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

Ye Yuan\textsuperscript{1} Xinshuo Weng\textsuperscript{1} Yanglan Ou\textsuperscript{2} Kris Kitani\textsuperscript{1}
\textsuperscript{1}Carnegie Mellon University \textsuperscript{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) \textit{time dimension}, where we model the influence of past agent states over future states; (2) \textit{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 prefer a method that allows an agent’s state at one time to \textit{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 substantially 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, \textit{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, \textit{i.e.}, each agent has its latent intent unobserved by the system (\textit{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.

Multi-agent social interaction modeling involves two key dimensions as illustrated in Fig. 1 (Top): (1) \textit{time dimension}, where we model how past agent states (positions and velocities) influence future agent states; (2) \textit{social dimension}, where we model how each agent’s state affects the
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 [47]) 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 [45, 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 [38] 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. 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 substantially 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 [52, 35], image captioning [53], machine translation [32], human pose estimation [56, 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, 61]. Prior work has also used RNNs to model the temporal dynamics of 3D human
pose \cite{11, 58, 60}. With the invention of Transformers and positional encoding \cite{47}, 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 \cite{9, 26, 54}. Beyond NLP, numerous visual Transformers have been proposed to tackle vision tasks, such as image classification \cite{10}, object detection \cite{4}, and instance segmentation \cite{50}. Recently, Transformers have also been used for trajectory forecasting. Transformer-TF \cite{12} applies the standard Transformer to predict the future trajectories of each agent independently. STAR \cite{55} uses separate temporal and spatial Transformers to forecast multi-agent trajectories. Interaction Transformer \cite{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 \cite{16}, Gaussian process (GP) \cite{49}, and RNNs \cite{1, 36, 48}. A thorough review of these deterministic methods is provided in \cite{43}. 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 \cite{21, 13, 40} such as conditional variational autoencoders (CVAEs) \cite{27, 57, 19, 46, 51, 45}, generative adversarial networks (GANs) \cite{15, 44, 25, 62}, and normalizing flows (NFs) \cite{41, 42, 14}. Most of these methods follow a seq2seq structure \cite{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 \cite{17, 7} and Transformers \cite{47} are the preferred temporal models \cite{18, 1, 55}, graph neural networks (GNNs) \cite{23, 31} are often employed as the social models for interaction modeling \cite{22, 29, 25}. One popular type of methods \cite{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 \cite{45, 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 \(X^t = (x_1^t, x_2^t, \ldots, x_N^t)\), where \(x_n^t \in \mathbb{R}^d\) 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 \(X = (X^{-H}, X^{-H+1}, \ldots, X^0)\) which includes the joint state at all \(H + 1\) observed timesteps. Similarly, the joint state of all \(N\) agents at future time \(t (t > 0)\) is denoted as \(Y^t = (y_1^t, y_2^t, \ldots, y_N^t)\), where \(y_n^t \in \mathbb{R}^d\) is the future position of agent \(n\) at time \(t\). We denote the future trajectories of all \(N\) agents over \(T\) future timesteps as \(Y = (Y^1, Y^2, \ldots, Y^T)\). Depending on the data, optional contextual information \(I\) 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 \(p_Y(Y|X, I)\) where \(\theta\) 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 \cite{47} 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 \(X\) can be denoted as a sequence \(X = (x_1^{-H}, x_2^{-H}, x_3^{-H+1}, \ldots, x_N^{-H+1}, x_1^0, x_2^0, \ldots, x_N^0)\) of length \(L_p = N \times (H + 1)\). Similarly, the future multi-agent trajectories can also be represented as a sequence
\[ Y = (y_1^1, \ldots, y_N^1, y_2^1, \ldots, y_N^2, \ldots, y_1^T, \ldots, y_N^T) \] 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., \( x_n^t \)) w.r.t. other elements in the sequence; for instance, \( x_n^t \) does not know \( x_n^{t+1} \) is a feature of the same timestep while \( x_n^{t+1} \) is a feature of the next timestep; (2) loss of agent information, since Transformers do not consider agent identities and no innate ordering between agents, and assigning encodings that we can directly apply standard Transformers to these agents; for example, when computing attention for \( x_n^t \), the keys \( K \) and values \( V \) are the past trajectory sequence \( X \in \mathbb{R}^{L_p \times d_k} \), and let the queries \( Q \) be the future trajectory sequence \( Y \in \mathbb{R}^{L_f \times d_k} \). Recall that \( X \) is of length \( L_p = N \times (H + 1) \) as \( X \) contains the trajectory features of \( N \) agents of \( H + 1 \) past timesteps; \( Y \) is of length \( L_f = N \times T \) containing trajectory features of \( T \) future timesteps. The output of agent-aware attention is computed as

\[
\text{AgentAwareAttention}(Q, K, V) = \text{softmax} \left( \frac{A}{\sqrt{d_k}} \right) V
\]

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

\[
M_{ij} = \mathbb{1}(i \mod N = j \mod N)
\]

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

**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 \( X \) as an example. For each element \( x_n^t \), the timestamp feature \( \tau_n^t \in \mathbb{R}^{d_r} \) is defined as

\[
\tau_n^t(k) = \begin{cases} 
\sin((t + H)/10000k/d_r), & k \text{ is even} \\
\cos((t + H)/10000(k-1)/d_r), & k \text{ is odd} 
\end{cases}
\]

where \( \tau_n^t(k) \) denotes the \( k \)-th feature of \( \tau_n^t \) and \( d_r \) is the feature dimension of the timestamp. The time encoder outputs a timestamped sequence \( X \) and each element \( x_n^t \in \mathbb{R}^{d_r} \) in \( X \) is computed as \( x_n^t = W_2(W_1x_n^t \oplus \tau_n^t) \) where \( W_1 \in \mathbb{R}^{d_r \times d_r} \) and \( W_2 \in \mathbb{R}^{d_r \times 2d_r} \) 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 \( K \), queries \( Q \) and values \( V \), each of which uses the sequence representation of multi-agent trajectories. As an example, let the keys \( K \) and values \( V \) be the past trajectory sequence \( X \in \mathbb{R}^{L_p \times d_k} \), and let the queries \( Q \) be the future trajectory sequence \( Y \in \mathbb{R}^{L_f \times d_k} \). The time encoder out-

![Illustration of agent-aware attention](image-url)
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 $A_{ij} = -∞$ between any query $q_i$ of agent $n$ and any key $k_j$ 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 $p_θ(Y | X, I)$ conditioned on past trajectories $X$ and contextual information $I$. To account for stochasticity and multi-modality in each agent’s future behavior, we introduce latent variables $Z = \{z_1, \ldots, z_N\}$ where $z_n \in \mathbb{R}^{d_z}$ represents the latent intent of agent $n$. We can then rewrite the future trajectory distribution as

$$p_θ(Y | X, I) = \int p_θ(Y | Z, X, I)p_θ(Z | X, I)dZ,$$  

(6)

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

$$L_{elbo} = -\mathbb{E}_{q_φ(Z|Y, X, I)}[\log p_θ(Y | Z, X, I)] + KL(q_φ(Z|Y, X, I) || p_θ(Z | X, I)),$$  

(7)

where $q_φ(Z|Y, X, I) = \prod_{n=1}^{N} q_φ(z_n | Y, X, I)$ is an approximate posterior distribution factorized over agents and parameterized by $φ$. In our probabilistic formulation, the latent codes $Z$ of all agents in the posterior $q_φ(Z|Y, X, I)$ are jointly inferred from the future trajectories $Y$ of all agents; similarly, the future trajectories $Y$ in the conditional likelihood $p_θ(Y | Z, X, I)$ are modeled using the latent codes $Z$ of all agents. This design allows each agent’s latent intent represented by $z_n$ 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 $I$ if provided by the data. Here, we assume $I \in \mathbb{R}^{H_0 \times W_0 \times C}$ 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 $I$ to align with the agent’s heading angle and crop an image patch $I_n \in \mathbb{R}^{H \times W \times C}$ around the agent. We use a hand-designed convolutional neural network (CNN) to extract visual features $v_n$ from $I_n$, 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 $X$. If the semantic map $I$ is provided, the past encoder concatenates each element $x_n^t \in X$ with the corresponding visual feature $v_n$. 

**CVAE Future Encoder.** The future encoder starts with the multi-agent future trajectory sequence $Y$. The encoder consists of encoder $CVAE$ and decoder $AgentFormer$. The encoder $CVAE$ encodes the past trajectory sequence into a latent code $Z$ and the decoder $AgentFormer$ generates the future trajectory sequence based on the latent code $Z$.
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 \( \mathbf{C} = (\mathbf{c}_1^{-H}, \ldots, \mathbf{c}_N^{-H}, \mathbf{c}_1^{-H+1}, \ldots, \mathbf{c}_N^{-H+1}, \ldots, \mathbf{c}_1^0, \ldots, \mathbf{c}_N^0) \) 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: \( \mathbf{C}_n = \text{mean}(\mathbf{c}_1^{-H}, \ldots, \mathbf{c}_N^{-H}) \). We then use a multilayer perceptron (MLP) to map \( \mathbf{C}_n \) to the Gaussian parameters \((\mu_n^p, \sigma_n^p)\) of the prior distribution \( p_0(\mathbf{z}_n|\mathbf{X}, \mathbf{I}) = \mathcal{N}(\mu_n^p, \text{Diag}(\sigma_n^p)^2) \).

**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 to produce a timestamped sequence. The timestamped sequence \( \mathbf{Y} \) is then passed through a future trajectory decoder \( \hat{\mathbf{Y}} \) to regressesively apply the decoder output sequence.

**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 \((\hat{y}_1^0, \ldots, \hat{y}_N^0)\) where \( \hat{y}_n^0 = x_1^0(\mathbf{x}_n^0) \) is the position feature inside \( \mathbf{x}_n^0 \), the future decoder module maps an input sequence \((\hat{y}_1^0, \ldots, \hat{y}_N^0, \ldots, \hat{y}_1^{t'}, \ldots, \hat{y}_N^{t'})\) to an output sequence \((\hat{y}_1^1, \ldots, \hat{y}_N^1, \ldots, \hat{y}_1^{t'+1}, \ldots, \hat{y}_N^{t'+1})\) and grows the input sequence into \((\hat{y}_1^1, \ldots, \hat{y}_N^1, \ldots, \hat{y}_1^{t'+1}, \ldots, \hat{y}_N^{t'+1})\). By autoregressively applying the decoder \( T \) times, we obtain the output sequence \( \hat{\mathbf{Y}} = (\hat{y}_1^1, \ldots, \hat{y}_N^1, \ldots, \hat{y}_1^T, \ldots, \hat{y}_N^T) \). Inside the future decoder module (Fig. 3 (Right)), we first form a feature sequence \( \mathbf{F} = (\mathbf{f}_1^1, \ldots, \mathbf{f}_N^1, \ldots, \mathbf{f}_1^T, \ldots, \mathbf{f}_N^T) \) where \( \mathbf{f}_n^t = \mathbf{y}_n^t \oplus \mathbf{z}_n \), 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 \( \mathbf{y}_3^t \)), 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 \( p_0(\mathbf{Y}|\mathbf{Z}, \mathbf{X}, \mathbf{I}) = \mathcal{N}(\mathbf{Y}, \mathbf{I}/\beta) \), where \( \mathbf{I} \) is the identity matrix and \( \beta \) is a weighting factor, the first term in Eq. (7) equals the mean squared error (MSE): \( \mathcal{L}_{\text{mse}} = \frac{1}{2\beta} || \mathbf{Y} - \hat{\mathbf{Y}} ||^2 \).

**Trajectory Sampler.** We adopt a diversity sampling technique, DLow [59], 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 \( \{\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(K)}\} \) where each set \( \mathbf{Z}^{(k)} = \{\mathbf{z}_1^{(k)}, \ldots, \mathbf{z}_N^{(k)}\} \) 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 \( \epsilon_n \in \mathbb{R}^{d_z} \):

\[
\mathbf{z}_n^{(k)} = \mathbf{A}_n^{(k)} \epsilon_n + \mathbf{b}_n^{(k)}, \quad \epsilon_n \sim \mathcal{N}(0, \mathbf{I}),
\]

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 \( p_\theta(\mathbf{z}_n^{(k)}|\mathbf{X}, \mathbf{I}) \) 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

\[
\mathcal{L}_{\text{asmp}} = \min_k \| \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \|^2 + \sum_{n=1}^{N} \| \text{KL}(r_\theta(\mathbf{z}_n^{(k)}|\mathbf{X}, \mathbf{I})||p_0(\mathbf{z}_n|\mathbf{X}, \mathbf{I})) \|
\]

\[
+ \frac{1}{K(K-1)} \sum_{k_1=1}^{K} \sum_{k_2=1, k_2 \neq k_1}^{K} \exp\left(-\frac{\| \hat{\mathbf{Y}}^{(k_1)} - \hat{\mathbf{Y}}^{(k_2)} \|^2}{\sigma_d} \right),
\]

where \( \sigma_d \) 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 \( \{\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(K)}\} \) using the trajectory sampler instead of sampling from the CVAE prior and decode the latent codes into trajectory samples \( \{\hat{\mathbf{Y}}^{(1)}, \ldots, \hat{\mathbf{Y}}^{(K)}\} \).
4. Experiments

Datasets. We evaluate our method on well-established public datasets: the ETH [38], 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 $\text{ADE}_K$ and final displacement error $\text{FDE}_K$ of $K$ trajectory samples of each agent compared to the ground truth:

$$\text{ADE}_K = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T} \| \hat{y}_{n,t}^{(k)} - y_{n,t} \|^2,$$

$$\text{FDE}_K = \min_{k=1}^{K} \sum_{t=T}^{T+K-1} \| \hat{y}_{n,t}^{(k)} - y_{n,t} \|^2,$$

where $\hat{y}_{n,t}^{(k)}$ denotes the future position of agent $n$ at time $t$ in the $k$-th sample and $y_{n,t}$ is the corresponding ground truth. $\text{ADE}_K$ and $\text{FDE}_K$ are the standard metrics for trajectory prediction [15, 44, 45, 39, 5].

Evaluation Protocol. For the ETH/UCY datasets, we adopt a leave-one-out strategy for evaluation, following prior work [15, 44, 45, 34, 55]. 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 [39, 5, 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 $\hat{x}_{n,0}$, so $\hat{x}_{n}^0$ is added to obtain $\hat{y}_{n,t}$ for each element in the output sequence. Following prior work [45, 55], random rotation of the scene is adopted for data augment. Our multi-agent prediction model (Fig. 3) uses two stacks (defined in [47]) of identical layers in each AgentFormer encoder/decoder with 0.1 dropout rate. The dimensions $d_k, d_v, d_r$ 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 $\beta$ 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 $\eta$ is set to 100. We train the CVAE model using the Adam optimizer [20] for 100 epochs on ETH/UCY and nuScenes. 

Implementation Details.

Evaluation Protocol.

Implementation Details.
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 substan-
tially 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. Mutipath: 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. 2019. 2, 3

[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. ICLR, 2020. 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 Rinehart. 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 Yingqing Song. 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 trajectory: 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, 12

[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, 12

[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. Socialbigat: 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
[53] 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

[54] 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

[55] 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

[56] 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

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

[58] 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

[59] 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

[60] 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

[61] 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

[62] 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
A. Handling a Time-Varying Number of Agents

For clarity and ease of exposition, we assume the number of agents remains the same across timesteps in the main paper. However, this assumption is not necessary, and our method can easily generalize to use cases where the number of agents changes over time due to agents going out of the scene or being missed by detection. We illustrate how to apply our method to such cases in Fig. 5. Owning to the flexible sequence representation we employ for multi-agent trajectories, we can simply remove the features of missing agents at each timestep from the sequence. The reason why we do not need to fill the missing features is that our method uses time encoding to preserve time information, unlike RNNs which have to use recurrence to encode timesteps and thus necessitate the features of all timesteps. As the number of agents is no longer \( N \) for all timesteps, the computation of the mask \( M \) in agent-aware attention needs to be changed accordingly:

\[
M_{ij} = \mathbb{I}(\text{Agent}(i) = \text{Agent}(j)) \tag{10}
\]

where \( \text{Agent}(\cdot) \) extracts the agent index of a query/key and \( \mathbb{I}(\cdot) \) denotes the indicator function. An example of mask \( M \) is shown in Fig. 5 (Right).

---

B. Additional Implementation Details

Encoding Semantic Maps. The semantic map \( I_n \in \mathbb{R}^{H \times W \times C} \) for each agent \( n \) has spatial dimensions \((100, 100)\) with 3 meters between adjacent pixels. It has \( C = 3 \) channels annotating drivable areas, road dividers, and lane dividers obtained using the official nuScenes software development kit. Since the semantic map is relatively easy to parse, we use a simple hand-designed CNN to extract visual features \( v_n \) from it. In particular, the CNN has four convolutional layers with channels \((32, 32, 32, 1)\), kernel size \((5, 5, 5, 3)\), and strides \((2, 2, 2, 1)\). A final linear layer is used to obtain a 32-dimensional feature.

Training Trajectory Sampler. The scaling factor \( \sigma_d \) in the trajectory sampler loss \( L_{samp} \) (Eq. (9) in the main paper) is set to 5 for ETH/UCY and 20 for nuScenes. We clip the maximum value of the KL term in \( L_{samp} \) down to 2. We train the trajectory sampler using the Adam optimizer [20] for 50 epochs on ETH/UCY and nuScenes. We use an initial learning rate of \( 10^{-4} \) and halve the learning rate every 5 epochs.

Ablation Study Details. We first provide details for the ablation study of separate social and temporal models (first group of Table 3 and 4 in the main paper). We first use a temporal model (LSTM or Transformer) to extract the temporal feature of each agent and then apply a social model (GCN [23] or Transformer) over the temporal features to obtain social features for each agent; final trajectories are decoded from the social features using either an LSTM or Transformer. For the GCN, we use two graph convolutional layers with channels \((256, 256)\) and residual connections within each layer. The hidden dimensions of the LSTMs are set to 256. The Transformers have two layers with key/query dimensions 256 and 8 heads; the feedforward layer has 512 hidden units, and the dropout ratio is 0.1. We use the positional encoding [47] for the temporal Transformer but not for the social Transformer as agents are permutation-invariant.

Next, we provide details for the ablation study of each key technical component (second group of Table 3 and 4 in the main paper). For the variant without joint latent modeling (“w/o joint latent”), we append the latent codes to the trajectory.
sequence after the AgentFormer decoder instead of before the decoder. In this way, the latent code of one agent will not affect the future trajectory of another agent. For the variant without the agent-aware attention (“w/o AA attention”), we replace our agent-aware attention with standard scaled dot-product attention used in the original transformer [47]. For the variant with agent encoding (“w/ agent encoding”), in addition to removing the agent aware attention, we also append an agent encoding to each element in the trajectory sequence. The agent encoding is computed similarly as the positional encoding [47] but uses the agent index instead of the position index. For the variant without semantic maps (“w/o semantic map”), we simply do not append any visual features extracted from the semantic maps to the trajectory sequence.

**Other Details.** Our models are implemented using PyTorch [37] and are trained with a single NVIDIA RTX 2080 Ti and standard CPUs. The training time is approximately one day for each dataset in ETH/UCY and three days for nuScenes.

**C. Additional Attention Visualization**

As discussed in the main paper, our method can attend to any agent at any previous timestep when predicting the future position of an agent. Here, we provide more visualization of the attention in Fig. 6 to understand the behavior of our model. Across all the examples, it is evident that when predicting the target future position of an agent, the model pays more attention to the agent’s own trajectories and recent timesteps, and it also attends more to nearby agents than distant agents.

**Figure 6. Attention Visualization on ETH/UCY.** We plot the attention to past (blue) and future (green) trajectory features of all agents when inferring a target position (red). Darker color means higher attention. When predicting the target future position of an agent, the model pays more attention to the agent’s own trajectories and recent timesteps, and it also attends more to nearby agents than distant agents.
D. Trajectory Sample Visualization

To demonstrate the importance of agent-aware attention, we also provide qualitative comparisons of our method against the variant without agent-aware attention (w/o AA attention) on the nuScenes dataset in Fig. 7. We can observe that the future trajectory samples produced by our method using agent-aware attention cover the ground truth (GT) future trajectories significantly better. Our method also produces much fewer implausible trajectories such as those going out of the road.

Figure 7. Trajectory Sample Visualization on nuScenes. We compare our method against the variant without agent-aware attention (w/o AA attention). The future trajectory samples produced by our method using agent-aware attention cover the ground truth (GT) future trajectories significantly better. Our method also produces much fewer implausible trajectories such as those going out of the road.