GTP-Force: Game-Theoretic Trajectory Prediction through Distributed Reinforcement Learning

Negar Emami, Antonio Di Maio, Torsten Braun

Institute of Computer Science, University of Bern, Switzerland
Email: {negar.emami, antonio.dimaio, torsten.braun}@unibe.ch

***Abstract*—This paper introduces Game-theoretic Trajectory Prediction through distributed reinForcement learning (GTP-Force), a system that tackles the challenge of predicting joint pedestrian trajectories in multi-agent scenarios. GTP-Force utilizes decentralized reinforcement learning agents to personalize neural networks for each competing player based on their non-cooperative preferences and social interactions with others. By identifying the Nash Equilibria, GTP-Force accurately predicts joint trajectories while minimizing overall system loss in non-cooperative environments. The system outperforms existing state-of-the-art trajectory predictors, achieving an average displacement error of 0.19m on the ETH+UCY dataset and 80% accuracy on the Orange dataset, which is -0.01m and 5% better than the best-performing baseline, respectively. Additionally, GTP-Force considerably reduces the model size of social mobility predictors compared to approaches with classical game theory.**

Keywords: Trajectory Prediction, Multi-Agent Social Interactions, Transformers, Reinforcement Learning, Neural Architecture Search, Non-cooperative Game Theory, Clustering.

I. Introduction

Forecasting mobile users' motion patterns, whether pedestrians or vehicles, has become increasingly important in urban planning and intelligent mobility systems. It can enable various technologies, such as intelligent transportation services, safety and emergency applications, rescue operations, autonomous vehicles, and road traffic engineering. Similarly, mobility prediction is a pivotal aspect of enabling various wireless network applications, including adaptive and anticipatory network management, resource allocation, handover management, and proactive service migration [5].

In crowded public spaces, mobile users follow certain social rules such as estimating other users' mobility status, respecting their space, or avoiding colliding with obstacles. This suggests a strong mutual influence among nearby users' mobility patterns and decisions. In this direction, socially-aware predictors have shown significant improvement over socially-unaware methods in multi-agent scenarios where users are not acting in isolation. Recently, the task of mobility prediction has shifted from individual models to joint models, where mutual influence among individuals in a complex dynamic environment helps to form group intelligence increasing the system's prediction performance.

While there have been advancements in the area of social-aware Trajectory Prediction (TP), existing predictors face significant limitations [1]. One major challenge is neural architecture inflexibility, which refers to the inability of existing models to adapt to new types of input data or incorporate new features. The majority of the existing works design the trajectory predictors' Neural Network (NN) architectures based on heuristics and experts' prior knowledge, which is a time-consuming and error-prone process.

Another limitation of existing social-aware trajectory predictors is their difficulty in handling non-cooperative social behaviors. These works mainly utilize centralized models, where the decision-making power is taken away from individuals and given to a centralized unit to take an optimal cooperative decision by aggregating everyone's data through a single neural architecture model. Therefore, all mobile users are considered to behave similarly, so that their motion can be predicted by the same model and with the same features [6]. However, real humans might rather optimize personal goals instead of joint strategies, making it challenging for mobility predictors to accurately forecast their future actions. In a social setting, a specific NN architecture may be well-suited to capturing the unique data characteristics of one user, while severely degrading the prediction performance of other users due to incompatibility with their data features.

On the other hand, privacy issues and communication network bottlenecks are other concerns regarding uploading large, private datasets to a centralized server. In this direction, the research community has introduced Federated Learning (FL) as a distributed Machine Learning (ML) approach to solve the centralized ML problems [7]. In a classic FL scenario, each federated participant trains a local model using its own dataset and only sends the model weights to a central server. The server then aggregates the weights from multiple clients by aggregating them to create a global model, which is then transmitted back to the local clients to be retrained for several communication rounds until the convergence. In this context, FL can be applied to a set of interacting users to create a decentralized social-aware TP model. This can help to protect the privacy of users while solving the scalability issues over the communication networks. However, the challenge with classical FL [7] is that aggregating through averaging locally trained models can limit the ability to personalize NN architectures, as local users must have identical NN architectures (in terms of number and sequence of layers and neurons) to enable matrix summation. Consequently, the NN architecture inflexibility issue arises in distributed TP as well.

To address the above-mentioned limitations, we propose

TABLE I
FEATURES OF EXISTING SOCIAL-AWARE TRAJECTORY PREDICTORS

Characteristics	Alahi et al. [1]	Bahram et al. [2]	Geiger et al. [3]	Ma et al. [4]	GTP-Force
Social-aware Model (Multi-Agent Setting)	✓	✓	✓	✓	✓
Intra- and Inter-Cluster Social Interactions	-	-	-	-	✓
Reinforcement Learning NN Design	-	-	-	-	✓
Decentralized Training Model	-	-	-	-	✓
Game-Theoretic Decision-making Polices	-	✓	✓	✓	✓
Non-Cooperative Strategic Game	-	-	✓	✓	✓
Simultaneous Game Modeling (Payoff Matrix)	-	-	✓	-	✓
Sequential Game Modeling (Game Tree)	-	✓	-	✓	-

Game-theoretic Trajectory Prediction through distributed reinForcement learning (GTP-Force), a distributed TP system that trains social Transformers (TFs) to predict joint trajectories of multiple competing users by personalizing the NN architecture of each user based on its unique interests while taking into account other users' strategies. GTP-Force clusters mobile users with similar trajectories resulting in inter-cluster users with distinct trajectories and competing mobility features. The inter-cluster users are then modeled as players in a non-cooperative game, with strategic choices made to ensure that no negative impact is placed on any individual user decision. To personalize the TF's architecture for each non-cooperative player, GTP-Force employs a Reinforcement Learning (RL) agent that is tailored to the specific mobility data features of the player, who actually represents the features of the cluster of users he comes from. This allows intra-cluster users with similar mobility features to be modeled using a shared NN, as in our other work [8], while inter-cluster users with different mobility features are treated as competitive players in a non-cooperative game. A set of pre-trained NN weights corresponding to highest-performance RL-designed TF architectures of each player is then transferred to the centralized server and a social pooling layer is added to form a social-aware trajectory predictor that captures the interactions of multiple interactive players. GTP-Force's centralized server trains different social-aware mobility predictors by combining players' different decisions (RL-designed TF architectures) and forms a payoff matrix for the non-cooperative game. The payoff matrix illustrates the possible rewards or penalties for various strategy combinations of interdependent players in a game. The payoff matrix is then returned to individuals, allowing them to choose an action that guarantees the Nash Equilibrium in the multi-agent setting.

Our contributions can be summarized as follows. 1. We propose a novel RL optimization method for developing high-accuracy and computationally-efficient TF NN trajectory predictors. 2. We propose an inter-cluster non-cooperative, but intra-cluster cooperative social game-theoretic trajectory predictor that effectively captures mobile users' interdependencies. 3. We validate our proposed trajectory predictor using two real-world datasets, capturing human mobility at both small and large scales. The rest of this paper is as follows. Section II presents the related work. Section III describes the GTP-Force operation. Section IV evaluates the GTP-Force performance. Finally, Section V concludes the contributions of this work.

II. RELATED WORK

In recent years, data-driven social-aware predictors are gaining popularity compared to the previously proposed *Social-force* models, which use simple repulsive and attraction forces [9]. The vast majority of modern human-trajectory predictors are based on deep learning models, such as Recurrent Neural Networks (RNNs), Long Short-Term Memorys (LSTMs), Convolutional Neural Networks (CNNs), and Attention NNs, which require less computation and achieve higher prediction accuracy compared to social-force models due to their better modeling of sequential patterns [5], [10], [11]. Instead of modeling kinetic forces and energy potentials as in social-force models, social-pooling [1], [12], attention [13], [14], and graph [15], [16] mechanisms complement NNs to share information about neighboring user's trajectories to capture complex interactions in crowded urban environments. Social-LSTM [1], Social-GAN [12], Sophie [14], Social-Ways [13], Social-STGCNN [16], and STAR [15] are various examples of social-aware TP models in the existing literature. Despite the popularity of above models, they face multiple limitations. The NN architectures used in these predictors are created manually by experts, which can be a time-consuming and error-prone process. Moreover, existing models assume cooperative behavior from all users in a multi-agent environment and employ a single NN architecture to train them. However, in reality, humans tend to optimize their personal goals, thus, applying a single inflexible NN model can lead to significant inaccuracies in joint and social TP.

According to the survey conducted by Rudenko et al. [6], classical AI and game-theoretic approaches hold great potential for modeling human behaviors in multi-agent settings. As a step in this direction, Ma et al. [4] and Geiger et al. [3] are among the few existing papers that partially address the problem of social-aware non-cooperative TP through Game Theory (GT) with some limitations. Ma et al. [4] propose a method for predicting the interactive dynamics of pedestrians using a combination of GT and deep learning-based visual analysis via Fictitious Play. In Fictitious Play [4], players play their best responses to their opponents. Each player updates their beliefs about the other players' strategies based on the observed outcomes and then selects their strategy for the next round. However, a limitation of this approach

is that Fictitious Play is designed for sequential games and may not be well-suited for modeling simultaneous games. In the context of decentralized TP problems, individuals may take actions simultaneously, necessitating the use of non-cooperative simultaneous games to more accurately model their impulsive behaviors. On the other hand, Geiger et al. [3] propose a game-theoretic framework for predicting the future trajectories of multiple agents in a social environment, using implicit layers to learn the best response of each agent in a Nash equilibrium of the game. The implicit layer is a NN layer that learns the underlying relationships between input and output data through a non-linear mapping function. However, this approach lacks personalized NN models for individual players to capture their non-necessarily-cooperative characteristics. Personalized NN models can capture the unique features and preferences of each player, which ultimately leads to more accurate joint predictions. In contrast, the use of a single implicit layer to perform non-cooperative TPs may not be sufficient to capture the full complexity of the social environment and the individuals within it.

To address the challenge of designing high-performance personalized neural architectures, some approaches employ Auto-ML hyperparameter optimization methods, which often rely on random search models [17]. However, Neural Architecture Search (NAS) is an NP-hard problem that conventional optimization methods, such as random search, cannot solve it in polynomial time as the search space expands. Some other works use Bayesian optimization methods to tune hyperparameters in ML models [18]. Bayesian optimization leverages probabilistic models to search for optimal hyperparameters, reducing the need for exhaustive evaluations. However, constructing a probabilistic model of the hyperparameters and iteratively updating it with new observations is computationally expensive, specifically when dealing with large search spaces.

We propose GTP-Force, which leverages RL to efficiently design personalized NNs for players participating in a non-cooperative simultaneous game. The objective of GTP-Force is to search for the optimal combination of NNs (optimal strategy profile) among inter-cluster users who participate in a game-theoretic social TP. The advantage of using RL-based optimization is its accumulative nature, enabling to partially train different NN architectures for a limited number of epochs in each episode. By leveraging RL's sequential and cumulative decision-making framework, we can narrow down a vast search space to find a high-performance neural architecture, while reducing the computational costs. Table I compares the characteristics of our solution GTP-Force with those of existing state-of-the-art social-aware trajectory predictors.

III. GTP-Force

A. System Model

We define a scenario in which n users move within an urban area containing S base stations providing Internet access via a cellular radio network. Each user has a wireless device that connects to the base station with the strongest signal. As users move, the received signal power from the base stations changes, which requires a *handover* to a new base station. The timestamps at which each user connects and disconnects from each base station are recorded. We assume that at any timestamp v_u, user u is located at coordinates $(x_u, y_u) \in \mathbb{R}^2$ and may be connected to a base station with ID $b_u \in \mathbb{N}$. The vector $p_u = (v_u, x_u, y_u, b_u) \in \mathbb{R}^3 \times \mathbb{N}$ represents a single data point about the user's status and is referred to as the *user information vector*. Each user u has a total of m_u user information vectors recorded. The set of these vectors $T_u = \{p_u(1), \ldots, p_u(m_u)\}$, is referred to as the user's trajectory, and the set of all user trajectories is denoted as $\Theta = \{T_1, \ldots, T_n\}$.

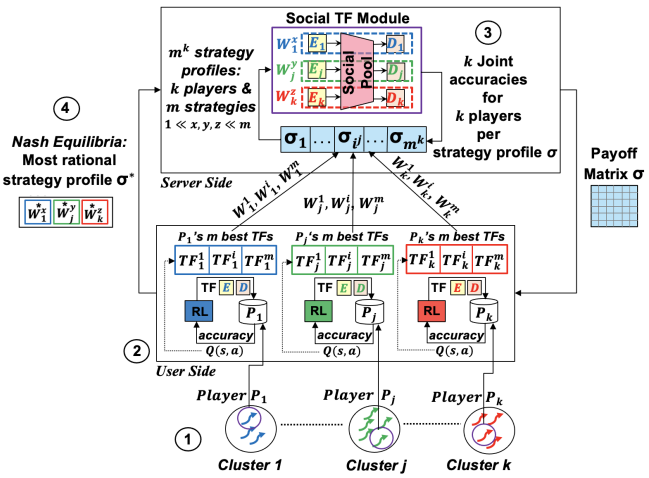

Fig. 1. GTP-Force Architecture.

B. Problem Formulation and GTP-Force Architecture

The GTP-Force workflow is divided into four phases: (1) the *Game Player Selection*, (2) the *Distributed RL-TF Training*, (3) the *Social Interaction Payoff Computing*, (4) and the *Non-Cooperative TP*, which are illustrated in Figure 1 and elaborated hereafter. Figure 1 shows a social trajectory predictor that trains on data from multiple non-cooperative players and jointly predicts multiple trajectories that are influenced by each other's mobility. Each player is selected from a distinct cluster that has unique mobility data features. On the user side, it can be observed that an RL agent is used to design the TF's Encoder and Decoder architecture *TF(E,D)* for each of the contesting inter-cluster players. On the server side, multiple *TF(E,D)s* of different individuals are then merged through a *social pool* unit to form a social-aware trajectory predictor and compute the corresponding payoff matrix values.

1) Game Player Selection: We assume that the n users in the system can be classified into k clusters of users with similar trajectories, where $k << n$. Users are partitioned into distinct clusters $C = c_i \subseteq \Theta | i = 1, \ldots, q$, where $c_i \cap c_j = \emptyset$ for all $i, j \in 1, \ldots, q$, $i \neq j$. Users within a cluster have comparable mobility features and can be modeled using a single TF NN model. In contrast, inter-cluster users have distinct mobility features and are treated as competitors, each with their own preferences and desirable strategies. To model the non-cooperative game among inter-cluster users, we select

one player from each cluster. The selection of a single user per cluster offers several advantages, including reducing computational and communication overheads in the wireless network and facilitating the identification of the Nash equilibrium. For selecting a *player* from each cluster, we prioritize those with the most reliable and periodic data quality. This guarantees that the users that are chosen to play in the game can design the best NN for the entire cluster. Since GTP-Force is a distributed system and the central server lacks access to local users' raw data, we propose a *quality-estimator metric* called *regularity ratio*. This metric allows local users to estimate the quality and periodicity of their own data and only transmit regularity ratio values to the centralized server through the wireless channel. This let the server select a player from each cluster with the most reliable estimated data.

We process the user's mobility data signal in both time and frequency domains to define *time-domain regularity ratio* and *frequency-domain regularity ratio*. On one hand, our extensive experiments have led us to infer that users whose data has more samples while visiting fewer locations tend to have more regular mobility patterns, resulting in improved TP accuracy. Therefore, we define the *time-domain regularity ratio* of each user data as the ratio between the total number of data samples and the number of unique visited locations. Users with a higher regularity ratio make it easier for NNs to identify periodic behavior compared to users with a lower regularity ratio. On the other hand, our observations indicate that converting the time-series mobility data signal to the frequency domain can also provide valuable insights. Specifically, a signal with a high power spectral density with a dominant frequency that is sufficiently high, coupled with a high Signal-to-Noise Ratio (SNR), suggests the presence of a strong and well-defined oscillation at a high frequency, with relatively low levels of noise. This results in user data that is easier to analyze and predict, as the oscillations are distinct and identifiable. Thus, we define the *frequency-domain regularity ratio* as the ratio between the SNR and the dominant frequency of the user's time-series data. The acceptable threshold of regularity ratios in both domains is set empirically based on the specific dataset being used. By combining information from both domains, we can achieve a decent estimation of the quality and periodicity of the local user data. The central server selects a user as the player for the game only if their regularity ratios meet the threshold criteria in both time and frequency domains.

2) Distributed RL-TF Training: During this phase, each of the chosen players trains an RL agent on its local data to identify the most fitting neural architectures for its own cluster's data features. Each RL agent uses the ε-greedy Q-learning policy to list m highest-performance TF architectures' trained weights. This list of pre-trained encoder-decoder weights will be transferred to the server presenting each player's m choice of NN strategies that will be played through a non-cooperative game. In a classic RL, an *agent* takes an *action* based on the *environment's* rules that affects the environment's *state* and receives a *reward* corresponding to the taken action. The sequence of aforementioned steps is called an *episode*. In RL-based NAS, the state signifies the present architecture or configuration of the NN during a specific moment of the training process. In each episode, as the RL agent adds a layer to the current architecture as its action, the environment evolves into a new state. State transitions in RL capture the essence of Markov Decision Processes, a powerful framework for solving sequential decision-making problems. The main goal of the RL agent is to find an action whose accumulated reward is maximized over a series of episodes. The RL agent can take actions from a large, finite search space of admissible NN architectures called *action space*. In GTP-Force, the action space expands to a set of diverse TF architectures made by all possible combinations of TF hyperparameter values.

TFs are composed of encoder and decoder stacks and characterized by a large set of hyperparameters named as the number, characteristics, and sequence of multi-head attention layers, add and norm layers, feed-forward layers, and dropout layers. The multi-head self attention module, as the most important element of a TF, contains a self-attention mechanism that accesses previous segments of input data and can differently weigh the importance of each segment based on segments' pairwise similarities. The self-attention feature enables parallel training for TFs, which considerably reduces training time compared to sequential RNNs. Each of the multi-head self attention layers could possess different numbers of heads and values for the dimension of attention keys. The term *add* in the add and norm layer refers to the TF's residual connection that adds the output of a previous layer to the input of a subsequent layer in order to prevent gradients from vanishing or exploding during training deep NNs. Add and norm layers could have different normalization values. Feed-forward layers could have different numbers of hidden neurons and activation functions. Dropout layers could have different dropout ratios. Therefore, the action space can become remarkably large relative to the input range of the hyperparameters (see the RL Agent Actions section of Table II).

At each episode, after the agent proposes a new NN architecture, by adding a layer to the current NN as its action, the unknown reward associated with the selected architecture is evaluated by training the suggested TF on the player's data for a few epochs. We assume that on each RL episode t, the proposed architecture achieves a loss $L_t \in \mathbb{R}$, which is defined as Sparse Classification Cross-Entropy for classification and Mean Square Error for regression problems. The RL agent uses a reward function $r_t = -L_t$, to evaluate the performance of the candidate NN and updates the Q-table to record state-action transitions using the Bellman equation (Equation 1), where $\gamma \in [0, 1]$ is the *discount factor* that regulates the relevance of recent rewards, s' is the resulting state from the action selected by the agent, and $\mathcal{R}(s')$ is the action space from state s'. We define the cumulative reward r_t^* as $r_t^* = \sum_{k=0}^{T'} \gamma^k r_{t+k}$, where T' is the number of episodes used to compute the cumulative reward. As the number of episodes increases, the Q-table's content corresponding to the optimal policy π tends to $Q^\pi(s_t, a_t) = \mathbb{E}[r_t^* | s_t, a_t]$. At the end of the

exploration stage, the RL agent enters the *exploitation stage* where it selects the m best NN architectures by searching the m states s^* corresponding to m most expected cumulative reward values from the Q-table, as shown in Equation 2.

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha\left(r_t + \gamma \max_{a' \in \mathcal{R}(s')} Q_t(s',a')\right) \tag{1}$$

$$s^* = \left(\underset{(s,a)\in\mathcal{S}\times\mathcal{R}(s)}{\arg\max}\ Q^{\pi}(s,a)\right)_1 \tag{2}$$

3) Social Interaction Payoff Computing: At this point, each player's trained RL agent sends a list of its m best NNs' trained weights as its m possible decisions or strategies for the game to the central server through the available network throughput. As shown in Figure 1, the central server contains a social-pool module where multiple users' transformers' encoder stacks E_i and decoder stacks D_i can be aggregated forming a social-aware model. This module takes multiple inputs from multiple users, aggregates them and captures their interdependencies, and outputs multiple joint predictions. Each player's architecture for E_i and D_i can be selected from their list of m top-performing RL-designed transformers, representing the player's *strategy* through the *game*. The joint predicted trajectories show each player's output considering other players' strategies, thereby forming the *payoff matrix*.

GTP-Force models the social TP through a simultaneous non-cooperative game, where it aims to find the best combination of NNs for different conflicting players so that the interest of each player is optimized while taking into account other users' strategies. In a classic non-cooperative simultaneous game, multiple players make decisions simultaneously, without knowing the decisions made by the other players. Let $K = \{p_1, \ldots, p_k\}$ be the set of players, where each player $p_j \in K$ can select a strategy $\sigma_j \in \Sigma$ among $m = |\Sigma|$ possible strategies. The payoff for each player depends on the combination of strategies chosen by all players. The game's outcomes for each player can be represented by a payoff matrix $A \in \mathbb{R}^{m^k \times k}$ with entries $a_{\sigma_1,\ldots,\sigma_k,j}$, where $\sigma_1, \ldots, \sigma_k \in \Sigma$. Each row of matrix A represents one of the m^k possible combinations of strategies for all players. The entry $a_{1,\ldots,k,j}$ represents the payoff for player j when the players choose the strategies $\sigma_1, \ldots, \sigma_k$. The matrix A can be constructed as follows:

$$A = \begin{bmatrix} a_{1,\ldots,1,1} & a_{1,\ldots,1,2} & \cdots & a_{1,\ldots,1,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,\ldots,m,1} & a_{m,\ldots,m,2} & \cdots & a_{m,\ldots,m,m} \end{bmatrix}. \tag{3}$$

4) Non-Cooperative TP: During this phase, the central server sends back the completed payoff matrix to all players through the wireless link. Each element of the matrix contains k joint prediction accuracies for k players, revealing the impact of each player's strategy on others in terms of social TP. In the analysis of the non-cooperative simultaneous game, the concept of Nash equilibrium is of great importance for determining the best strategy profile (in our case, determining the best combination of NN architectures) for competing players. The Nash equilibrium signifies a set of strategies (a strategy profile) in which no player is motivated to alter its strategy, provided they are aware of the impacts of the strategies of the other players. Nash equilibrium is not necessarily the combination that yields the maximum total or expected payoff, but rather a set of strategies that is rational given the strategies chosen by other players. As players act independently and may not know other players' deterministic strategies, it may not always be possible to select the optimal solution.

A strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_k^*)$ is a Nash equilibrium if, for each player $j \in K$, their strategy σ_j^* is the best response b_j to the strategies of the other players, i.e., $\forall \sigma \in \Sigma^k : u_j(\sigma^*) \geq u_j(\sigma)$. In this context, u_j denotes the utility function of player j, which represents the player's preference over the possible outcomes. In other words, player j's strategy σ_j^* is a best response to the strategies $(\sigma_1^*\sigma_2^* \ldots, \sigma_{j-1}^*\sigma_{j+1}^* \ldots, \sigma_k^*)$ chosen by the other players. To find the best response of player j in a non-cooperative game, we use the following formula:

$$b_j(\sigma_{-j}) = \arg\max_{\sigma_j \in \Sigma} u_j(\sigma_j, \sigma_{-j}), \tag{4}$$

where σ_{-j} denotes the strategies of all other players in the game. To determine whether a given strategy profile is a Nash equilibrium, we can find the best response of each player to the strategies of the other players, using the formula for the best response given above, substituting σ_{-j} with $(\sigma_1^*, \sigma_2^*, \ldots, \sigma_{j-1}^*, \sigma_{j+1}^*, \ldots, \sigma_k^*)$.

Given that players are rational and using the Nash equilibrium to personalize the non-cooperative NNs, GTP-Force evaluates the performance of the social-aware TP by selecting the best combination of NAS decisions in a multi-agent scenario. After the non-cooperative training, GTP-Force distributes the pre-trained models to all users of each cluster who were not played in the game from their respective clusters' players. This approach ensures that every user in the system can perform prediction tasks effectively, without compromising the accuracy or computational complexity of the model.

Algorithm 1 details the GTP-Force system's workflow. The first section of the algorithm (lines 1 to 5) describes how mobile users from contesting clusters are chosen to be the players of the non-cooperative game. The second section (lines 6 to 22) describes how players train locally the RL-designed transformers tailored to their unique characteristics. The third section of the algorithm (lines 23 to 25) describes how social interaction among players is extracted for different combinations of their strategies to form the payoff matrix. Finally, the fourth section of the algorithm (lines 26 to 27) explains how the non-cooperative game can be efficiently played by satisfying the Nash Equilibrium. Once the optimal strategy profile has been identified, the joint trajectory prediction can be performed. The algorithm includes comments indicating where computations are performed at the *central server* and

Algorithm 1: GTP-Force Workflow

Input: Set of trajectories Θ and clusters C
Output: Optimal social-TF multi-RL architecture F_k^* through Nash Equilibrium

// Compute locally regularity ratio of each user u_i with trajectory T_i and transmit it to the server
foreach $T_i \in \Theta$ **do**
  Compute Regularity Ratio r_i;
// Build a social transformer F_k, $\forall c_k \in C$
foreach $c_k \in C$ // 1. Game Player Selection
do
  // Elect the user with max regularity ratio as the player by the server and let the player knows
  $p_k \leftarrow \arg\max_{u_k \in c_k} r(u_k)$;
  // 2.Distributed RL-TF Training
  // Locally initialize RL agent A_k to optimize the TF architecture using data of player p_k
  $A_k \leftarrow \texttt{InitAgentRL}(\gamma, \alpha, \varepsilon, \varepsilon_0)$
  // Initialize state-action table to zero for all states and actions
  $\forall (s,a) \in S \times A : Q(s,a) \leftarrow 0$;
  // Initialize exploration probability ε to maximum and empty architecture state
  $\varepsilon \leftarrow 1$, $s \leftarrow \emptyset$;
  // Optimize TF architecture up to $v_{\max}$ episodes
  foreach $v \in \{1, \ldots, v_{max}\}$ **do**
    // Decrease exploration every $v_{\max}\varepsilon_0$ episodes
    if $v \mod v_{max}\varepsilon_0 = 0$ **then**
      $\varepsilon \leftarrow \varepsilon - \varepsilon_0$;
    // ε-greedy to select next TF architecture
    if $\texttt{RandomSample}([0,1]) \leq \varepsilon$ **then**
      $a_v \leftarrow$ random action $a \in A(s)$;
    else
      $a_v \leftarrow \arg\max_{a \in A(s)} Q(s,a)$;
    // Update state according to action a_v
    $s' \leftarrow \texttt{UpdateState}(s, a_v)$;
    // Train the TF with data of the representative user p_k for a few epochs θ_s and compute model error w and reward ρ_v
    $s'_* \leftarrow \texttt{Train}(s', r_k, \theta_s)$;
    $w \leftarrow \texttt{ComputeModelError}(s'_*, r_k)$;
    $\rho_v \leftarrow 1/w$;
    // Update Q-learning table by Bellman equation
    $Q(s, a_v) \leftarrow (1-\alpha) Q(s, a_v) + \alpha\left(\rho_v + \gamma \max_{a \in A(s')} Q(s', a)\right)$;
    $s \leftarrow s'$;
  // Select m trained NN weights with lowest loss
  $W_k \leftarrow \arg\max_{W' \subseteq \{w_1, \ldots, w_m\}} Q(s, \cdot)$;
// Transmit locally-trained W_k to the server
// 3. Social Interaction Payoff Computing
foreach $\sigma \in \{1, \ldots, m^k\}$ **do**
  // Train the Social-TF for k players with k Encoders connected to k Decoders through one Social-Pool by the server
  $F_k \leftarrow \texttt{Train}(f_k, c_k, \theta_s)$;
// Form the Payoff Matrix A from strategy σ_k and prediction accuracy F_k and send A to local players
$A \leftarrow (\sigma_k, F_k), \forall k \in K$;
// 4. Non-Cooperative Trajectory Prediction
// Compute locally *Nash Equilibria* through finding the best response of player $j \in K$
$b_j(\sigma_{-j}) \leftarrow \arg\max_{\sigma_j \in \Sigma} u_j(\sigma_j, \sigma_{-j})$;
$F_k^* \leftarrow \sigma^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_k^*)$;

where they are carried out *locally* by decentralized players.

IV. Evaluation

A. Experimental Setup

In our evaluation, we compare the performance of GTP-Force against several mobility predictors in *small-scale* and *large-scale* mobility scenarios. We leverage the TF NN architecture, implement the RL process, and simulate GT along with other state-of-the-art TP models using Keras, TensorFlow, TensorFlow GPU, and Nashpy open-source libraries. The constant parameters used to train the Social TP model, TF predictor, and RL agent used through GTP-Force are shown in the first two sections of Table II. Additionally, the third section of Table II displays the search space of hyperparameters that form the transformer architecture, with each row corresponding to one of the RL's potential actions. After clustering similar trajectory users and reducing the number of players in a non-cooperative game, we are able to solve the Nash equilibrium using a brute force approach. However, it should be noted that finding a Nash equilibrium is a computationally challenging task and the brute force approach may not be practical for large games involving many players. In such scenarios, we consider using more advanced algorithms or heuristics, such as the Lemke-Howson algorithm, which can offer more efficient solutions for finding Nash equilibrium.

1) Small-scale Scenario: In the small-scale scenario, we consider a limited moving area of users covering only a few tens of meters. In this scenario, base station information is irrelevant and has no impact on the TP task. Instead, the mobility is characterized by the sequence of location coordinates. The ETH and UCY public datasets are used to gather information on small-scale pedestrian mobility in urban settings. A total of 1536 pedestrian trajectories are extracted from the datasets with a sampling rate of $0.4\,\mathrm{s}$ using the method described in [11]. Unique user data is saved separately to simulate a distributed-dataset scenario. The ETH dataset contains data from two urban locations (ETH and Hotel), while the UCY dataset includes data from three urban locations (Univ, Zara1, and Zara2). Each trajectory T_u in this scenario consists of a sequence of up to 100 user information vectors $p_u = (v_u, x_u, y_u)$, where the timestamp granularity of any two consecutive vectors is $0.4\,\mathrm{s}$. The prediction models are trained to observe a user's trajectory over the past $T_{\text{obs}} = 8$ timestamps ($3.2\,\mathrm{s}$) and predict the next $T_{\text{pred}} = 12$ timestamps ($4.8\,\mathrm{s}$) to allow for a fair comparison with state-of-the-art works using this popular dataset. In this scenario the goal of the mobility predictor is to forecast next location coordinates, and thus, TP is modeled as a *regression* problem.

To assess the performance of GTP-Force in predicting future location coordinates of users in the *small-scale* scenario, we compared it with several social-aware trajectory predictors including: Social-LSTM [1], Social-GAN [12], Sophie [14], Social-Ways [13], Social-STGCNN [16], STAR [15], INTRAFORCE [10], and FedForce [8], based on their *Average Displacement Error (ADE)*. The *displacement error* in predicting the trajectory of user u over a prediction window

of T_{pred}, given the observation window of T_{obs}, is defined as the average squared Euclidean distance between the predicted points of the trajectory and the true locations, expressed as:

$$E(u) = \frac{1}{T_{\text{pred}}} \sum_{t=T_{\text{obs}}+1}^{T_{\text{obs}}+T_{\text{pred}}+1} (\hat{x}_u^t - x_u^t)^2 + (\hat{y}_u^t - y_u^t)^2. \quad (5)$$

To evaluate the performance of the prediction models, we use the ADE calculated as the average of the $E(u)$ values over n users as $\frac{1}{n}\sum_{u=1}^{n} E(u)$.

2) Large-scale Scenario: In the *large-scale* scenario, the users are assumed to be moving over a large area that covers several kilometers, and their specific location within a small area over a short time window is not relevant. Instead, their mobility is characterized by the sequence of base stations to which they connect over time. The dataset used in this scenario was provided by Orange S.A., France, containing management data of a private cellular network, which includes timestamps and base station IDs to which each of the 1.3×10^6 users connect while moving over a Paris district between July and September 2019 [5]. User identities have been anonymized for privacy reasons, and the location coordinates of the 131 identified base stations are not available. In this scenario, user trajectories consist of sequences of a few thousand information vectors, where the timestamps of any two consecutive vectors are a few minutes apart. To simulate a distributed-dataset scenario, we generate a local dataset for each user by splitting the Orange dataset into several sub-datasets, grouping samples by unique user IDs, and storing these local datasets on each user. We assume that the predictor processes a user's past trajectory for the previous $T_{\text{obs}} = 16$ timestamps and predicts for the future $T_{\text{pred}} = 1$ timestamp. In this scenario the goal of the mobility predictor is to forecast next base station IDs, and thus, TP is modeled as a *classification* problem.

For evaluating the GTP-Force performance in predicting future base station IDs in the *large-scale* scenario, we conduct two experiments. In the first experiment, we compare the prediction *accuracy* and *build time* of Reinforced Transformer (RL-TF) with several other trajectory predictors, including NN-based predictors RL-CNN, RL-LSTM, HO-LSTM, and GS-LSTM, as well as non-NN predictors J48 Decision Tree, and XGBoost and RF ensemble models. RL-LSTM uses RL to optimize LSTM architecture, while HO-LSTM and GS-LSTM use Hyperopt (Auto-ML) and Grid Search, respectively. The aim of this experiment is to demonstrate that TFs and RL outperform other ML predictors and hyperparameter optimization models. In the second experiment, we evaluate the performance of the GTP-Force, applying GT to competing RL-TFs, as a social-aware TP model in terms of *accuracy* and *model size* performance metrics, in comparison to RL-TF without GT, GT-based TF without RL personalization (GT-TF), and the classical social-aware model using TFs without GT and RL (Social-TF). *Accuracy* is defined as the ratio of correctly predicted next locations to the total number of predictions made by the model. *Build time* and *train time* correspond to the time to build and train the personalized model by RL agent and the average time to train multiple users individually or in the social GTP-Force framework, respectively. We define *model size* as the total number of a NN's training parameters.

TABLE II
PARAMETERS FOR SMALL-SCALE AND LARGE-SCALE SCENARIOS

Transformer Parameters	
Batch size (small-scale, large-scale)	10, 200
Learning rate decay	0.002
Social Transformer training epochs θ_{l}	200
Early stopping patience (in epochs)	10
Early stopping improvement delta threshold (small-scale, large-scale)	0.05, 0.1
Dense layers' activation func. (hidden, output)	ReLU, SoftMax
Reinforcement Learning Parameters	
Maximum RL training episodes υ_{max}	500
Training epochs per episode θ_{s}	20
Discount factor γ, learning rate α	1, 0.01
Exploration rate decay ε_0	0.1
Training target per episode (small-scale) η	0.05
Training target per episode (large-scale) η	0.1
Exploration training validation (small-scale)	4 sets train, 1 set test
Exploration training validation (large-scale)	10-fold x-validation, 70% train, 30% test
RL Agent Actions: Transformer Hyperparameters Space	
Number of hidden layers	$10, 11, \ldots, 50$
Number ξ of encoder and decoder layers	$1, 2, 3, 4, 5$
Number of heads h in a multi-head attention layer	$2, 4, 6, 8$
Dimension of the key for a self-attention layer	$64, 128, 265$
Normalization layer parameter	$10^{-2}, 10^{-3}, 10^{-6}$
Number of perceptrons in dense layer	20, 50, 80, 100, 150
Dropout ratio in dropout layer	0.15, 0.25, 0.5, 0.75

TABLE III
AVERAGE DISPLACEMENT ERROR (ADE) [M] OF DIFFERENT SOCIAL TRAJECTORY PREDICTORS FOR THE SMALL-SCALE SCENARIO (ETH+UCY DATASETS)

Work	ETH	Hotel	Univ	Zara1	Zara2	**Mean**
Social-LSTM [1]	1.09	0.79	0.67	0.47	0.56	**0.72**
Social-GAN [12]	0.81	0.72	0.60	0.34	0.42	**0.58**
SoPhie [14]	0.70	0.76	0.54	0.30	0.38	**0.54**
Social-BiGAT [19]	0.69	0.49	0.55	0.30	0.36	**0.48**
Social-Ways [13]	0.39	0.39	0.55	0.44	0.51	**0.46**
Social-STGCNN [16]	0.64	0.49	0.44	0.34	0.30	**0.44**
PECNet [20]	0.54	0.18	0.35	0.22	0.17	**0.29**
STAR [15]	0.36	0.17	0.31	0.26	0.22	**0.26**
INTRAFORCE [10]	0.31	0.24	0.22	0.14	0.23	**0.22**
FedForce [8]	0.28	0.21	0.22	0.14	0.19	**0.20**
GTP-Force	**0.27**	**0.17**	**0.21**	**0.16**	**0.18**	**0.19**

B. Results

1) Small-scale Scenario: Table III shows the results of the small-scale experiment (ETH+UCY datasets), in which GTP-Force achieves the lowest ADE, 0.19m, compared to several state-of-the-art social trajectory predictors. Our other works, INTRAFORCE [10] and FedForce [8], focus on centralized and decentralized models respectively for intra-cluster cooperative-user social interaction extraction. We can observe

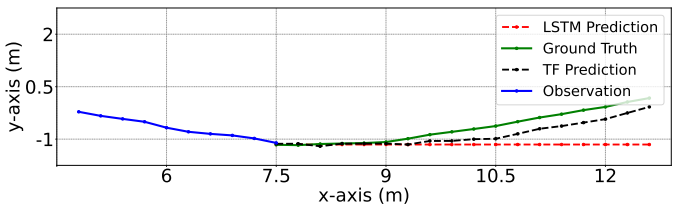

Fig. 2. TF- and LSTM-predicted trajectories versus the ground-truth path for the ETH+UCY datasets.

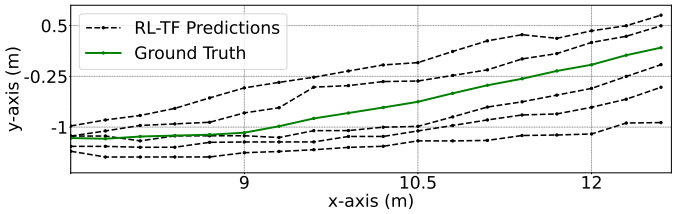

Fig. 3. Various RL-designed TF predictions for the ETH+UCY datasets.

that GTP-Force model outperforms both INTRAFORCE and FedForce due to its capacity of modeling adversaries.

Figure 2 displays predicted trajectories by both TF and LSTM, as compared to the ground-truth trajectory. We observe that the optimal RL-designed TF path more closely resembles the ground-truth than the optimal RL-designed LSTM path.

Figure 3 depicts various RL-designed TFs' predicted trajectories as compared to the ground-truth path. This figure's goal is mainly to show the impact of different combinations of TF hyperparameters, forming a specific neural configuration, on the predication performance.

2) Large-scale Scenario: The results obtained from the large-scale experiment (Orange dataset) reveal that utilizing RL to explore the NN architecture of the TF, and GT to model the decision-making of non-cooperative multi-agent NN designs leads to superior prediction accuracy, expedited model building time, and reduced training parameter size compared to other trajectory prediction methods.

In Figure 4a, we compare the Kernel Density Estimation (KDE) of accuracy between RL-TF and other predictors. KDE of accuracy refers to the estimation of the distribution of prediction accuracy values across the trained and tested 100 random users. It can be observed that the distribution of user prediction accuracy for RL-TF is biased towards higher values, indicating that it is better at predicting user trajectories with greater precision. Individual RL-TF achieves mean accuracy of (75%), which is 10% better than the other reinforced models (RL-LSTM and RL-CNN) and almost 20% higher than non-NN models (XGBoost, RF, and J48). Achieving a higher prediction accuracy over the Orange dataset is restricted by the limited dataset size (63 days) and the significant diversity in users' data sample distributions. The obtained accuracy of 75% is the average accuracy over 100 random users with highly variable data quality and periodicity.

In Figure 4b, it is evident that RL-TF demands slightly more than one hour of build time, which is comparable to RL-CNN and RF. Despite having larger architectures than CNNs and LSTMs, TFs' attention mechanism notably diminishes the

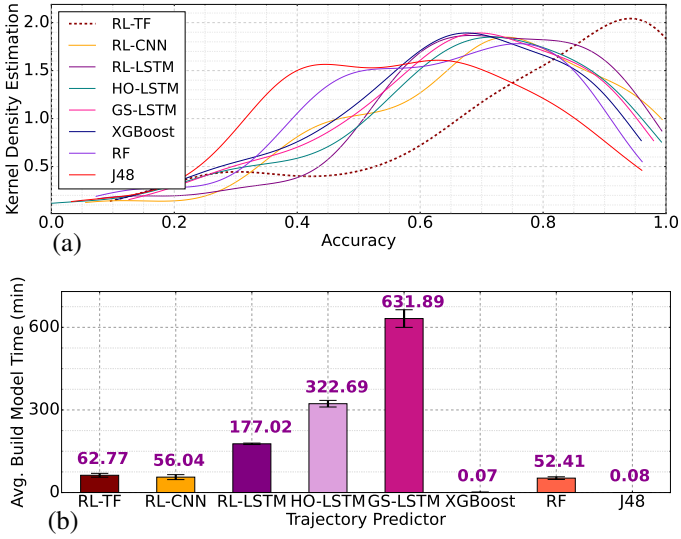

Fig. 4. Accuracy KDE (a) and build time (b) of different trajectory predictors for the large-scale Orange dataset.

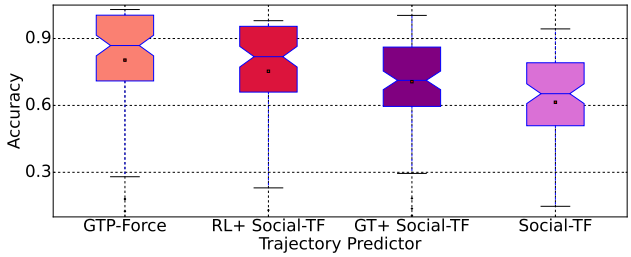

Fig. 5. Accuracy of GTP-Force with respect to other social trajectory predictors for the large-scale Orange dataset.

build time. Furthermore, the RL-TF build time is comparable to that of non-neural RF, which does not necessitate architecture exploration (no NAS) but only training. RL-LSTM and HO-LSTM involve extended build times due to the sequential nature of LSTMs and the extensive training of Hyperopt's random search-based exploration.

To have fair evaluations, we compare the average accuracy of GTP-Force, which employs non-cooperative GT in social TP via RL-designed TFs, with respect to RL-designed Social-TF without GT, Game-theoretic Social-TF without RL personalization, and Social-TF without RL personalization or non-cooperative GT modeling, as shown in Figure 5. It can be observed that GTP-Force achieves an average accuracy of 80%. This represents a 5% improvement over the RL-based TF predictor, a 10% improvement over the classical GT-based TF predictor without RL personalization, and a 15% improvement over the simple Social-TF without RL or GT.

Figure 6 compares the average model size of GTP-Force, an RL- and cluster-based game theoretic approach, with that of the classical GT approach for social-aware TF trajectory prediction across various numbers of users. We observe that the model size difference gap between GTP-Force and the classical GT approach increase as the number of users, and subsequently the number of clusters, rises. The reduction in model size of GTP-Force is attributed to the utilization of optimized RL technique and the clustering of similar-trajectory

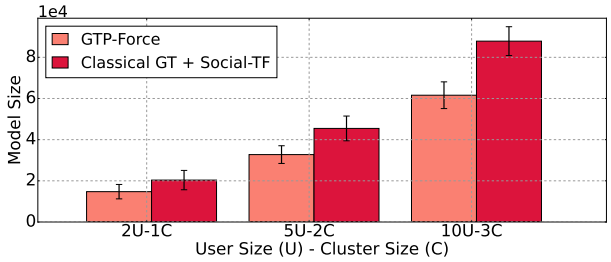

Fig. 6. Model size of the GTP-Force with respect to the classical GT for the social-TF trajectory prediction on Orange dataset.

users, enabling the training of a single player per cluster. This smaller model size improves computational efficiency and scalability of the system in large wireless networks.

In Figure 6, we conducted a comparison between GTP-Force and the classical game-theoretic TP model. However, we did not evaluate the computational complexity of GTP-Force in comparison to non-game theoretic predictors within this analysis. The reason is that, as shown in Figure 5, GTP-Force exhibited significantly higher prediction accuracy compared to non-game theoretic predictors by a factor of 15%. Due to the significant performance improvement by GTP-Force, we prioritize accuracy gains over potential increases in the algorithm complexity resulting from the utilization of GT.

V. Conclusions

We present GTP-Force, a multi-agent trajectory predictor that takes into account the social interactions among adversarial mobile users. The system clusters similar-trajectory users and leverages Reinforcement Learning (RL) to design transformer neural network architectures based on the collaborative *intra-cluster* user mobility features. It utilizes non-cooperative Game Theory (GT) among *inter-cluster* users who possess diverging mobility features. Evaluations on the small-scale ETH+UCY and large-scale Orange mobility datasets show that GTP-Force outperforms existing baselines by improving ADE by -0.01m, achieving 5% higher accuracy, and a 70% reduction in training time, respectively. GTP-Force achieves 80% average accuracy, which is 5% higher than RL-designed Social-TF, 10% higher than game theoretic Social-TF, and 15% higher than the simple Social-TF (without RL and GT). As the number of users within the social system increases, the model size gap between GTP-Force and the classical GT approach widens, with GTP-Force consistently achieving a smaller model size due to its clustering mechanism and the RL-based neural architecture search technique.

Acknowledgments

This work was funded by the SNF Intelligent Mobility Services project (No. 184690). We thank Orange S.A., France, for providing the dataset used for the experiments.

References

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

[2] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr, "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3981–3992, 2015.

[3] P. Geiger and C.-N. Straehle, "Learning game-theoretic models of multiagent trajectories using implicit layers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 4950–4958, 2021.

[4] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani, "Forecasting interactive dynamics of pedestrians with fictitious play," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 774–782, 2017.

[5] Z. Zhao, N. Emami, H. Santos, L. Pacheco, M. Karimzadeh, T. Braun, A. Braud, B. Radier, and P. Tamagnan, "Reinforced-lstm trajectory prediction-driven dynamic service migration: A case study," *IEEE Transactions on Network Science and Engineering*, 2022.

[6] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[8] N. Emami, A. Di Maio, and T. Braun, "Fedforce: Network-adaptive federated learning for reinforced mobility prediction," in *48th International Conference on Local Computer Networks (LCN)*, pp. to appear–, IEEE, 2023.

[9] D. Helbing, L. Buzna, A. Johansson, and T. Werner, "Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions," *Transportation science*, vol. 39, no. 1, pp. 1–24, 2005.

[10] N. Emami, A. Di Maio, and T. Braun, "Intraforce: Intra-cluster reinforced social transformer for trajectory prediction," in *18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 333–338, IEEE, 2022.

[11] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 10335–10342, IEEE, 2021.

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

[13] J. Amirian, J.-B. Hayet, and J. Pettré, "Social ways: Learning multi-modal distributions of pedestrian trajectories with gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[14] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. 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*, pp. 1349–1358, 2019.

[15] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-temporal graph transformer networks for pedestrian trajectory prediction," in *European Conference on Computer Vision*, pp. 507–523, Springer, 2020.

[16] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14424–14432, 2020.

[17] C. Wang, X. Chen, J. Wang, and H. Wang, "Atpfl: Automatic trajectory prediction model design under federated learning framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6563–6572, 2022.

[18] M. Kusner, J. Gardner, R. Garnett, and K. Weinberger, "Differentially private bayesian optimization," in *International conference on machine learning*, pp. 918–927, PMLR, 2015.

[19] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *European Conference on Computer Vision*, pp. 759–776, Springer, 2020.