ATPFL: Automatic Trajectory Prediction Model Design under Federated Learning Framework

Chunnan Wang[1], Xiang Chen[1], Junzhe Wang[1], Hongzhi Wang[1,2*]
[1]Harbin Institute of Technology
[2]Peng Cheng Laboratory
WangChunnan@hit.edu.cn, {20s003052,1190201815}@stu.hit.edu.cn, wangzh@hit.edu.cn

Abstract

Although the Trajectory Prediction (TP) model has achieved great success in computer vision and robotics fields, its architecture and training scheme design rely on heavy manual work and domain knowledge, which is not friendly to common users. Besides, the existing works ignore Federated Learning (FL) scenarios, failing to make full use of distributed multi-source datasets with rich actual scenes to learn more a powerful TP model. In this paper, we make up for the above defects and propose ATPFL to help users federate multi-source trajectory datasets to automatically design and train a powerful TP model. In ATPFL, we build an effective TP search space by analyzing and summarizing the existing works. Then, based on the characters of this search space, we design a relation-sequence-aware search strategy, realizing the automatic design of the TP model. Finally, we find appropriate federated training methods to respectively support the TP model search and final model training under the FL framework, ensuring both the search efficiency and the final model performance. Extensive experimental results show that ATPFL can help users gain well-performed TP models, achieving better results than the existing TP models trained on the single-source dataset.

1. Introduction

Human Trajectory Prediction (TP) models aim to predict the movement of pedestrians [13, 20]. Their high performance greatly depends on abundant trajectory data. However, in real applications, the TP data sources are generally monitor devices scattered across different regions. They contain trajectory data in a variety of scenarios but can not be shared due to privacy protection, which brings limitations to the existing TP works. In order to break the TP data island problem, Federated Learning (FL) framework needs

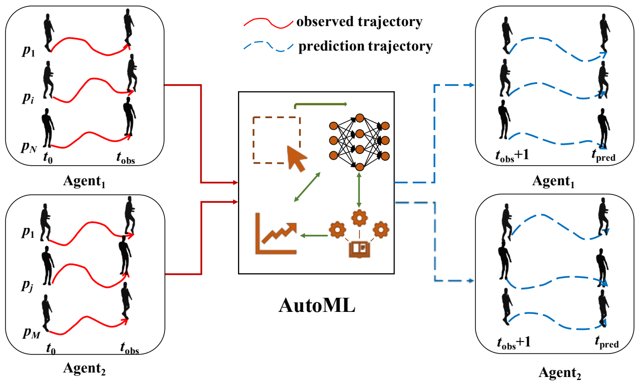

Figure 1. ATPFL combines AutoML with FL techniques on TP area, aiming to utilize multi-source TP datasets to jointly design and train the powerful TP model.

to be introduced to unite these multiple data sources jointly obtain a more robust and general TP model in a distributed and privacy-preserving manner. While this idea brings two major challenges.

On one hand, the design of TP models under the FL framework is difficult. Specifically, the TP model design process requires both heavy manual work and domain knowledge. However, FL users are generally non-experts. They fail to realize the independent development without the domain knowledge, which brings great obstacles to the general application of TP models under the FL framework.

On the other hand, the FL method suitable for TP models has not been studied or discussed yet. The existing FL works are mainly built around CNN [6,8,19] without paying attention to TP models. How to train TP models effectively under the FL framework remains to be further studied.

In this paper, we aim to tackle the above two challenges and propose ATPFL algorithm, which combines Automated Machine Learning (AutoML) with FL (as is shown in Figure 1), to federate multi-source trajectory datasets to automatically design and train powerful TP models.

*Corresponding author.

For the first challenge, in ATPFL, we design an AutoML algorithm suitable for the TP area, thus achieving the TP model's automatic design. We summarize the design process of the TP model, collect available operations for each step and identify the limitation of each operation by analyzing the existing TP works. We integrate the above experience and knowledge in a relation graph and thus construct an effective search space for the TP area.

Besides, considering the complex restrictive relations, temporal relations and technical connections among operations, we design a relation-sequence-aware strategy to effectively and efficiently explore the TP search space. This strategy can utilize the constructed relation graph, Graph Neural Network (GNN) [4] combined with Recurrent Neural Network (RNN) to learn high-level features of the selected operation sequence, and thus provide an effective reference for designing subsequent steps. Also, it can avoid invalid model design schemes by consulting the relation graph at each step, thus greatly improving the search efficiency. Compared with the traditional search strategy for AutoML which ignores relations among operations during the model design [10, 12, 21], our strategy is more suitable for the TP area.

As for the second challenge, we find appropriate federated training methods for TP models, enabling ATPFL to perform effectively and efficiently under the FL framework. We identify a method with fast convergence to support the fast evaluation of TP model candidates in AutoML, thus ensuring the search efficiency of ATPFL. Besides, we choose the most effective federated training method to train the optimal TP model discovered by ATPFL, so as to further improve the final performance of ATPFL.

Our major contributions are summarized as follows.

1. *Knowledge*: We construct a detailed knowledge graph for operations in the TP area. This graph can deepen our understanding of TP models and provide favorable help for further study.

2. *Novelty*: We simultaneously break the data island and professional restrictions, empower non-experts to combine multi-source trajectory datasets to design powerful TP models automatically.

3. *Effectiveness*: Extensive experiments show that our designed search strategy and selected federated training methods are well suited to the TP area, and helpful for obtaining more powerful TP models, which demonstrate the effectiveness of ATPFL.

2. Related Works

2.1. Trajectory Prediction Models

The deep neural network based TP models [1, 2, 11, 13, 14, 16, 20, 25, 29, 30, 32] have emerged recently as powerful tools for forecasting future trajectories of humans. Social-LSTM [1] is one of the earliest deep TP models, which applies an RNN and a pooling mechanism to model the motion pattern of pedestrians and form social features between them. Social-GAN [11] extends Social-LSTM into a generative adversarial model to further explore the multimodality of human behaviors and achieve better results. STGAT [13] presents a novel spatial-temporal graph attention network to capture both spatial and temporal features of the crowd interactions, and achieves good performance. More recently, Social-STGCNN [20] proposes to model the pedestrians' trajectories as a spatio-temporal graph, and directly manipulates over the graph to model pedestrians' interactions using a graph Convolutional Neural Network(CNN) and a temporal CNN.

These existing neural models focus on different insights to solve the TP problems, having made promising progress in real applications. In this paper, we aim to flexibly use the model design experience provided by them to support the automatic design of TP models.

2.2. Federated Training Methods

FL [31] aims to train a high-quality centralized model based on datasets that are distributed across multiple clients without sharing their data. It makes it possible to vigorously develop neural models in the privacy-preserving era, and attracts great attention of scholars. FedAvg [19] is the first federated training method designed for neural models. It uses local SGD updates in each client and builds a global model from a subset of clients with non-i.i.d. data. Per-FedAvg [7] adds the idea of personalization to FedAvg. It allows each client to perform one gradient update based on the global model using its local dataset to obtain a personalized model solution. More recently, pFedMe [6] was proposed to further improve the performance of Per-FedAvg. It allows each client to use any optimization method for multi-step updates without deviating too much from the global model parameters to obtain better personalized model solutions. It can parallelly optimize the personalized models with low complexity and achieve good results.

These federated training methods on neural models provide strong supports for TP model training under the FL framework. But previous works only analyze their performance on classification models or other motion prediction models [9, 18], while ignoring their characteristics on TP ones. In this paper, we aim to fill the gap and identify appropriate federated training methods for TP models, ensuring the search efficiency and final performance of ATPFL under the FL framework.

2.3. Neural Architecture Search Algorithms

The Neural Architecture Search (NAS) which leans to automatically search for good neural architectures [27], is an important research topic in AutoML. The existing NAS algorithms can be classified into three categories, Reinforcement Learning (RL) based methods, Evolutionary Algorithm (EA) based methods and gradient-based methods. The RL-based NAS [3, 10] uses an RNN as the controller to determine a sequence of operators and connection tokens, thus constructing networks sequentially. EA-based NAS [5,24] initializes a population of architectures first and then evolves them with their validation accuracies as fitnesses. As for the gradient-based NAS methods [12,17,21], they relax the search space to be continuous, so that the architecture can be optimized with respect to its validation performance by gradient descent.

These NAS algorithms are generally designed for CNN or GNN classification models, where operations in the search space do not have complex relations. They are unable to tackle valuable connections among TP operations to further improve the final performance, which is not well suited for TP-based NAS problems. This paper aims to fill this gap and design a more suitable NAS solution for the TP area, realizing efficient and automatic TP model design.

3. Our Approach

In this section, we first design an AutoML algorithm to realize the automated design of the TP model in ATPFL (Section 3.1). Then, we determine the appropriate federated training methods to guide ATPFL to effectively and efficiently work under the FL framework (Section 3.2).

3.1. Automatic design of TP model

We utilize the existing TP model design experience to construct an effective TP search space (3.1.2), and design a relation-sequence-aware search strategy to guide ATPFL to efficiently search for a high-performance TP model (3.1.3). Section 3.1.1 gives the notations on the TP model and defines the search target of ATPFL.

3.1.1 Notations and Search Target

Notations. Assume there are N pedestrians involved in a scene, represented as $p_1, p_2, \dots, p_N$. The position of pedestrian p_i at time-step t is denoted as $p_i^t = (x_i^t, y_i^t)$, and the set of observed history positions of all pedestrians over a time period t_{obs} is denoted as $\mathbb{X} = \left\{p_i^{1:t_{\text{obs}}} \middle| i = 1, \dots, N\right\}$. A TP model $\mathcal{M}$ can predict the upcoming trajectories of all pedestrians over a future time horizon t_{pred}, which is denoted by $\mathbb{Y} = \left\{p_i^{t_{\text{obs}}+1:t_{\text{pred}}} \middle| i = 1, \dots, N\right\}$, according to $\mathbb{X}$. We use $\hat{\mathbb{Y}} = \mathcal{M}(\mathbb{X})$ to represent the prediction of model $\mathcal{M}$, and compare $\hat{\mathbb{Y}}$ and $\mathbb{Y}$ using the Average Displacement Error (ADE) metric or a certain loss function, so as to examine the effectiveness of the TP model $\mathcal{M}$.

Search Target. Given a TP search space $\mathbb{S}$ and a federated TP dataset $\mathbb{D} = \{D_1, \dots, D_C\}$ that are distributed and non-shared across C clients, the AutoML part of ATPFL algorithm aims to find an optimal TP model $\mathcal{M}^* \in \mathbb{S}$ that minimizes the overall validation ADE score on $\mathbb{D}$.

$$\begin{aligned} \mathcal{M}^* &= \arg\min_{\mathcal{M}\in\mathbb{S}} \text{ADE}_{\mathbb{D}_{\text{val}}}(\mathbf{W}^*_{\mathcal{M}}, \mathcal{M}) \\ \text{s.t. } \mathbf{W}^*_{\mathcal{M}} &= \arg\min_{\mathbf{W}} \mathcal{L}_{\mathbb{D}_{\text{train}}}(\mathbf{W}, \mathcal{M}) \end{aligned} \tag{1}$$

where $\mathcal{L}_{\mathbb{D}_{\text{train}}}(\mathbf{W}, \mathcal{M})$ denotes the overall training loss of TP model $\mathcal{M}$ on $\mathbb{D}$ under weights $\mathbf{W}$, and $\mathbf{W}^*_{\mathcal{M}}$ can be learned using a federated training method.

3.1.2 Search Space

We sum up 5 stages of the TP model design by learning from the existing TP models (these stages are common to existing TP models). We apply 10 parameters to describe the main contents of these stages and extract effective operations of each parameter from 5 state-of-the-art TP models, including SGCN [26] ($\mathcal{M}_1$), LB-EBM [22] ($\mathcal{M}_2$), Social-STGCNN [20] ($\mathcal{M}_3$), Social Ways [2] ($\mathcal{M}_4$), STGAT [13] ($\mathcal{M}_5$). Table 1 summarizes all contents of this part.

In ATPFL, we utilize the experience information in Table 1 to construct an effective TP search space. Specifically, apart from the 10 parameters defined above, which outline the design process of a TP model, we add two parameters: $\mathbf{FExM_{add}}$ and $\mathbf{FEnM_{add}}$ to TP search space, corresponding to an additional group of feature extraction and enhancement operation in Stage 2, so as to acquire more powerful TP models. We apply these 12 parameters to describe the design scheme of a TP model, allowing them to be set to their options in the fourth column of Table 1, and thus obtain a search space with diversified TP models (about 1.3×10^6 TP models design scheme contained in the search space).

3.1.3 Relation-Sequence-Aware Search Strategy

In this part, we aim to design an effective strategy for ATFFL to efficiently search for the high-performance TP model from the huge search space designed in Section 3.1.2.

Features of TP search space. We notice that TP search space is more complex than the traditional CNN search space, where operations are simple and do not have restrictions on use. There are multiple associations among operations in the TP search space (Details are shown in Figure 2):

R$_1$ **Temporal Relations.** Each operation in the search space corresponds to only one of the stages of TP model design, and the operations of Stage 1 to Stage 5 should be sequentially selected (as follows) during the TP model design process.

Table 1. 5 stages of the TP model design. The 4th column lists options of 10 parameters used to describe a TP model. Options are extracted from 5 TP models: SGCN [26] ($\mathcal{M}_1$), LB-EBM [22] ($\mathcal{M}_2$), Social-STGCNN [20] ($\mathcal{M}_3$), Social Ways [2] ($\mathcal{M}_4$), STGAT [13] ($\mathcal{M}_5$).

Stage	Function Description	Involved Operations (Parameters)	Solutions (Parameters' Options) Provided by Existing Works
Stage 1: Data Preprocessing Stage	Enhance the representation power of the input.	Input Processing Method (IPM) $\mathbb{X}' = \mathbf{IPM}(\mathbb{X})$	$\mathbf{IPM}_1$: Real Position ($\mathcal{M}_2$) $\mathbf{IPM}_2$: Relative Position ($\mathcal{M}_1, \mathcal{M}_3, \mathcal{M}_5$) $\mathbf{IPM}_3$: Real + Relative Position ($\mathcal{M}_4$)
Stage 2: Feature Extraction Stage	Capture features of the historical trajectory.	Feature Extraction Method (FExM) $\mathbb{F} = \mathbf{FExM}(\mathbb{X}', \mathbb{X})$	$\mathbf{FExM}_1$: Sparse Graph Convolution Network ($\mathcal{M}_1$) $\mathbf{FExM}_2$: Multilayer Perceptron Network ($\mathcal{M}_2$) $\mathbf{FExM}_3$: Spatio-Temporal Graph CNN ($\mathcal{M}_3$) $\mathbf{FExM}_4$: LSTM based Motion Encoder Module ($\mathcal{M}_4$) $\mathbf{FExM}_5$: GAT-based Crowd Interaction Modeling ($\mathcal{M}_5$)
		Feature Enhancement Method (FEnM) $\mathbb{F}' = \mathbf{FEnM}(\mathbb{X}', \mathbb{X}, \mathbb{F})$	$\mathbf{FEnM}_1$: None ($\mathcal{M}_1, \mathcal{M}_3$) $\mathbf{FEnM}_2$: Latent Belief Energy-based Module ($\mathcal{M}_2$) $\mathbf{FEnM}_3$: Attention Pooling Module ($\mathcal{M}_4$) $\mathbf{FEnM}_4$: LSTM-based Temporal Correlation Modeling ($\mathcal{M}_5$)
Stage 3: Feature Fusion Stage	Combine features of historical trajectory.	Feature Fusion Method (FFM) $\mathbb{F}_{\text{all}} = \mathbf{FFM}(\mathbb{F}, \mathbb{F}')$	$\mathbf{FFM}_1$: Concentrate All Features in Stage 2 ($\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$) $\mathbf{FFM}_2$: Concentrate All Features in Stage 2 and Noise ($\mathcal{M}_4, \mathcal{M}_5$)
Stage 4: Trajectory Prediction Stage	Transform the output of the Stage 3 into the expected prediction.	Prediction Processing Structure (PPS) $\hat{\mathbb{Y}} = \mathbf{PPS}(\mathbb{F}_{\text{all}})$	$\mathbf{PPS}_1$: Time Convolution Network ($\mathcal{M}_1$) $\mathbf{PPS}_2$: Time-Extrapolator Convolution Neural Network ($\mathcal{M}_3$) $\mathbf{PPS}_3$: Multiple Fully-Connected Layer ($\mathcal{M}_2, \mathcal{M}_4$) $\mathbf{PPS}_4$: LSTM + Fully-Connected Layer ($\mathcal{M}_5$)
		Output Contents (OC)	$\mathbf{OC}_1$: Predict Coordinates Sequentially ($\mathcal{M}_4$) $\mathbf{OC}_2$: Predict Coordinates Directly ($\mathcal{M}_2, \mathcal{M}_5$) $\mathbf{OC}_3$: Predict Parameters of Bi-Variate Gaussian Distribution ($\mathcal{M}_1, \mathcal{M}_3$)
Stage 5: Model Training Stage	Determine suitable training setting for the designed Trajectory Prediction model.	Loss Function (LF)	$\mathbf{LF}_1$: L_2 loss ($\mathcal{M}_2, \mathcal{M}_4, \mathcal{M}_5$) $\mathbf{LF}_2$: Distribution-based Negative Log-Likelihood Loss ($\mathcal{M}_1, \mathcal{M}_3$)
		Training Mode (TM)	$\mathbf{TM}_1$: General LF-based Model Training ($\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$) $\mathbf{TM}_2$: Generative Adversarial Network based Model Training ($\mathcal{M}_4$) $\mathbf{LM}_3$: Variety Loss based Model Training ($\mathcal{M}_5$)
		Learning Rate (LR)	$\mathbf{LR}_1$: 1e-2 ($\mathcal{M}_1, \mathcal{M}_3$) $\mathbf{LR}_2$: 1e-4 ($\mathcal{M}_2$) $\mathbf{LR}_3$: 0.0015 $\mathbf{LR}_4$: 1e-3 ($\mathcal{M}_4, \mathcal{M}_5$)
		Optimization Function (OF)	$\mathbf{OF}_1$: Adam ($\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_4, \mathcal{M}_5$) $\mathbf{OF}_2$: SGD ($\mathcal{M}_3$)

$$\mathbf{IPM}{\to}\mathbf{FExM}{\to}\mathbf{FEnM}{\to}\mathbf{FExM_{add}}{\to}\mathbf{FEnM_{add}}$$
$$\to\mathbf{FFM}{\to}\mathbf{PPS}{\to}\mathbf{OC}{\to}\mathbf{LF}{\to}\mathbf{TM}{\to}\mathbf{LR}{\to}\mathbf{OF}$$

$\mathbf{R_2}$ **Restrictive Relations.** Some operations may fail to cooperate with certain operations to construct effective TP models due to special requirements.

For example, $\mathbf{LF}_2$ in Table 1 is designed for TP models which estimate bi-variate distribution ($\mathbf{OC}_3$), not applicable to $\mathbf{OC}_1$ and $\mathbf{OC}_2$. $\mathbf{FExM}_2$ is unable to deal with variable-length trajectories, and thus fail to predict coordinates sequentially ($\mathbf{OC}_1$).

$\mathbf{R_3}$ **Technical Connections.** Some operations may apply the same type of neural architectures or techniques.

For example, both $\mathbf{FExM}_3$ and $\mathbf{PPS}_1$ use CNN, $\mathbf{FExM}_5$ and $\mathbf{FEnM}_3$ apply the attention mechanism.

These association relationships are valuable and can help improve the search performance: $\mathbf{R_1}$–$\mathbf{R_3}$ can assist search strategy better understanding characteristics of each operation, obtaining better TP models; $\mathbf{R_1}$ and $\mathbf{R_2}$ can guide the search strategy to avoid invalid operation combinations and thus improve the search efficiency. In addition, we note that the valid and optimal options of the subsequent operations can be affected by the selected TP operation sequence.

Relation-Sequence-Aware Strategy. Based on above features, we design a relation-sequence-aware search strategy in ATPFL, which can utilize relational information among operations, combining with the historical operation sequence to sequentially and efficiently select the optimal subsequent operations, and thus obtain effective TP models. Figure 2 gives the overall framework of our strategy.

Our strategy contains two parts, i.e., GNN based embedding learning and masked RNN optimizer.

Part1: GNN based Embedding Learning. We firstly use GNN to learn the effective embedding representation of each operation from relational information in the TP search space. We treat the operations in search space as nodes and transform $\mathbf{R_1}$–$\mathbf{R_3}$ into three different types of edges to connect related operation pairs, and thus construct a heterogeneous graph to represent relations among operations. Then, we introduce FAGCN [4], an effective GCN with a self-gating mechanism, to automatically learn associations between nodes and obtain high-level node features by adaptively integrating related neighboring information.

Note that different types of neighbors may make different contributions to the final embedding of the target operation node. Therefore, we make FAGCN adaptively learn the importance of different edge types on the target node.

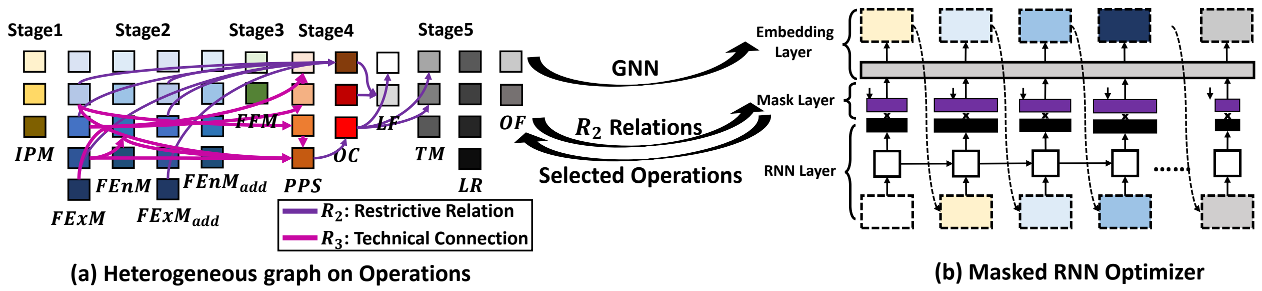

Figure 2. The overall framework of the relation-sequence-aware search strategy in ATPFL. Note that $\mathbf{R_1}$, i.e., temporal relation, exists between adjacent operations with different types and we omit these edges in the heterogeneous graph.

The embedding learning formula for each operation in the search space is as follows:

$$\mathbb{E}_i' = \epsilon \cdot \mathbf{x}_i + \sum_{k=1}^{3} \sum_{j \in \mathcal{N}_{i,\mathbf{R_k}}} \frac{\alpha_{i,j,\mathbf{R_k}}}{\sqrt{d_{i,\mathbf{R_k}} d_{j,\mathbf{R_k}}}} \mathbf{x}_j \tag{2}$$

where $\mathbf{x}_i$, $\mathcal{N}_{i,\mathbf{R_k}}$ and $d_{i,\mathbf{R_k}}$ denote the initial embedding representation, neighbor set and node degree w.r.t. edge $\mathbf{R_k}$ of node i, respectively. The attention coefficients $\alpha_{i,j,\mathbf{R_k}}$ are computed based on the trainable parameter vector $\mathbf{a_{R_k}}$.

$$\alpha_{i,j,\mathbf{R_k}} = \mathbf{tanh}(\mathbf{a}_{\mathbf{R_k}}^\top [\mathbf{x}_i, \mathbf{x}_j]) \tag{3}$$

Part2: Masked RNN Optimizer. Then, we use RNN, heterogeneous graph and the learned high-level operation embeddings to sequentially and efficiently obtain the optimal and valid TP model design scheme.

We fed embeddings of the selected operations $o_{1\sim t} = (o_1, \dots, o_t)$ into RNN sequentially to extract the effective features $\mathbb{F}_{o_{1\sim t}}$ of the historical operation sequence. And predict the possibility $\mathbb{P}_{t+1}^{o_{1\sim t}}$ that each next-step operation is the optimal according to $\mathbb{F}_{o_{1\sim t}}$.

$$\mathbb{P}_{t+1}^{o_{1\sim t}} = \mathbf{softmax}\left(\mathbf{FC}\left(\mathbf{RNN}\left(\mathbb{E}_{o_1}, \dots, \mathbb{E}_{o_t}\right)\right)\right) \tag{4}$$

$\mathbb{P}_{t+1}^{o_{1\sim t}} \in \mathbb{R}^{|\mathcal{S}_{t+1}|}$, where $\mathcal{S}_{t+1}$ denotes the set of operations in the next step, can guide us to explore more promising TP models, but may recommend invalid TP model design schemes due to ignorance of $\mathbf{R_2}$ restrictive relations among operations. In order to avoid invalid exploration and further improve the search efficiency, we construct a mask vector $\mathbb{M}_{t+1}^{o_{1\sim t}} \in \mathbb{R}^{|\mathcal{S}_{t+1}|}$ for $\mathbb{P}_{t+1}^{o_{1\sim t}}$ to shield out invalid next-step operations. Specifically, we identify $\mathbf{R_2}$ neighbors of $o_{1\sim t}$ from $\mathcal{S}_{t+1}$, setting their mask values to 0 while keep the other values to 1, and thus obtain $\mathbb{M}_{t+1}^{o_{1\sim t}}$. With the help of $\mathbb{M}_{t+1}^{o_{1\sim t}}$, $\mathbb{P}_{t+1}^{o_{1\sim t}}$ is modified to $\widetilde{\mathbb{P}_{t+1}^{o_{1\sim t}}}$, and thus RNN can filter out the effective operation options and obtain the next best operation more efficiently.

$$\widetilde{\mathbb{P}_{t+1}^{o_{1\sim t}}} = \mathbf{softmax}\left(\mathbb{M}_{t+1}^{o_{1\sim t}} \cdot \mathbb{P}_{t+1}^{o_{1\sim t}}\right) \tag{5}$$

Repeat above steps, then a promising and valid TP model design scheme can be obtained.

As for the model parameters θ involved in the GCN and RNN optimizer, we optimize their weights following the reinforce policy [10, 28].

$$\begin{aligned} &\nabla_\theta \mathbb{E}_{P(o_{1\sim 12};\theta)}[\text{Reward}] \\ &= \sum_{t=1}^{12} \mathbb{E}_{P(o_{1\sim t};\theta)} \left[\nabla_\theta \log \widetilde{\mathbb{P}_t^{o_{1\sim t-1}}} \left(\text{Reward} - b\right) \right] \end{aligned} \tag{6}$$

where b is an exponential moving average of the previous model rewards, and $Reward$ is the negative ADE score of the TP model generated by RNN optimizer. This reinforce strategy can effectively update the model weights by maximizing the expected benefits of the strategy, guiding our search strategy to recommend better TP models.

3.2. Federated Training of the TP Model

ATPFL needs to find suitable federated TP model training methods, to be able to work effectively and efficiently under the FL framework.

Specifically, the optimal model search stage of ATPFL generally needs to evaluate many TP models, and requires a federated training method with fast convergence. In this way, ATPFL can distinguish TP models with different performances using few federated training epochs, and thus reducing evaluation cost and increasing search efficiency. In addition, the optimal TP model training stage of ATPFL requires the most effective federated training method. In this way, ATPFL can gain a more powerful TP model under the FL framework, further improving the final performance.

Based on the above demands, we analyze the convergence speed and federated performance of three federated training methods, including FedAvg [19], Per- FedAvg [7] and pFedMe [6], on TP models, aiming to find appropriate ones to work for ATPFL.

Three Federated Training Methods. Given a dataset $\mathbb{D} = \{D_1, \dots, D_C\}$ that are distributed and non-shared across C clients, federated training methods optimize the

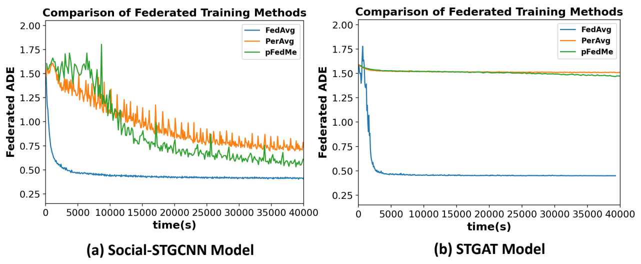

Figure 3. Performance curves of different federated training methods on TP models and a federated TP dataset. We simulates this federated dataset by equally distributing trajectory data of ETH [23] and UCY [15] to 20 clients.

global model weight $\mathbf{w}$ as follows.

$$\begin{gathered}\nabla F(\mathbf{w}) = \frac{1}{C}\sum_{i=1}^{C}\nabla F_i(\mathbf{w}) \\ \mathbf{w} = \mathbf{w} + \eta\nabla F(\mathbf{w})\end{gathered} \tag{7}$$

where $F_i(\mathbf{w})$ is the local training loss of client i, and is calculated differently in different federated training methods. FedAvg directly considers $f_i(\mathbf{w})$, i.e., the training loss of $\mathbf{w}$ on local dataset D_i, as $F_i(\mathbf{w})$:

$$F_i(\mathbf{w}) := f_i(\mathbf{w}) \tag{8}$$

Per-FedAvg uses personalized models to obatin $F_i(\mathbf{w})$:

$$F_i(\mathbf{w}) := f_i(\theta_i(\mathbf{w})) = f_i(\mathbf{w} - \alpha\nabla f_i(\mathbf{w})) \tag{9}$$

pFedMe generates more flexible personalized weights ϑ_i under the guidance of global weight $\mathbf{w}$ to measure $F_i(\mathbf{w})$:

$$F_i(\mathbf{w}) := \min_{\vartheta_i}\left\{f_i(\vartheta_i) + \frac{\lambda}{2}\|\vartheta_i - \mathbf{w}\|^2\right\} \tag{10}$$

Features of Three Methods. We analyze the performance curves of above three methods on existing TP models (Figure 3 gives an example). We find that FedAvg coverage the fastest among them at the early training stage, and also achieves the best final performance among them. pFedMe and Per-FedAvg are not prominent in both convergence speed and federated performance, and thus not suitable for our ATPFL algorithm. These findings show us that powerful TP models heavily rely on abundant data sources, and personalized training methods, which pay more attention to the local data, do not help much in TP area.

Based on these observations, we use FedAvg to quickly evaluate TP models during the TP model search stage of ATPFL, and train the optimal TP model in ATPFL. With its help, ATPFL can achieve better performance under FL framework.

4. Experiments

In this section, we examine the performance of ATPFL. We analyze the importance of federated training on TP models, and compare the AutoML part of ATPFL with existing AutoML algorithms (Section 4.2). In addition, ablation experiments are conducted to analyze the relation-sequence-aware search strategy designed in ATPFL (Section 4.3). All experiments are implemented using Pytorch.

4.1. Experimental Setup

Datasets. In order to evaluate the performance of our approach under FL framework, we construct a federated TP dataset $\mathbb{D}$ by equally distributing trajectory data of two publicly available datasets: ETH [23] and UCY [15], to 20 clients. The ETH dataset contains two scenes of real-world human trajectories, i.e., ETH and Hotel, each with 750 different pedestrians. The UCY dataset has 3 components: ZARA01, ZARA02 and UNIV, containing two scenes with 786 people. In total, our federated TP dataset $\mathbb{D}$ contains 5 sets of data with 4 different trajectory scenes. For each client, we hold the 60%, 20%, 20% of its local dataset as the training set, validation set and test set, respectively. As for the evaluation of TP models under single-source datasets, we take ETH, Hotel, ZARA01, ZARA02 and UNIV as 5 single-source datasets applying the same split ratio.

Evaluation Metrics. We use Average Displacement Error (ADE) [23] and Final Displacement Error (FDE) [1] to examine prediction errors of the TP models.

$$\begin{aligned}\mathrm{ADE} &= \frac{\sum_{i\in\mathcal{P}}\sum_{t=t_{\mathrm{obs}}+1}^{t_{\mathrm{pred}}}\left\|((\hat{x}_t^i,\hat{y}_t^i)-(x_t^i,y_t^i))\right\|_2}{|\mathcal{P}|\cdot t_{\mathrm{pred}}} \\ \mathrm{FDE} &= \frac{\sum_{i\in\mathcal{P}}\left\|((\hat{x}_t^i,\hat{y}_t^i)-(x_t^i,y_t^i))\right\|_2}{|\mathcal{P}|},\ t=t_{pred}\end{aligned} \tag{11}$$

where $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ is the set of pedestrians, $(\hat{x}_t^i, \hat{y}_t^i)$ are the predicted coordinates at time t and (x_t^i, y_t^i) are the ground-trurth coordinates.

As for the final federated performance of TP models, we use the federated ADE/FDE score on test set to compare their performance under FL framework:

$$\mathrm{ADE}_{\mathbb{D}_{\mathrm{test}}} = \sum_{i=1}^{C}\frac{|D_{i,\mathrm{test}}|}{|\mathbb{D}_{\mathrm{test}}|}\mathrm{ADE}(\mathcal{M}, D_{i,\mathrm{test}}) \tag{12}$$

where $D_{i,\mathrm{test}}$ is the test set of client i, and $\mathbb{D}_{\mathrm{test}}$ denotes all test data in the federated TP dataset $\mathbb{D}$. Replace ADE to FDE then we get $\mathrm{ADE}_{\mathbb{D}_{\mathrm{test}}}$.

Baselines. We compare ATPFL with two popular search strategies for AutoML: an RL search strategy that combines recurrent neural network controller [10] and EA-based search strategy for multi-objective optimization [5], and a commonly used baseline in AutoML, Random Search.

Table 2. Performance comparison of ATPFL, AutoML algorithms and manually designed TP models. The first part examines the performance of existing TP models on single-source TP datasets. The second part compares different federated training methods on a TP model. The third part compares different AutoML methods under FL framework. Best-i denotes the i^{st} best ADE score achieved by the clients.

FL Methods	**TP Models**	**ADE/FDE on single-sorce dataset**								
		AVG	**STD**	**MIN**	**MAX**	**Datasets**				
						ETH	**Hotel**	**UNIV**	**ZARA01**	**ZARA02**
None	**Social-STGCNN**	0.78/1.06	0.4142 /0.7060	0.31/0.45	2.04/2.71	2.04/2.71	0.45/0.52	0.53/0.94	0.46/0.70	0.31/0.45
	Social Ways	1.07/1.88	1.4042 /3.9911	0.23/0.40	3.39/5.76	3.39/5.76	0.43/0.90	0.23/0.40	0.35/0.56	0.94/1.79
	STGAT	0.97/1.78	0.2421 /0.6898	0.65/1.27	1.94/3.41	1.94/3.41	0.68/1.28	0.68/1.26	0.89/1.67	0.65/1.27
	SGCN	0.38/0.58	0.0761 /0.1516	**0.13/0.18**	0.92/1.32	0.92/1.32	0.13/0.18	0.34/0.50	0.26/0.49	0.26/0.39

FL Methods	**TP Models**	**Federated ADE/FDE under FL framework**								
		AVG	**STD**	**MIN**	**STD**	**Best-1**	**Best-2**	**Best-3**	**Best-4**	**Best-5**
FedAvg	**SGCN**	**0.32/0.59**	**0.0004 /0.0023**	**0.27/0.50**	**0.35/0.66**	0.27/0.50	0.29/0.51	0.29/0.52	0.29/0.52	0.30/0.55
Per-FedAvg	**SGCN**	0.50/0.86	0.0013 /0.0046	0.42/0.73	0.55/0.98	0.42/0.73	0.44/0.76	0.45/0.78	0.46/0.79	0.47/0.79
pFedMe	**SGCN**	0.42/0.75	0.0005 /0.0022	0.37/0.65	0.46/0.83	0.37/0.65	0.38/0.67	0.39/0.69	0.40/0.71	0.41/0.71

FL Methods	**AutoML Methods**	**Federated ADE/FDE under FL + AutoML framework**								
		AVG	**STD**	**MIN**	**MAX**	**Best-1**	**Best-2**	**Best-3**	**Best-4**	**Best-5**
FedAvg	**RL**	0.40/0.65	0.0006 /0.0038	0.36/0.53	0.46/0.79	0.36/0.53	0.36/0.56	0.37/0.58	0.37/0.59	0.37/0.60
FedAvg	**Evolution**	0.57/1.14	0.0008 /0.0042	0.52/1.05	0.63/1.26	0.52/1.05	0.53/1.05	0.54/1.06	0.541/1.10	0.55/1.11
FedAvg	**Random**	0.49/1.02	0.0008 /0.0051	0.45/0.89	0.55/0.11	0.45/0.89	0.46/0.92	0.46/0.92	0.47/0.93	0.47/0.96
FedAvg	**ATPFL**	**0.30/0.57**	**0.0003 /0.0029**	0.27/0.46	**0.35/0.65**	0.27/0.46	0.28/0.47	0.28/0.48	0.28/0.51	0.28/0.53

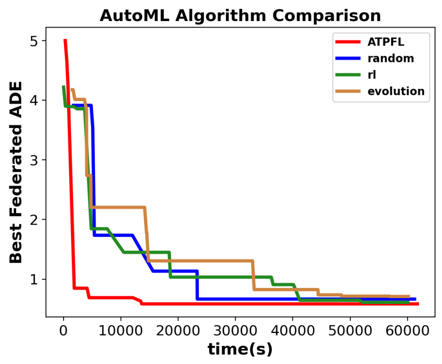

Figure 4. The federated performance of the optimal TP model searched by different AutoML algorithms.

We replace the relation-sequence-aware strategy in ATPFL to these three search strategies, so as to examine their performance under TP models and FL framework.

In addition, we take 4 state-of-the-art human-invented TP models: Social Ways [2], STGAT [13], NS [20] and SGCN [26], as baselines, to show the importance of automatic TP model design under FL framework.

1. Social Ways: It uses a GAN to sample plausible predictions for any agent in the scene avoid mode collapsing and dropping.

2. STGAT: A spatial temporal graph attention network which based on a sequence-to-sequence architecture to predict future trajectories of pedestrians.

3. Social-STGCNN: It substitutes the need of aggregation methods by modeling the interactions as a graph.

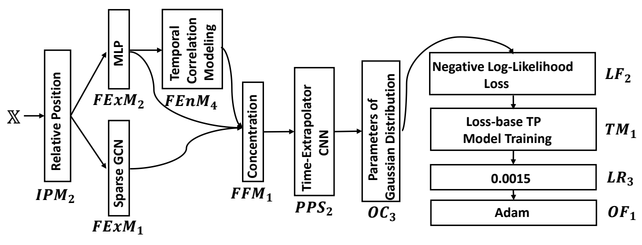

Figure 5. Optimal TP model design scheme searched by ATPFL.

4. SGCN: It models the sparse directed interaction and motion tendency with a sparse directed graph to improve the predicted trajectories.

Implementation Details. In ATPFL, the embedding size and hidden size are set to 100. RNN optimizer is trained with the Adam optimizer with a learning rate of 3.5e-4. For each TP model candidate, we train it for 5 epochs using FedAvg. After ATPFL searches for 1.5 GPU days, we choose the best TP model that achieves the highest federated performance on validation datasets, and train it for 500 epochs using FedAvg. As for the compared AutoML algorithms, we follow implementation details reported in their papers, and control the running time of each AutoML algorithm to be the same.

4.2. Performance Evaluation

The performance of 4 AutoML algorithms under FL framework and 4 manually designed TP models are shown in Table 2 and Figure 4, and Figure 5 visualizes TP models searched by ATPFL algorithm.

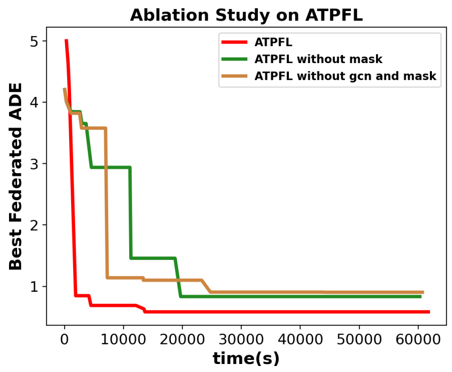

Figure 6. The federated performance of the optimal TP model searched by different versions of ATPFL.

We can observe from the third part of Table 2 and Figure 4 that ATPFL exceeds the other AutoML algorithms on the federated TP dataset, discovering more powerful TP models within the same search time. This result shows the effectiveness of our proposed algorithm and demonstrates the importance of considering the relations among operations in the TP search space. With the help of relation-sequence-aware search strategy, ATPFL can take full advantage of the relations among operations in the search space to avoid useless evaluations to improve search efficiency, and learn effective features of operations to further improve the prediction precision, thus providing good recommendation results more efficiently.

In addition, we observe from the second and third part of Table 2 that TP models discovered by AutoML algorithms generally outperform existing human-invented TP models under FL framework. This result shows the practicability and validity of the automatic design of the TP model under FL framework. Our proposed ATPFL empowers non-experts to jointly deploy appropriate and powerful TP models, greatly reduces the labour of human experts, which is meaningful and practical.

We also notice that federated training methods, especially FedAvg, can obtain a more powerful TP model, compared to the non-federated method. This result shows the importance of FL on TP area. The high performance of the TP model heavily relies on a large number of trajectory data with multiple scenes, and FL framework is essential for obtaining powerful TP models.

4.3. Ablation Experiments

We further investigate the effect of the GNN based embedding method and the masked RNN optimizer, two core components of our relation-sequence-aware search strategy, on the performance of ATPFL algorithm using the following two variants of ATPFL, thus verifying the innovations presented in this paper.

1. *ATPFL without mask*: This version of ATPFL algorithm removes the masking operation from the masked RNN optimizer. It ignores the restrictive relations among operations, assuming that all TP model design schemes contained in the search space are valid.

2. *ATPFL without GCN and mask*: This version of ATPFL algorithm applies the RNN optimizer without the masking operation and replaces the GNN based embedding method to a common embedding layer. It ignores the multiple relations among operations in the search space, and does not apply GNN to extract high-level representations of the various operations.

Corresponding results are shown in Figure 6, we can see that ATPFL has much better performance than *ATPFL without mask*. Owing to neglect of restrictive relations among operations in the search space, a great amount of invalid design schemes for the TP model are formed in *ATPFL without mask*. These noisy choices increase the search difficulty making *ATPFL without mask* less effective under the same search time. This result further demonstrates the importance of considering restrictive relations among operations when designing the AutoML tools for TP area, and the rationality of our proposed algorithm.

Besides, we observe that *ATPFL without GCN and mask* performs much worse than ATPFL. This result shows us the significance and necessity of considering multiple relations among operations and using GNN technique in the ATPFL algorithm. GNN can extract high-level features of each operation in the search space by adaptively aggregating information from highly relevant operations, and thus providing a more rational basis for recommending an effective design scheme of the TP model. Therefore, applying the GNN technique in ATPFL is reasonable and effective.

5. Conclusion and Future Works

In this paper, we combine AutoML and FL techniques on the TP area, and propose ATPFL to help users to federate private and multiple-source trajectory datasets to automatically design and train powerful TP models. As we know, this is the first TP work that simultaneously breaks data island and technical limitations. Extensive experiments show that our designed search strategy and selected federated training methods are well suited to the TP area. They empower ATPFL to gain well-performed TP models under the FL framework, achieving better results than manually designed TP models trained on the single-source dataset. In future works, we will try to further enrich our search space, and design a more efficient search strategy for further improving the performance of ATPFL.

Acknowledgements

This paper was supported by NSFC grant (U1866602).

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 *CVPR*, pages 961–971, 2016. 2, 6

[2] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *CVPR*, pages 0–0, 2019. 2, 3, 4, 7

[3] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. In *ICML*, volume 70, pages 459–468. PMLR, 2017. 3

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *AAAI*, pages 3950–3957. AAAI Press, 2021. 2, 4

[5] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. RENAS: reinforced evolutionary neural architecture search. In *CVPR*, pages 4787–4796. Computer Vision Foundation / IEEE, 2019. 3, 6

[6] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020. 1, 2, 5

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020. 2, 5

[8] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS*, 2020. 1

[9] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. PMF: A privacy-preserving human mobility prediction framework via federated learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(1):10:1–10:21, 2020. 2

[10] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In Christian Bessiere, editor, *IJCAI*, pages 1403–1409. ijcai.org, 2020. 2, 3, 5, 6

[11] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, pages 2255–2264, 2018. 2

[12] Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. DSRNA: differentiable search of robust neural architectures. In *CVPR*, pages 6196–6205. Computer Vision Foundation / IEEE, 2021. 2, 3

[13] Yingfan Huang, HuiKun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, pages 6272–6281, 2019. 1, 2, 3, 4, 7

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

[15] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007. 6

[16] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *CVPR*, pages 5725–5734, 2019. 2

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*. OpenReview.net, 2019. 3

[18] Nathalie Majcherczyk, Nishan Srishankar, and Carlo Pinciroli. Flow-fl: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. In *ICRA*, pages 8836–8842. IEEE, 2021. 2

[19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 2, 5

[20] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *CVPR*, pages 14424–14432, 2020. 1, 2, 3, 4, 7

[21] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. ASAP: architecture search, anneal and prune. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 2020. 2, 3

[22] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *CVPR*, pages 11814–11824. Computer Vision Foundation / IEEE, 2021. 3, 4

[23] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268. IEEE, 2009. 6

[24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, pages 4780–4789. AAAI Press, 2019. 3

[25] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *CVPR*, pages 1349–1358, 2019. 2

[26] Liushuai Shi, Le Wang, Chengjiang Long, Sanping Zhou, Mo Zhou, Zhenxing Niu, and Gang Hua. SGCN: sparse graph convolution network for pedestrian trajectory prediction. In *CVPR*, pages 8994–9003. Computer Vision Foundation / IEEE, 2021. 3, 4, 7

[27] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *CVPR*, pages 6418–6427. Computer Vision Foundation / IEEE, 2021. 2

[28] Williams and J. Ronald. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. 5

[29] Yanyu Xu, Zhixin Piao, and Shenghua Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *CVPR*, pages 5275–5284, 2018. 2

[30] Hao Xue, Du Q Huynh, and Mark Reynolds. Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1186–1194. IEEE, 2018. 2

[31] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019. 2

[32] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *CVPR*, pages 12085–12094, 2019. 2