of past trajectories. In contrast, our AgentFormer-based trajectory prediction framework can directly attend to features of any agent at any previous timestep when inferring an agent's future position. Moreover, our approach models the future trajectories of all agents jointly to predict socially-aware trajectories.

Social Interaction Modeling. Methods for social interaction modeling can be categorized based on how they model the time and social dimensions. While RNNs [17, 7] and Transformers [46] are the prefered temporal models [18, 1, 54], graph neural networks (GNNs) [23, 31] are often employed as the social models for interaction modeling [22, 29, 25]. One popular type of methods [25, 1, 15] first uses temporal models to summarize trajectory features over time for each agent independently and then feeds the temporal features to social models to obtain socially-aware agent features. Alternatively, approaches like [44, 18] first use social models to produce social features of each agent at each independent timestep and then apply temporal models to summarize the social features over time for each agent. One common characteristic of these prior works is that they model the time and social dimensions on separate levels. This can be suboptimal since it prevents an agent's feature at one time from directly interacting with another agent's feature at a different time, thus limiting the model's ability to capture long-range dependencies. Instead, our method models both the time and social dimensions simultaneously, allowing direct feature interaction across time and agents.

3. Approach

We formulate multi-agent trajectory prediction as modeling the generative future trajectory distribution of N (variable) agents conditioned on their past trajectories. For observed timesteps $t \leq 0$, we represent the joint state of all N agents at time t as $\mathbf{X}^t = (\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_N^t)$, where $\mathbf{x}_n^t \in \mathbb{R}^{d_s}$ is the state of agent n at time t, which includes the position, velocity and (optional) heading angle of the agent. We denote the history of all agents as $\mathbf{X} = \left(\mathbf{X}^{-H}, \mathbf{X}^{-H+1}, \ldots, \mathbf{X}^0\right)$ which includes the joint agent state at all $H+1$ observed timesteps. Similarly, the joint state of all N agents at future time t ($t > 0$) is denoted as $\mathbf{Y}^t = (\mathbf{y}_1^t, \mathbf{y}_2^t, \ldots, \mathbf{y}_N^t)$, where $\mathbf{y}_n^t \in \mathbb{R}^{d_p}$ is the future position of agent n at time t. We denote the future trajectories of all N agents over T future timesteps as $\mathbf{Y} = \left(\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^T\right)$. Depending on the data, optional contextual information $\mathbf{I}$ may also be given, such as a semantic map around the agents (annotations of sidewalks, road boundaries, *etc*.). Our goal is to learn a generative model $p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{I})$ where θ are the model parameters.

In the following, we first introduce the proposed agent-aware Transformer, AgentFormer, for joint modeling of socio-temporal relations. We then present a stochastic multi-agent trajectory prediction framework that jointly models the latent intent of all agents.

3.1. AgentFormer: Agent-Aware Transformers

Our agent-aware Transformer, AgentFormer, is a model that learns representations from multi-agent trajectories over both time and social dimensions simultaneously, in contrast to standard approaches that model the two dimensions in separate stages. AgentFormer has two types of modules – encoders and decoders, which follow the encoder and decoder design of the original Transformer [46] but with two major differences: (1) it replaces positional encoding with a time encoder; (2) it uses a novel agent-aware attention mechanism instead of the scaled dot-product attention. As we will discuss below, these two modifications are motivated by a sequence representation of multi-agent trajectories that is suitable for Transformers.

Multi-Agent Trajectories as a Sequence. The past multi-agent trajectories $\mathbf{X}$ can be denoted as a sequence $\mathbf{X} = \left(\mathbf{x}_1^{-H}, \ldots, \mathbf{x}_N^{-H}, \mathbf{x}_1^{-H+1}, \ldots, \mathbf{x}_N^{-H+1}, \ldots, \mathbf{x}_1^0, \ldots, \mathbf{x}_N^0\right)$ of length $L_p = N \times (H+1)$. Similarly, the future multi-agent trajectories can also be represented as a sequence

$\mathbf{Y} = \left(\mathbf{y}_1^1, \ldots, \mathbf{y}_N^1, \mathbf{y}_1^2, \ldots, \mathbf{y}_N^2, \ldots, \mathbf{y}_1^T, \ldots, \mathbf{y}_N^T\right)$ of length $L_f = N \times T$. We adopt this sequence representation to be compatible with Transformers. At first glance, it may seem that we can directly apply standard Transformers to these sequences to model temporal and social relations. However, there are *two problems* with this approach: (1) **loss of *time* information**, as Transformers have no notion of time when computing attention for each element (*e.g.*, $\mathbf{x}_n^t$) *w.r.t.* other elements in the sequence; for instance, $\mathbf{x}_n^t$ does not know $\mathbf{x}_m^t$ is a feature of the same timestep while $\mathbf{x}_n^{t+1}$ is a feature of the next timestep; (2) **loss of *agent* information**, since Transformers do not consider agent identities when applying attention to each element, and elements of the same agent are not distinguished from elements of other agents; for example, when computing attention for $\mathbf{x}_n^t$, both $\mathbf{x}_n^{t+1}$ and $\mathbf{x}_m^{t+1}$ are treated the same, disregarding the fact that $\mathbf{x}_n^{t+1}$ is from the same agent while $\mathbf{x}_m^{t+1}$ is from a different agent. Below, we present the solutions to these two problems – (1) time encoder and (2) agent-aware attention.

Time Encoder. To inform AgentFormer about the timestep associated with each element in the trajectory sequence, we employ a time encoder similar to the positional encoding in the original Transformer. Instead of encoding the position of each element based on its index in the sequence, we compute a timestamp feature based on the timestep t of the element. The timestamp uses the same sinusoidal design as the positional encoding. Let us take the past trajectory sequence $\mathbf{X}$ as an example. For each element $\mathbf{x}_n^t$, the timestamp feature $\boldsymbol{\tau}_n^t \in \mathbb{R}^{d_\tau}$ is defined as

$$\boldsymbol{\tau}_n^t(k) = \begin{cases} \sin((t+H)/10000^{k/d_\tau}), & k \text{ is even} \\ \cos((t+H)/10000^{(k-1)/d_\tau}), & k \text{ is odd} \end{cases}$$

where $\boldsymbol{\tau}_n^t(k)$ denotes the k-th feature of $\boldsymbol{\tau}_n^t$ and d_τ is the feature dimension of the timestamp. The time encoder outputs a timestamped sequence $\bar{\mathbf{X}}$ and each element $\bar{\mathbf{x}}_n^t \in \mathbb{R}^{d_\tau}$ in $\bar{\mathbf{X}}$ is computed as $\bar{\mathbf{x}}_n^t = \mathbf{W}_2(\mathbf{W}_1\mathbf{x}_n^t \oplus \boldsymbol{\tau}_n^t)$ where $\mathbf{W}_1 \in \mathbb{R}^{d_\tau \times d_s}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_\tau \times 2d_\tau}$ are weight matrices and $\oplus$ denotes concatenation.

Agent-Aware Attention. To preserve agent information in the trajectory sequence, it may be tempting to employ a similar strategy to the time encoder, such as an agent encoder that assigns an agent index-based encoding to each element in the sequence. However, using such agent encoding is not effective as we will show in the experiments. The reason is that, different from time which is naturally ordered, there is no innate ordering between agents, and assigning encodings based on agent indices will break the required permutation invariance of agents and create artificial dependencies on agent indices in the model.

We tackle the loss of agent information from a different angle by proposing a novel agent-aware attention mechanism. The agent-aware attention takes as input keys $\mathbf{K}$, queries $\mathbf{Q}$ and values $\mathbf{V}$, each of which uses the sequence representation of multi-agent trajectories. As an example, let the keys $\mathbf{K}$ and values $\mathbf{V}$ be the past trajectory sequence $\mathbf{X} \in \mathbb{R}^{L_p \times d_s}$, and let the queries $\mathbf{Q}$ be the future trajectory sequence $\mathbf{Y} \in \mathbb{R}^{L_f \times d_p}$. Recall that $\mathbf{X}$ is of length $L_p = N \times (H+1)$ as $\mathbf{X}$ contains the trajectory features of N agents of $H+1$ past timesteps; $\mathbf{Y}$ is of length $L_f = N \times T$ containing trajectory features of T future timesteps. The output of agent-aware attention is computed as

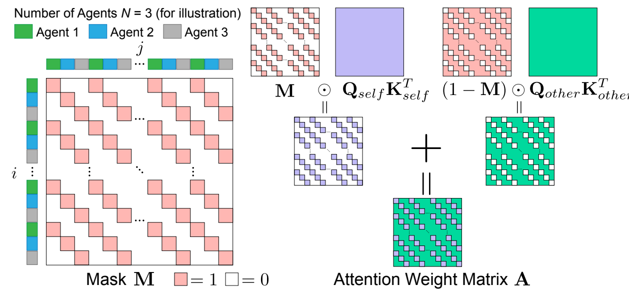

Figure 2. **Illustration of agent-aware attention.** The mask $\mathbf{M}$ allows the attention weights in $\mathbf{A}$ to be computed differently based on whether the i-th query and j-th key belong to the same agent.

$$\text{AgentAwareAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{A}}{\sqrt{d_k}}\right) V \quad (1)$$

$$\mathbf{A} = \mathbf{M} \odot (\mathbf{Q}_{self}\mathbf{K}_{self}^T) + (1-\mathbf{M}) \odot (\mathbf{Q}_{other}\mathbf{K}_{other}^T) \quad (2)$$

$$\mathbf{Q}_{self} = \mathbf{Q}\mathbf{W}_{self}^Q, \qquad \mathbf{K}_{self} = \mathbf{K}\mathbf{W}_{self}^K \quad (3)$$

$$\mathbf{Q}_{other} = \mathbf{Q}\mathbf{W}_{other}^Q, \quad \mathbf{K}_{other} = \mathbf{K}\mathbf{W}_{other}^K \quad (4)$$

where $\odot$ denotes element-wise product and we use two sets of projections $\{\mathbf{W}_{self}^Q, \mathbf{W}_{self}^K\}$ and $\{\mathbf{W}_{other}^Q, \mathbf{W}_{other}^K\}$ to generate projected keys $\mathbf{K}_{self}, \mathbf{K}_{other} \in \mathbb{R}^{L_p \times d_k}$ and queries $\mathbf{Q}_{self}, \mathbf{Q}_{other} \in \mathbb{R}^{L_f \times d_k}$ with key (query) dimension d_k. Each element A_{ij} in the attention weight matrix $\mathbf{A}$ represents the attention weight between the i-th query $\mathbf{q}_i$ and j-th key $\mathbf{k}_j$. As illustrated in Fig. 2, when computing the attention weight matrix $\mathbf{A} \in \mathbb{R}^{L_f \times L_p}$, we also use a mask $\mathbf{M} \in \mathbb{R}^{L_f \times L_p}$ which is defined as

$$M_{ij} = \mathbb{1}(i \bmod N = j \bmod N) \quad (5)$$

where M_{ij} denotes each element inside the mask $\mathbf{M}$ and $\mathbb{1}(\cdot)$ denotes the indicator function. As $\cdot \bmod N$ computes the agent index of a query/key, M_{ij} equals to one if the i-th query $\mathbf{q}_i$ and j-th key $\mathbf{k}_j$ belongs to the same agent, and M_{ij} equals to zero otherwise, as shown in Fig. 2. Using the mask $\mathbf{M}$, Eq. (2) computes each element A_{ij} of the attention weight matrix $\mathbf{A}$ differently based on the agreement of agent identity: If $\mathbf{q}_i$ and $\mathbf{k}_j$ have the same agent identity, A_{ij} is computed using the projected queries $\mathbf{Q}_{self}$ and keys $\mathbf{K}_{self}$ designated for intra-agent attention (agent to itself); If $\mathbf{q}_i$ and $\mathbf{k}_j$ have different agent identities, A_{ij} is computed using the projected queries $\mathbf{Q}_{other}$ and keys $\mathbf{K}_{other}$ designated for inter-agent attention (agent to other agents). In this

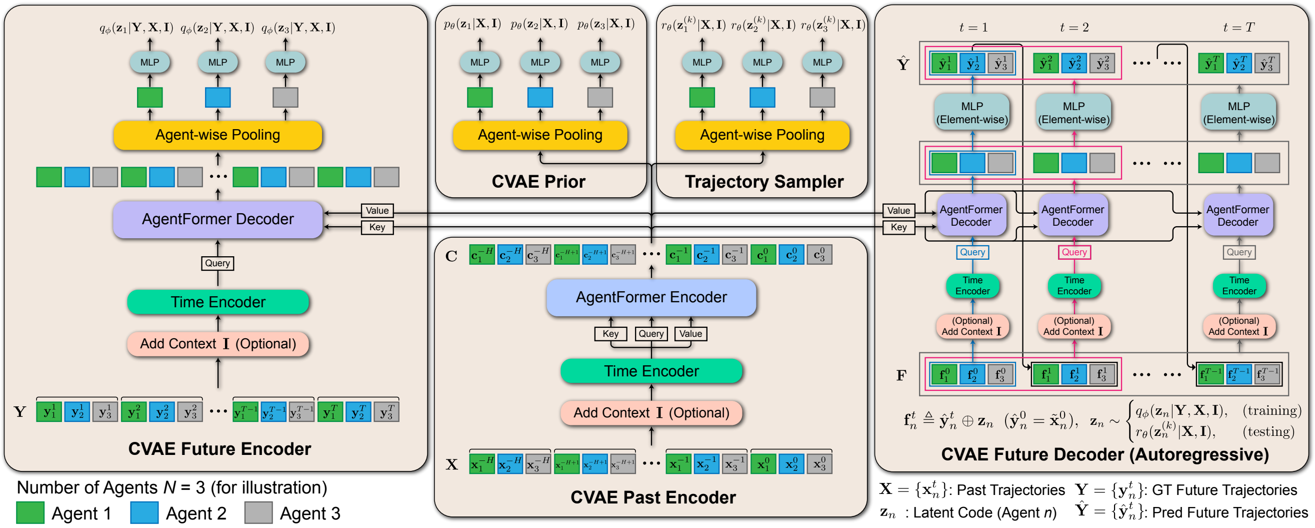

Figure 3. **Overview of our AgentFormer-based multi-agent trajectory prediction framework.**

way, the agent-aware attention learns to attend to elements of the same agent in the sequence differently than elements of other agents, thus preserving the notion of agent identity. Note that AgentFormer only uses agent-aware attention to replace the scaled dot-product attention in the original Transformer and still allows multi-head attention to learn distributed representations.

Encoding Agent Connectivity. AgentFormer can also encode rule-based agent connectivity information by masking out the attention weights between unconnected agents. Specifically, we define that two agents n and m are connected if their distance D_{nm} at the current time ($t = 0$) is smaller than a threshold η. If agents n and m are not connected, we set the attention weight $A_{ij} = -\infty$ between any query $\mathbf{q}_i$ of agent n and any key $\mathbf{k}_j$ of agent m.

3.2. Multi-Agent Prediction with AgentFormer

Having introduced AgentFormer for modeling temporal and social relations, we are now ready to apply it in our multi-agent trajectory prediction framework based on CVAEs. As discussed at the start of Sec. 3, the goal of multi-agent trajectory prediction is to model the future trajectory distribution $p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{I})$ conditioned on past trajectories $\mathbf{X}$ and contextual information $\mathbf{I}$. To account for stochasticity and multi-modality in each agent's future behavior, we introduce latent variables $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ where $\mathbf{z}_n \in \mathbb{R}^{d_z}$ represents the latent intent of agent n. We can then rewrite the future trajectory distribution as

$$p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{I}) = \int p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X},\mathbf{I}) p_\theta(\mathbf{Z}|\mathbf{X},\mathbf{I}) d\mathbf{Z}\,, \tag{6}$$

where $p_\theta(\mathbf{Z}|\mathbf{X},\mathbf{I}) = \prod_{n=1}^{N} p_\theta(\mathbf{z}_n|\mathbf{X},\mathbf{I})$ is a conditional Gaussian prior factorized over agents and $p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X},\mathbf{I})$ is a conditional likelihood model. To tackle the intractable integral in Eq. (6), we use the negative evidence lower bound (ELBO) $\mathcal{L}_{elbo}$ in the CVAE as our loss function:

$$\begin{aligned}\mathcal{L}_{elbo} = &- \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X},\mathbf{I})}[\log p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X},\mathbf{I})] \\ &+ \mathrm{KL}(q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X},\mathbf{I}) \| p_\theta(\mathbf{Z}|\mathbf{X},\mathbf{I}))\,,\end{aligned} \tag{7}$$

where $q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X},\mathbf{I}) = \prod_{n=1}^{N} q_\phi(\mathbf{z}_n|\mathbf{Y},\mathbf{X},\mathbf{I})$ is an approximate posterior distribution factorized over agents and parametrized by ϕ. In our probabilistic formulation, the latent codes $\mathbf{Z}$ of all agents in the posterior $q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X},\mathbf{I})$ are jointly inferred from the future trajectories $\mathbf{Y}$ of all agents; similarly, the future trajectories $\mathbf{Y}$ in the conditional likelihood $p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X},\mathbf{I})$ are modeled using the latent codes $\mathbf{Z}$ of all agents. This design allows each agent's latent intent represented by $\mathbf{z}_n$ to affect not just its own future trajectory but also the future trajectories of other agents, which enables us to generate socially-aware multi-agent trajectories. Having described the probabilistic formulation, we now introduce the detailed model architecture as outlined in Fig. 3.

Encoding Context (Semantic Map). As aforementioned, our model can optionally take as input contextual information $\mathbf{I}$ if provided by the data. Here, we assume $\mathbf{I} \in \mathbb{R}^{H_0 \times W_0 \times C}$ is a semantic map around the agents at the current timestep ($t = 0$) with annotated semantic information (*e.g.*, sidewalks, crosswalks, and road boundaries). For each agent n, we rotate $\mathbf{I}$ to align with the agent's heading angle and crop an image patch $\mathbf{I}_n \in \mathbb{R}^{H \times W \times C}$ around the agent. We use a hand-designed convolutional neural network (CNN) to extract visual features $\mathbf{v}_n$ from $\mathbf{I}_n$, which will later be used by other modules in the model.

CVAE Past Encoder. The past encoder starts with the multi-agent past trajectory sequence $\mathbf{X}$. If the semantic map $\mathbf{I}$ is provided, the past encoder concatenates each element $\mathbf{x}_n^t \in \mathbf{X}$ with the corresponding visual feature $\mathbf{v}_n$

of agent n. The new sequence is then fed into the time encoder to obtain a timestamped sequence, which is then input to the AgentFormer encoder as keys, queries, and values. The output of the encoder is a past feature sequence $\mathbf{C} = \left(\mathbf{c}_1^{-H}, \dots \mathbf{c}_N^{-H}, \mathbf{c}_1^{-H+1}, \dots \mathbf{c}_N^{-H+1}, \dots, \mathbf{c}_1^0, \dots, \mathbf{c}_N^0\right)$ that summarizes the past agent trajectories $\mathbf{X}$ and context $\mathbf{I}$.

CVAE Prior. The prior module first performs an agent-wise pooling that computes a mean agent feature $\mathbf{C}_n$ from the past features across timesteps: $\mathbf{C}_n = \text{mean}(\mathbf{c}_n^{-H}, \dots, \mathbf{c}_n^0)$. We then use a multilayer perceptron (MLP) to map $\mathbf{C}_n$ to the Gaussian parameters $(\boldsymbol{\mu}_n^p, \boldsymbol{\sigma}_n^p)$ of the prior distribution $p_\theta(\mathbf{z}_n|\mathbf{X}, \mathbf{I}) = \mathcal{N}(\boldsymbol{\mu}_n^p, \text{Diag}(\boldsymbol{\sigma}_n^p)^2)$.

CVAE Future Encoder. Given the multi-agent future trajectory sequence $\mathbf{Y}$, similar to the past encoder, the future encoder appends visual features from the semantic map $\mathbf{I}$ to $\mathbf{Y}$ and feeds the resulting sequence to the time encoder to produce a timestamped sequence. The timestamped sequence is then input as queries to the AgentFormer decoder along with the past feature sequence $\mathbf{C}$ which serves as both keys and values. We use the AgentFormer decoder here because it allows the feature extraction of $\mathbf{Y}$ to condition on $\mathbf{X}$ through $\mathbf{C}$, thus effectively modeling the $\mathbf{X}$-conditioning in the posterior $q_\phi(\mathbf{Z}|\mathbf{Y}, \mathbf{X}, \mathbf{I})$. We then perform an agent-wise mean pooling across timesteps on the output sequence of the AgentFormer decoder to extract a feature for each agent. Each agent feature is then input to an MLP to obtain the Gaussian parameters $(\boldsymbol{\mu}_n^q, \boldsymbol{\sigma}_n^q)$ of the approximate posterior distribution $q_\phi(\mathbf{z}_n|\mathbf{Y}, \mathbf{X}, \mathbf{I}) = \mathcal{N}(\boldsymbol{\mu}_n^q, \text{Diag}(\boldsymbol{\sigma}_n^q)^2)$.

CVAE Future Decoder. Unlike the original Transformer decoder, our future trajectory decoder is autoregressive, which means it outputs trajectories one step at a time and feeds the currently generated trajectories back into the model to produce the trajectories of the next timestep. This design mitigates compounding errors during test time at the expense of training speed. Starting from an initial sequence $(\hat{\mathbf{y}}_1^0, \dots, \hat{\mathbf{y}}_N^0)$ where $\hat{\mathbf{y}}_n^0 = \tilde{\mathbf{x}}_n^0$ ($\tilde{\mathbf{x}}_n^0$ is the position feature inside $\mathbf{x}_n^0$), the future decoder module maps an input sequence $(\hat{\mathbf{y}}_1^0, \dots, \hat{\mathbf{y}}_N^0, \dots, \hat{\mathbf{y}}_1^{t'}, \dots, \hat{\mathbf{y}}_N^{t'})$ to an output sequence $(\hat{\mathbf{y}}_1^1, \dots, \hat{\mathbf{y}}_N^1, \dots, \hat{\mathbf{y}}_1^{t'+1}, \dots, \hat{\mathbf{y}}_N^{t'+1})$ and grows the input sequence into $(\hat{\mathbf{y}}_1^0, \dots, \hat{\mathbf{y}}_N^0, \dots, \hat{\mathbf{y}}_1^{t'+1}, \dots, \hat{\mathbf{y}}_N^{t'+1})$. By autoregressively applying the decoder T times, we obtain the output sequence $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1^1, \dots, \hat{\mathbf{y}}_N^1, \dots, \hat{\mathbf{y}}_1^T, \dots, \hat{\mathbf{y}}_N^T)$. Inside the future decoder module (Fig. 3 (Right)), we first form a feature sequence $\mathbf{F} = (\mathbf{f}_1^0, \dots, \mathbf{f}_N^0, \dots, \mathbf{f}_1^{t'}, \dots, \mathbf{f}_N^{t'})$ where $\mathbf{f}_n^t = \hat{\mathbf{y}}_n^t \oplus \mathbf{z}_n$, thus concatenating the currently generated trajectories with the corresponding latent codes. The latent codes are sampled from the approximate posterior during training but from the trajectory sampler (as discussed below) at test time. The feature sequence $\mathbf{F}$ is then concatenated with the semantic map features and timestamped before being input as queries to the AgentFormer decoder alongside the past feature sequence $\mathbf{C}$ which serves as keys and values. The AgentFormer decoder enables the future trajectories to directly attend to features of any agent at any previous timestep (*e.g.*, $\mathbf{c}_3^{-H}$ or $\hat{\mathbf{y}}_2^1$), allowing the model to effectively infer future trajectories based on the whole agent history. We use proper masking inside the AgentFormer decoder to enforce causality of the decoder output sequence. Each element of the output sequence is then passed through an MLP to generate the decoded future agent position $\hat{\mathbf{y}}_n^t$. As we use a Gaussian to model the conditional likelihood $p_\theta(\mathbf{Y}|\mathbf{Z}, \mathbf{X}, \mathbf{I}) = \mathcal{N}(\hat{\mathbf{Y}}, I/\beta)$, where I is the identity matrix and β is a weighting factor, the first term in Eq. (7) equals the mean squred error (MSE): $\mathcal{L}_{mse} = \frac{1}{2\beta}\|\mathbf{Y} - \hat{\mathbf{Y}}\|^2$.

Trajectory Sampler. We adapt a diversity sampling technique, DLow [58], to our multi-agent trajectory prediction setting and employ a trajectory sampler to produce diverse and plausible trajectories once our CVAE model is trained. The trajectory sampler generates K sets of latent codes $\{\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(K)}\}$ where each set $\mathbf{Z}^{(k)} = \{\mathbf{z}_1^{(k)}, \dots, \mathbf{z}_N^{(k)}\}$ contains the latent codes of all agents and can be decoded by the CVAE decoder into a multi-agent future trajectory sample $\hat{\mathbf{Y}}^{(k)}$. Each latent code $\mathbf{z}_n^{(k)} \in \mathbf{Z}^{(k)}$ is generated by a linear transformation of a Gaussian noise $\boldsymbol{\epsilon}_n \in \mathbb{R}^{d_z}$:

$$\mathbf{z}_n^{(k)} = \mathbf{A}_n^{(k)} \boldsymbol{\epsilon}_n + \mathbf{b}_n^{(k)}, \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, I), \tag{8}$$

where $\mathbf{A}_n^{(k)} \in \mathbb{R}^{d_z \times d_z}$ is a non-singular matrix and $\mathbf{b}_n^{(k)} \in \mathbb{R}^{d_z}$ is a vector. Eq. (8) induces a Gaussian sampling distribution $r_\theta(\mathbf{z}_n^{(k)}|\mathbf{X}, \mathbf{I})$ over $\mathbf{z}_n^{(k)}$. The distribution is conditioned on $\mathbf{X}$ and $\mathbf{I}$ because its inner parameters $\{\mathbf{A}_n^{(k)}, \mathbf{b}_n^{(k)}\}$ are generated by the trajectory sampler module (Fig. 3) through agent-wise pooling of the past feature sequence $\mathbf{C}$ and an MLP. The trajectory sampler loss is defined as

$$\begin{aligned} \mathcal{L}_{samp} &= \min_k \|\hat{\mathbf{Y}}^{(k)} - \mathbf{Y}\|^2 \\ &+ \sum_{n=1}^{N} \text{KL}(r_\theta(\mathbf{z}_n^{(k)}|\mathbf{X}, \mathbf{I}) \| p_\theta(\mathbf{z}_n|\mathbf{X}, \mathbf{I})) \\ &+ \frac{1}{K(K-1)} \sum_{k_1=1}^{K} \sum_{k_1 \neq k_2}^{K} \exp\left(-\frac{\|\hat{\mathbf{Y}}^{(k_1)} - \hat{\mathbf{Y}}^{(k_2)}\|^2}{\sigma_d}\right), \end{aligned} \tag{9}$$

where σ_d is a scaling factor. The first term encourages the future trajectory samples $\hat{\mathbf{Y}}^{(k)}$ to cover the ground truth $\mathbf{Y}$. The second KL term encourages each latent code $\mathbf{z}_n^{(k)}$ to follow the prior and be plausible; the KL can be computed analytically as both distributions inside are Gaussians. The third term encourages diversity among the future trajectory samples $\hat{\mathbf{Y}}^{(k)}$ by penalizing small pairwise distance. When training the trajectory sampler with Eq. (9), we freeze the weights of the CVAE modules. At test time, we sample latent codes $\{\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(K)}\}$ using the trajectory sampler instead of sampling from the CVAE prior and decode the latent codes into trajectory samples $\{\hat{\mathbf{Y}}^{(1)}, \dots, \hat{\mathbf{Y}}^{(K)}\}$.

4. Experiments

Datasets. We evaluate our method on well-established public datasets: the ETH [37], UCY [28], and nuScenes [3] datasets. The ETH/UCY datasets are the major benchmark for pedestrian trajectory prediction. There are five datasets in ETH/UCY, each of which contains pedestrian trajectories captured at 2.5Hz in multi-agent social scenarios with rich interaction. nuScenes is a recent large-scale autonomous driving dataset, which consists of 1000 driving scenes with each scene annotated at 2Hz. nuScenes also provides HD semantic maps with 11 semantic classes.

Metrics. We report the minimum average displacement error ADE_K and final displacement error FDE_K of K trajectory samples of each agent compared to the ground truth: $\text{ADE}_K = \frac{1}{T}\min_{k=1}^{K}\sum_{t=1}^{T}\|\hat{\mathbf{y}}_n^{t,(k)} - \mathbf{y}_n^t\|^2$, $\text{FDE}_K = \min_{k=1}^{K}\|\hat{\mathbf{y}}_n^{T,(k)} - \mathbf{y}_n^T\|^2$, where $\hat{\mathbf{y}}_n^{t,(k)}$ denotes the future position of agent n at time t in the k-th sample and $\mathbf{y}_n^T$ is the corresponding ground truth. ADE_K and FDE_K are the standard metrics for trajectory prediction [15, 43, 44, 38, 5].

Evaluation Protocol. For the ETH/UCY datasets, we adopt a leave-one-out strategy for evaluation, following prior work [15, 43, 44, 34, 54]. We forecast 2D future trajectories of 12 timesteps (4.8s) based on observed trajectories of 8 timesteps (3.2s). Similar to most prior works, we do not use any semantic/visual information for ETH/UCY for fair comparisons. All metrics are computed with $K = 20$ samples. For the nuScenes dataset, following prior work [38, 5, 8, 33], we use the vehicle-only train-val-test split provided by the nuScenes prediction challenge and predict 2D future trajectories of 12 timesteps (6s) based on observed trajectories of 4 timesteps (2s). We report results with metrics computed using $K = 1, 5$ and 10 samples.

Implementation Details. For all datasets, we represent trajectories in a scene-centered coordinate where the origin is the mean position of all agents at $t = 0$. The future decoder in Fig. 3 outputs the offset to the agent's current position $\tilde{\mathbf{x}}_n^0$, so $\tilde{\mathbf{x}}_n^0$ is added to obtain $\hat{\mathbf{y}}_n^t$ for each element in the output sequence. Following prior work [44, 54], random rotation of the scene is adopted for data augment. Our multi-agent prediction model (Fig. 3) uses two stacks (defined in [46]) of identical layers in each AgentFormer encoder/decoder with 0.1 dropout rate. The dimensions d_k, d_v, d_τ of keys, queries, and timestamps in AgentFormer are all set to 256, and the hidden dimension of feedforward layers is 512. The number of heads for multi-head agent-aware attention is 8. All MLPs in the model have hidden dimensions (512, 256). For the CVAE, the latent code dimension d_z is 32, the coefficient β of the MSE loss equals 1, and we clip the maximum value of the KL term in L_{elbo} (Eq. (7)) down to 2. We also use the variety loss in SGAN [15] in addition to L_{elbo}. The agent connectivity threshold η is set to 100. We train the CVAE model using the Adam optimizer [20] for 100 epochs on ETH/UCY and nuScenes. We use an initial learning rate of 10^{-4} and halve the learning rate every 10 epochs. More details including the CNN for encoding semantic maps and the training procedure of the trajectory sampler can be found in the supplementary materials.

Method	$\text{ADE}_{20}/\text{FDE}_{20}\downarrow$ (m), $K = 20$ Samples					
	ETH	Hotel	Univ	Zara1	Zara2	Average
SGAN [15]	0.81/1.52	0.72/1.61	0.60/1.26	0.34/0.69	0.42/0.84	0.58/1.18
SoPhie [43]	0.70/1.43	0.76/1.67	0.54/1.24	0.30/0.63	0.38/0.78	0.54/1.15
Transformer-TF [12]	0.61/1.12	0.18/0.30	0.35/0.65	0.22/0.38	0.17/0.32	0.31/0.55
STAR [54]	0.36/0.65	0.17/0.36	0.31/0.62	0.26/0.55	0.22/0.46	0.26/0.53
PECNet [34]	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++ [44]	0.39/0.83	0.12/0.21	**0.20/0.44**	**0.15**/0.33	**0.11**/0.25	0.19/0.41
Ours (AgentFormer)	**0.26/0.39**	**0.11/0.14**	0.26/0.46	**0.15/0.23**	0.14/**0.24**	**0.18/0.29**

Table 1. **Baseline comparisons** on the ETH/UCY datasets. Our method outperforms previous art with large FDE improvements.

Method	$K = 5$ Samples		$K = 10$ Samples		$K = 1$
	$\text{ADE}_5\downarrow$	$\text{FDE}_5\downarrow$	$\text{ADE}_{10}\downarrow$	$\text{FDE}_{10}\downarrow$	$\text{FDE}_1\downarrow$
MTP [8]	2.93	-	2.93	-	9.23
MultiPath [5]	2.32	-	1.96	-	9.19
CoverNet [38]	1.96	-	1.48	-	9.26
DSF-AF [33]	2.06	4.67	1.66	3.71	-
DLow-AF [58]	2.11	4.70	1.78	3.58	-
Trajectron++ [44]	1.88	-	1.51	-	9.52
Ours (AgentFormer)	**1.59**	**3.14**	**1.31**	**2.48**	**6.45**

Table 2. **Baseline comparisons** on the nuScenes dataset. Our method outperforms prior state-of-the-art methods consistently for 1, 5 and 10 samples. Symbol "-" means results are not available.

4.1. Results

Baseline Comparisons. On the ETH/UCY datasets, we compare our approach with current state-of-the-art methods – Trajectron++ [44], PECNet [34], STAR [54], and Transformer-TF [12] – as well as common baselines – SGAN [15] and Sophie [43]. The performance of all methods is summarized in Table 1, where we use officially-reported results for the baselines. We can observe that our AgentFormer significantly outperforms the baselines in prediction accuracy as measured by ADE and FDE. Particularly, our method reduces the FDE of the current state of the art, Trajectron++, from 0.41 to 0.29, achieving a 41% increase in performance. As FDE measures the final displacement error of predicted trajectories, it places more emphasis on a method's ability to predict distant futures than ADE. We believe the strong performance of our method in FDE can be attributed to the design of AgentFormer, which can model long-range trajectory dependencies effectively by directly attending to features of any agent at any previous timestep when inferring an agent's future position.

Compared to ETH/UCY, the trajectories in nuScenes are much longer as we evaluate with a longer time horizon (6s) and vehicles are much faster than pedestrians. Thus, nuScenes presents a different challenge for multi-agent prediction methods. On the nuScenes dataset, we evaluate our approach against state-of-the-art vehicle prediction methods – Trajectron++ [44], MTP [8], MultiPath [5], Cover-

Model		ADE$_{20}$/FDE$_{20}$ ↓ (m), $K = 20$ Samples					
Social	Temporal	ETH	Hotel	Univ	Zara1	Zara2	Average
GCN	LSTM	0.33/0.43	0.17/0.27	0.28/0.50	0.19/0.33	0.19/0.34	0.23/0.37
GCN	TF	0.36/0.50	0.12/0.18	0.27/0.50	0.18/0.31	0.16/0.28	0.22/0.35
TF	LSTM	0.35/0.43	0.16/0.24	0.27/0.48	0.18/0.30	0.18/0.33	0.23/0.36
TF	TF	0.35/0.45	0.12/0.19	0.28/0.52	0.19/0.32	0.15/0.27	0.22/0.35
Joint Socio-Temporal		ETH	Hotel	Univ	Zara1	Zara2	Average
Ours w/o joint latent		0.31/**0.39**	**0.11**/0.15	0.28/0.51	0.18/0.30	0.16/0.29	0.21/0.33
Ours w/o AA attention		0.31/0.40	0.13/0.20	0.30/0.53	0.18/0.28	0.19/0.34	0.22/0.35
Ours w/ agent encoding		0.30/0.40	0.13/0.20	0.30/0.54	0.18/0.29	0.19/0.34	0.22/0.35
Ours (AgentFormer)		**0.26/0.39**	**0.11/0.14**	**0.26/0.46**	**0.15/0.23**	**0.14/0.24**	**0.18/0.29**

Table 3. **Ablation studies** on the ETH/UCY datasets. "TF" means Transformer and "AA Attention" denotes agent-aware attention.

Model		$K = 5$ Samples		$K = 10$ Samples	
Social	Temporal	ADE$_5$ ↓	FDE$_5$ ↓	ADE$_{10}$ ↓	FDE$_{10}$ ↓
GCN	LSTM	1.97	3.97	1.58	2.93
GCN	TF	1.74	3.52	1.39	2.59
TF	LSTM	1.79	3.65	1.48	2.76
TF	TF	1.98	4.20	1.54	3.07
Joint Socio-Temporal		ADE$_5$ ↓	FDE$_5$ ↓	ADE$_{10}$ ↓	FDE$_{10}$ ↓
Ours w/o semantic map		1.73	3.57	1.46	2.81
Ours w/o joint latent		1.66	3.28	1.40	2.60
Ours w/o AA attention		1.82	3.70	1.49	2.83
Ours w/ agent encoding		1.83	3.70	1.50	2.82
Ours (AgentFormer)		**1.59**	**3.14**	**1.31**	**2.48**

Table 4. **Ablation studies** on the nuScenes dataset. "TF" means Transformer and "AA Attention" denotes agent-aware attention.

Net [38], DSF-AF [33], and DLow-AF [58]. We report the performance of all methods in Table 2, where the results of Trajectron++ are taken from the nuScenes prediction challenge leaderboard, the performance of DLow-AF is from [33], and we also use the officially-reported results for the other baselines. The FDE of some baselines is not available since the number has not been reported. We can see that our approach, AgentFormer, outperforms the baselines consistently in ADE and FDE for different settings ($K = 1, 5$ and 10 samples). Notably, our method reduces the state-of-the-art ADE$_5$ from 1.88 to 1.59. Among methods that report FDE, our approach achieves an FDE$_1$ of 6.45, which is significantly lower than 9.19 from the second-best method and achieves a 42% performance improvement.

Ablation Studies. We further perform extensive ablation studies on ETH/UCY and nuScenes to investigate the contribution of key technical components in our method. The first ablation study explores variants of our method that use separate social and temporal models to replace our joint socio-temporal model, AgentFormer, in our multi-agent prediction framework. We choose GCN [23] or Transformer (TF) as the social model, and LSTM or Transformer as the temporal model. In total, there are 4 (2×2) combinations of social and temporal models. The ablation results are summarized in the first group of Table 3 and 4. It is evident that all combinations of separate social and temporal models lead to inferior performance compared to our method which models the social and temporal dimensions jointly.

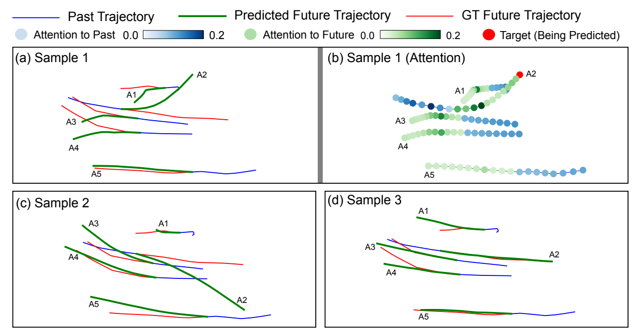

Figure 4. **(a,c,d)** Three samples of forecasted multi-agent futures (green) via our method. **(b)** Attention visualization for sample 1.

The second ablation study investigates the role of (1) joint latent intent modeling, (2) agent-aware attention, and (3) semantic maps, and we denote the corresponding variants as "w/o joint latent", "w/o AA attention", and "w/o semantic map". We further test a variant "w/ agent encoding" where we replace agent-aware attention with agent encoding. The results are reported in the second group of Table 3 and 4. We can see that all variants lead to considerably worse performance compared to our full method. In particular, the variants "w/o AA attention" and "w/ agent encoding" result in pronounced performance drop, which indicates that agent-aware attention is essential in our method and alternatives like agent encoding are not effective.

Trajectory Visualization. Fig. 4 (a,c,d) shows three samples of forecasted multi-agent futures of the same scene via our method. We can see that the samples correspond to different modes of socially-aware and non-colliding trajectories, and exhibit behaviors like following (A3 & A4) and collision avoidance (A1 & A2 in (a), A2 & A3 in (c)). Fig. 4 (b) visualizes the attention of sample 1 and shows that, when predicting the target (red), the model pays more attention to key timesteps (turning point) of adjacent agents and also spreads out attention to the target's past timesteps to reason about the dynamics and curvature of its trajectory.

5. Conclusion

In this paper, we proposed a new Transformer, AgentFormer, that can simultaneously model the time and social dimensions of multi-agent trajectories using a sequence representation. To preserve agent identities in the sequence, we proposed a novel agent-aware attention mechanism that can attend to features of the same agent differently than features of other agents. Based on AgentFormer, we presented a stochastic multi-agent trajectory prediction framework that jointly models the latent intent of all agents to produce diverse and socially-aware multi-agent future trajectories. Experiments demonstrated that our method significantly improved state-of-the-art performance on challenging pedestrian and autonomous driving datasets.

Acknowledgments. This work is supported by the Qualcomm Innovation Fellowship.

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. 2, 3

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 3

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 2, 7

[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. 3

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, pages 86–99. PMLR, 2020. 7

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 3

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2, 3

[8] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 7

[9] 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. 3

[10] 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. 3

[11] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 3

[12] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. *ICPR*, 2020. 3, 7

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 3

[14] Jiaqi Guan, Ye Yuan, Kris M Kitani, and Nicholas Rhinehart. Generative hybrid representations for activity forecasting with no-regret learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 173–182, 2020. 3

[15] 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. 2, 3, 7

[16] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 3

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 3

[18] 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, 3

[19] 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, 3

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 3

[22] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018. 3

[23] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2, 3, 8

[24] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020. 2

[25] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian D Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Advances in Neural Information Processing Systems 2019*. Neural Information Processing Systems (NIPS), 2019. 2, 3

[26] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. 3

[27] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire:

Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 3

[28] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007. 2, 7

[29] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[30] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. End-to-end contextual perception and prediction with interaction transformer. *arXiv preprint arXiv:2008.05927*, 2020. 3

[31] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 3

[32] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 2

[33] Yecheng Jason Ma, Jeevana Priya Inala, Dinesh Jayaraman, and Osbert Bastani. Diverse sampling for normalizing flow based trajectory forecasting. *arXiv preprint arXiv:2011.15084*, 2020. 7, 8

[34] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, pages 759–776. Springer, 2020. 7

[35] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015. 2

[36] Jeremy Morton, Tim A Wheeler, and Mykel J Kochenderfer. Analysis of recurrent neural networks for probabilistic modeling of driver behavior. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1289–1298, 2016. 3

[37] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268. IEEE, 2009. 2, 7

[38] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 7, 8

[39] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015. 3

[40] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 3

[41] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019. 3

[42] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020. 3

[43] 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. 3, 7

[44] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. *arXiv preprint arXiv:2001.03093*, 2020. 2, 3, 7

[45] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2019. 3

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017. 2, 3, 7

[47] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pages 4601–4607. IEEE, 2018. 3

[48] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007. 3

[49] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020. 3

[50] Xinshuo Weng, Ye Yuan, and Kris Kitani. Joint 3d tracking and forecasting with graph neural network and diversity sampling. *arXiv preprint arXiv:2003.07847*, 2020. 3

[51] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE, 2018. 2

[52] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015. 2

[53] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. 3

[54] 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. 3, 7

[55] Ye Yuan and Kris Kitani. 3d ego-pose estimation via imitation learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 735–750, 2018. 2

[56] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. *arXiv preprint arXiv:1907.04967*, 2019. 3

[57] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10082–10092, 2019. 3

[58] Ye Yuan and Kris Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *European Conference on Computer Vision*, pages 346–364. Springer, 2020. 6, 7, 8

[59] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *Advances in Neural Information Processing Systems*, 2020. 3

[60] 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

[61] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019. 3