INTRAFORCE: Intra-Cluster Reinforced Social Transformer for Trajectory Prediction

Negar Emami, Antonio Di Maio, Torsten Braun

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

Abstract—**Predicting mobile users' trajectories accurately is essential for improving the performance of wireless networks and autonomous systems. In this paper, we tackle the problem of trajectory prediction in a multi-agent scenario where the social interaction among users is taken into consideration. We propose Intra-Cluster Reinforced Social Transformer (INTRAFORCE), a novel system to design and train Social-Transformer neural networks that learn the spatio-temporal interactions among neighboring mobile users and predict their joint future trajectories. Unlike state-of-the-art social-aware trajectory predictors that either miss the large-distance interactions or are computationally expensive due to the pooling of all users' interactions, INTRAFORCE clusters users with similar trajectories and learns their interactions. INTRAFORCE performs Neural Architecture Search to optimize each transformer's architecture to fit each cluster's user mobility features using Reinforcement Learning. Through experimental validation, we show that INTRAFORCE outperforms several state-of-the-art trajectory predictors on five widely used small-scale pedestrian mobility datasets and one large-scale privacy-oriented cellular mobility dataset by achieving lower prediction error, training time, and computational complexity.**

Keywords: Social-aware Trajectory Prediction, Transformers, Reinforcement Learning, Neural Architecture Search, Clustering.

I. Introduction

Accurate user mobility prediction can improve Quality of Service (QoS) of network applications through adaptive and anticipatory network management, such as proactive caching, resource allocation, service migration, and handover management [1]. On the other hand, trajectory prediction is of great importance in urban safety, autonomous driving, and robot-to-human interacting systems. In the real world, mobile users such as pedestrians or vehicles can move along similar paths during certain hours of the day, either coincidentally or intentionally as a group trip. When users move in a crowded public space, they normally follow common social rules such as estimating other users' mobility status (e.g., position and speed) and respecting their space or avoiding to collide with obstacles. This suggests that there is a strong mutual influence among users' mobility patterns and decisions.

Understanding social interactions and spatio-temporal dependencies among neighboring mobile users' paths can help to better predict their complex motion behaviors. Recently, deep learning models such as Recurrent Neural Networks (RNNs), have proven to be successful to tackle the problem of sequential-data prediction. RNN-based social models consists of a group of recurrent networks merged in a shared social pool distributing information between neighbor trajectory users [2], [3]. However, despite the advancements of RNNs, they are criticized for their inability to capture long-term dependencies and complex social interactions efficiently and for their slow training time. To alleviate the aforementioned limitations, attention-based neural networks, especially Transformers (TF) [4], have shown great improvements over RNNs. Regardless of the neural network type, existing social-aware trajectory predictors still suffer from several shortcomings. First, they either capture interactions within a fixed local area around each agent, which results in missing larger distant interactions, or they consider all agents of the scene, which ends up in high computational costs. Second, they design the neural network architecture following human-based heuristics, which is a time-consuming and error-prone process. Third, they apply the same neural architecture to every user data type not guaranteeing the optimal prediction performance.

In this article, the research question we aim to solve is the design and study of a system that automatically builds a trajectory predictor, leveraging the mutual dependency of inter-user mobility to reduce the model size without affecting accuracy. We propose INTRAFORCE, a system to design and train *Social-Transformers* to capture joint interactions and reduce the required computation by measuring user trajectory similarity and clustering users before feeding them to a social-aware trajectory predictor. For each cluster, INTRAFORCE employs a Reinforcement Learning (RL) agent aiming at maximizing the performance of the multi-agent model based on the mobility features of the cluster.

The rest of this paper is organized as follows. Section II presents the related works. Section III describes INTRAFORCE operation. Section IV evaluates INTRAFORCE performance. Finally, Section V concludes and summarizes the contributions of this work.

II. Related Work

In recent years, data-driven social mobility predictors are gaining popularity compared to the previously proposed *Social-force* models, which use simple repulsive and attraction forces [7]. The vast majority of modern human-trajectory predictors are based on deep learning models, such as RNNs, Long Short-Term Memorys (LSTMs), Convolutional Neural Networks (CNNs), and attention-based neural networks, such

TABLE I
FEATURES OF SOCIAL-AWARE TRAJECTORY PREDICTORS

Trajectory Predictor	Neural Network	Social Module	NAS
Alahi et al. [2]	LSTM	Social Pool	-
Gupta et al. [3]	LSTM-GAN	Social Pool and Pooling Vector	-
Sadeghian et al. [5]	LSTM-GAN	Attention	-
Yu et al. [6]	Transformer	Graph Convolution	-
INTRAFORCE	Transformer	Social Pool and Clustering	RL

as Transformers, which require less computation and achieve higher prediction accuracy compared to social-force models due to their better modeling of sequential patterns [1], [8], [9]. Instead of modeling kinetic forces and energy potentials as in social-force models, social-pooling [2], [3], attention [10], [5], and graph [6], [11] mechanisms complement neural networks to share information about neighboring user's trajectories to capture complex interactions in crowded environments.

Alahi et. al. [2] introduce the Social-LSTM, which relies on a social pooling layer that connects multiple LSTMs to model interaction among pedestrians. Social-LSTM considers user interactions within a fixed-size local area, which reduces the flexibility in considering the interaction among any two users that might affect each other's trajectory. To solve such limitation, Social-GAN [3], a Generative Adversarial Network (GAN)-based trajectory predictor, extracts social interactions among all users in the system. Social-GAN stores the relative positions between all users of the social pool in a *pool vector*, in order to let each user make mobility decisions based on information of other users at the decoder side. However, Social-GAN weights each user trajectory's influence on the model identically. Sophie [5] solves this limitation by proposing another GAN-based social network that feeds all users of the scene to a social attention component, which aggregates various agents interactions and extracts users who have more influence on each other from the surrounding neighbors. Even though the social-pooling algorithms perform well in learning and predicting social interactions, they are computationally complex and resource-intensive because they feed trajectories of all users within a scene to the social predictor's encoders. In contrast to these works, INTRAFORCE generates a set of Social-Transformer trajectory predictors, each trained on the data of a different group of users with similar mobility. For an environment containing n mobile users, INTRAFORCE learns the social interactions among the $n_k \ll n$ users in each cluster c_k. In this way, the cluster's social-transformer is trained over n_k datasets instead of n, saving the computational resources needed for training over the $n - n_k$ datasets associated to users whose mobility features are irrelevant for cluster c_k.

Another shortcoming of the existing social trajectory predictors is that their neural architectures are designed by human experts, which is a time-consuming and error-prone process, and that the same neural network is applied to a wide range of mobility datasets with no neural architecture adaption. To mitigate such drawbacks, we propose to personalize the neural architecture design given each group of neighbor users' motion behaviors. In other fields than mobility prediction, various algorithms are proposed for Neural Architecture Search (NAS) [12]. INTRAFORCE formulates the neural hyper-parameter optimization as a RL problem. Most of the existing RL-based NAS approaches are proposed for image classification tasks, where it is simple to define appropriate search spaces due to researchers' prior knowledge and experience in manually-designed models. Hence, the absence of RL search mechanisms in less-explored fields is apparent [12]. RC-TL [8] suggests applying RL for individual trajectory prediction, where users are left isolated. In this direction, INTRAFORCE proposes and studies a RL-based NAS method in the field of social trajectory prediction. Table I compares the characteristics of our solution with those of existing state-of-the-art social-aware trajectory predictors.

III. INTRAFORCE

We define a scenario in which n users move in an urban environment that contains S base stations forming a cellular radio access network to the Internet. Each mobile user is provided with a wireless device that connects to the base station from which the strongest signal is received. When users move, the power signal received from base stations can vary and, thus, a *handover* procedure is performed to connect to a new base station. The timestamps of connection and disconnection to each base station are recorded. We assume that user u, at a time $t_u \in \mathbb{R}$, is located at coordinates $(x_u, y_u) \in \mathbb{R}^2$ and may be connected to a base station with ID $b_u \in \mathbb{N}$. We define the vector $p_u = (t_u, x_u, y_u, b_u) \in \mathbb{R}^3 \times \mathbb{N}$ as the *user information vector* that summarizes one data point about the user status. This generic formulation covers cases in which information about base stations or exact users positions may not be known to the system for privacy purposes. For user u, the system has recorded a total of m_u user information vectors. We now define the trajectory T_u as a set of user information vectors $T_u = \{p_u(1), \ldots, p_u(m_u)\}$, and $\Theta = \{T_1, \ldots, T_n\}$ as the set of all user trajectories. The goal of INTRAFORCE, is to predict the future trajectory for each user u, based on a user's past mobility data and other users' mutual influences.

INTRAFORCE operates through three modules: the *clustering module*, the *social-transformer module*, and the *architecture search module*, represented in Figure 1 and explained hereafter. Algorithm 1 describes INTRAFORCE's workflow, which includes the operation of the clustering module (lines 1 to 4), the architecture search module (lines 5 to 25), and the social-transformer module (lines 26 to 28).

The clustering module groups users with similar trajectories by applying a clustering algorithm (e.g., Birch, DBSCAN, K-Means, and Ward) considering the Longest Common Sub-Sequence (LCSS) distance [13] between their trajectories. The advantage of adopting LCSS is its ability to measure similarity between trajectories with different number of data points p_u.

The architecture search module uses ε-greedy Q-learning RL algorithm to select an optimal Transformer architecture for each cluster of users with similar trajectories. In RL, an

Algorithm 1: INTRAFORCE Workflow

```
Input: Set of trajectories $\Theta$
Output: A Social-TF $F_k$ for each cluster $c_k \in C$
// Compute similarity matrix $A = (a_{ij})$ and set $C$ of
   clusters containing similar trajectories
foreach $(T_i, T_j) \in \Theta^2$ // Clustering Module
do
    $a_{ij} \leftarrow$ LCSS$(T_i, T_j)$
$C \leftarrow$ Clustering$(A)$ ;
// Build a social transformer $F_k$, $\forall c_k \in C$
foreach $c_k \in C$ // Architecture Search Module
do
    Elect the representative user $r_k$ for cluster $c_k$;
    // Initialize RL agent $A_k$ to optimize the
       Transformer architecture using data of user $r_k$
    $A_k \leftarrow$ 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 $\varepsilon$ 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
            $\varepsilon \leftarrow \varepsilon - \varepsilon_0$;
        // $\varepsilon$-greedy strategy to select next TF
           architecture modification
        if 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 $a_v$
        $s' \leftarrow$ UpdateState$(s, a_v)$;
        // Train the transformer with data of the
           representative user $r_k$ for a few epochs $\theta_s$.
           Compute model error $w$ and reward $\rho_v$ of
           architecture modification (action $a$)
        $s'_* \leftarrow$ Train$(s', r_k, \theta_s)$;
        $w \leftarrow$ ComputeModelError$(s'_*, r_k)$;
        $\rho_v \leftarrow 1/w$;
        // Update state-action table
        $Q(s, a_v) \leftarrow (1 - \alpha) Q(s, a_v) +$
          $\alpha \left(\rho_v + \gamma \max_{a \in A(s')} Q(s', a)\right)$;
        // Stop architecture search if model error $w$
           (MSE for regression, Error Rate for
           classification) is below threshold $\eta$
        if $w \leq \eta$ then
            exit loop;
        $s \leftarrow s'$;
    // Selects the TF architecture $f_k$ with lowest
       model error
    $f_k \leftarrow \arg\max_{s \in S} Q(s, \cdot)$ // Social-TF Module
    // Initialize the Social-TF architecture for
       cluster $c_k$, with $n_k$ Encoder Stacks connected to
       $n_k$ Decoder Stacks through one social pool
    $F_k \leftarrow$ InitSocialTF$(f_k)$;
    // Train the Social-TF $F_k$ with data from all $n_k$
       users in $c_k$ for several epochs $\theta_l$
    $F^*_k \leftarrow$ Train$(F_k, c_k, \theta_s)$;
```

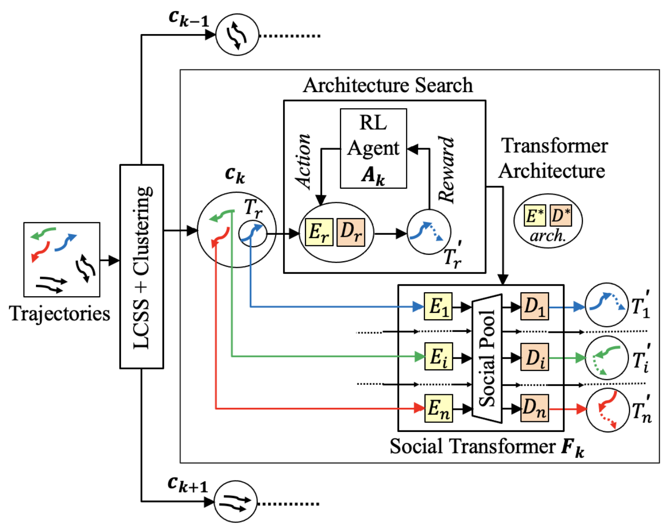


Fig. 1. INTRAFORCE Architecture.

agent takes an *action* that has an impact on an *environment*, then observes the environment's *state*, and finally receives a *reward* from the environment. INTRAFORCE uses RL to select the optimal transformer architecture from a finite and fixed space of admissible architectures (the *state space*). The state space is a subset of all the possible combinations of the values of the transformer's hyperparameters, namely the number, characteristics, and sequence of multi-head attention layers, normalization layers, feed forward layers, and dropout layers in the model. Each multi-head attention layer can have different numbers of heads h and dimension of key. Feed-forward layers can have different numbers of neurons. Normalization layers can have different epsilon values for the encoder. Dropout layers can have different dropout ratios. The state space can become considerably large depending on the range of admissible values for the hyperparameters (e.g., see the RL action part of Table II), so INTRAFORCE uses RL to search for the optimal architecture without performing an exhaustive grid search. After the agent has taken an action (i.e., selecting a candidate model architecture), the reward associated to the selected architecture is unknown, and therefore, must be measured by training the transformer generated by the RL agent with the cluster's representative users' data for a few epochs (*exploration phase*). At the end of the RL process, INTRAFORCE selects the transformer architecture with the highest accuracy among all those explored and completes its training until convergence (*exploitation phase*), over a larger number of epochs compared to the exploration epochs.

The social-transformer module spawns a neural network to predict the trajectory of each user within a specific cluster by tuning an Encoder Stack per cluster user, pooling cluster users' mutual information in a Social Pool, and predicting one trajectory per cluster user using a Decoder Stack. The Encoder Stack consists of a positional encoding layer and several encoder layers, in charge of converting a trajectory to an abstract representation, followed by the Decoder Stack to output a

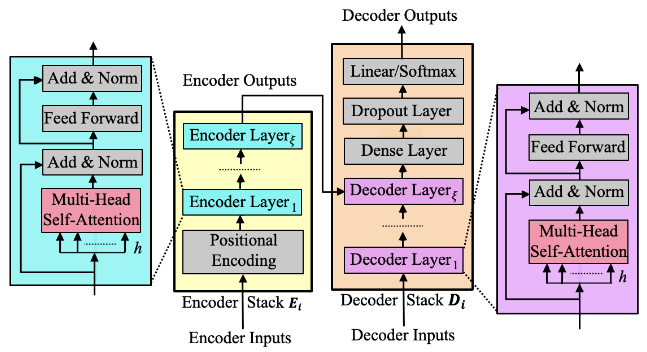


Fig. 2. Structure of a Transformer for the i-th user. The architecture contains an Encoder Stack E_i made of ξ Encoder Layers and a Decoder Stack D_i made of ξ Decoder Layers.

predicted trajectory from the learned abstract representations of the user mobility. The Decoder Stack consists of several decoding layers, dense layers, dropout layers, and an output layer (linear for regression and softmax for classification problems). Figure 2 represents the general structure of a Transformer highlighting its tuneable hyperparameters. Encoder and decoder layers consist of multi-head modules, normalization layers, feed forward layers, dropout layers, and two residual connections. The multi-head attention module contains a self-attention mechanism that accesses previous segments of input data and can differently weight the importance of each segment based on segments' pairwise similarities. The self-attention feature enables parallel training for Transformers, which considerably reduces training time compared to RNNs.

IV. Evaluation

A. Experimental Setup

We evaluate INTRAFORCE's performance on a *small-scale* and a *large-scale* mobility scenario, against a set of state-of-the-art mobility predictors. The two scenarios are named based on the size of the users' moving area. In the *small-scale* scenario, we assume that users move within an area that spans a few tens of meters. Therefore, the information about the base station to which users are connected does not characterize inter-user mobility interaction, nor contribute to improving the mobility prediction task. For this reason, we disregard any information about base stations and only consider small-scale user position coordinates in their trajectories. For the small-scale scenario, we use the information contained in the ETH [14] and UCY [15] public datasets containing video streams of real-world, small-scale pedestrian mobility in urban scenarios captured from bird-eye-view cameras, where users interact with each other and influence respective movements according to real social interactions at a microscopic scale. From both video datasets we extract the trajectories of 1536 pedestrians from five different urban scenarios with sampling rate of $0.4\,\mathrm{s}$ following the method proposed at [9]. The ETH dataset contains two urban scenarios named ETH and Hotel whereas, the UCY dataset contains three scenarios named Univ, Zara1, and Zara2. As a result, each trajectory T_u in the small-scale scenario will contain a sequence of up to 100 user information vectors $p_u = (t_u, x_u, y_u)$, where the timestamp granularity of any two consecutive vectors is $0.4\,\mathrm{s}$. We assume that the prediction models deployed in the small-scale scenario observe each user's trajectory for the past $T_{\text{obs}} = 8$ timestamps ($3.2\,\mathrm{s}$) and predicts the trajectory for the future $T_{\text{pred}} = 12$ frames timestamps ($4.8\,\mathrm{s}$). We compare INTRAFORCE performance with those of popular social trajectory-predicion models, namely: Social-LSTM [2], Social-GAN [3], Sophie [5], Social-BiGAT [16], Social-Ways [10], Social-STGCNN [11], PECNet [17], and STAR [6]. To measure the model performance for the regression task of predicting future location coordinates for each user, we use the *Average Displacement Error (ADE)*, which is the average squared Euclidean distance between all predicted points of a user trajectory and true locations. In this scenario we perform one experiment in which we run the whole INTRAFORCE system, including user clustering, RL search for Transformer architecture, and the social-transformer trajectory prediction with social pooling, and then collect the ADE measurements.

In the *large-scale* scenario, we assume that users can move within an area that spans several kilometers. In this case, the information about the microscopic user mobility (small-scale coordinates) over a short time-interval is not relevant to characterize mutual interactions between users. Hence, we consider only the information about which base station the user is connected to. For the large-scale scenario, we use a private cellular network management dataset provided by Orange telecommunication S.A., France [1]. This dataset contains the timestamps and the connected base station IDs for each of the 1.3 million users that move near a district of Paris between July and September 2019. For privacy reasons, the exact location coordinates of the subset of 131 identified base stations are inaccessible, and the user identities are anonymized. Trajectories in the large-scale scenario will contain sequences of a few thousands of information vectors $p_u = (t_u, b_u)$, where the timestamps of any two consecutive vectors are separated by a few minutes from each other. We assume that the prediction models deployed in the large-scale scenario observe each user's trajectory for the past $T_{\text{obs}} = 16$ timestamps and predicts the trajectory for the future $T_{\text{pred}} = 1$ timestamp. To measure the model performance for the classification task of predicting future base station IDs for each user, we define *accuracy* as the ratio between correctly predicted next locations and the total number of predictions made by the model. Formulating the large-scale scenario through classification instead of regression (predicting the next ID and not the exact location coordinates) is due to the unavailability of the base stations' coordinates. To measure the computational performance we measure the *build time* and the *model size* of the compared approaches, defined as the time to build and train the model and the number of its training parameters, respectively. We compare the performance of Reinforced Transformer (RL-TF) with those of other state-of-the-art trajectory predictors, namely RL-CNN, RL-LSTM, HO-LSTM, GS-LSTM, RF, and J48. The first four baselines are neural-network-based predictors, while the last two, Random Forest (RF) and J48 Decision Tree, are

TABLE II
EXPERIMENTAL 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 $v_{\max}$	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

non-neural predictors, which means that they do not require architecture search. RL-LSTM uses Reinforcement Learning to search for an optimal LSTM architecture, while HO-LSTM and GS-LSTM use Hyperopt (HO), an AutoML hyperparameter optimizer, and Grid Search (GS) for the same purpose. In the large-scale scenario, we perform two experiments. The first experiment compares the performance of a RL-TF with the aforementioned predictors in terms of prediction accuracy and build time for the individual user trajectory prediction, disregarding users' social interaction. The goal is to demonstrate the outperformance of Transformers concerning other machine learning predictors, and the outperformance of RL regarding other hyperparameter optimization models. The second experiment quantifies the impact of user clustering, cluster size, and the social-transformer model on accuracy, build time, and model size.

Table II shows the parameters for the Transformer and RL agent training. The RL Agent Actions section of Table II shows the features of the search space for the hyperparameters that defines the Transformer's architecture, where each row corresponds to one of the RL potential actions. We used Keras and TensorFlow to implement the transformer model, the RL agent used through INTRAFORCE, and the other evaluated trajectory prediction methods.

B. Results

Table III shows the results of the small-scale experiment (ETH+UCY datasets), in which INTRAFORCE achieves the

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

Work	ETH	Hotel	Univ	Zara1	Zara2	**Mean**
Social-LSTM [2]	1.09	0.79	0.67	0.47	0.56	**0.72**
Social-GAN [3]	0.81	0.72	0.60	0.34	0.42	**0.58**
SoPhie [5]	0.70	0.76	0.54	0.30	0.38	**0.54**
Social-BiGAT [16]	0.69	0.49	0.55	0.30	0.36	**0.48**
Social-Ways [10]	0.39	0.39	0.55	0.44	0.51	**0.46**
Social-STGCNN [11]	0.64	0.49	0.44	0.34	0.30	**0.44**
PECNet [17]	0.54	0.18	0.35	0.22	0.17	**0.29**
STAR [6]	0.36	0.17	0.31	0.26	0.22	**0.26**
INTRAFORCE	**0.31**	**0.24**	**0.22**	**0.14**	**0.23**	**0.22**

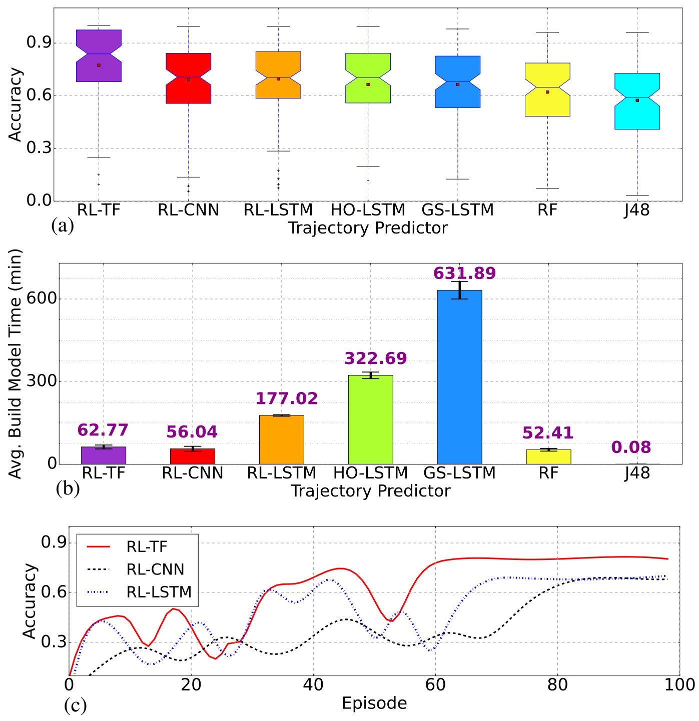
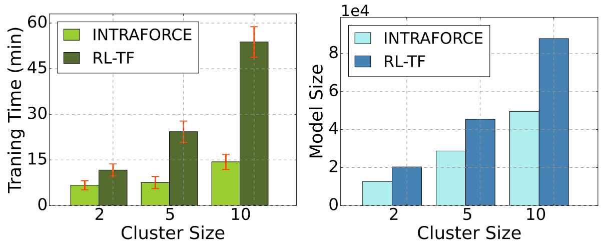


Fig. 3. Accuracy (a) and build time (b) of different trajectory predictors (three neural predictors whose architectures are optimized by different NAS mechanisms and two none-neural predictors) trained on an individual user's data for the large-scale scenario (Orange dataset). The evaluations are performed over 100 random users' data. Accuracy convergence (c) of the RL-designed predictors during the model building (neural architecture exploration and exploitation) phase.



(a) Average training time (b) Number of training parameters

Fig. 4. Average build time and model size of individual (RL-TF) and social (INTRAFORCE) trajectory prediction models, evaluated for different cluster sizes in the large-scale scenario (Orange dataset). The user clustering process within the social INTRAFORCE model is performed over the same set of 100 random users used in the individual RL-TF model.

lowest ADE (0.22) compared to several state-of-the-art social trajectory predictors.

The results of the large-scale experiment (Orange dataset) show that using RL to search the Transformer's neural architecture leads to a higher prediction accuracy and a faster build model time compared to other trajectory prediction approaches. Furthermore, training one single social-transformer with a cluster's users data reduces the average training time and model size compared to training one separate model for each individual user.

Figure 3a shows that RL-TF achieves the highest mean accuracy (77%), which is 10% higher than the other reinforced models (RL-LSTM and RL-CNN) and almost 20% higher than non-neural models (RF and J48). We note that achieving higher prediction accuracy over the Orange dataset is limited by the restricted dataset size (63 days) and the huge diversity in users' data sample distributions. The accomplished accuracy of 77% is the average accuracy over 100 random users with acutely variable data quality and periodicity. Figure 3b shows that RL-TF requires slightly above one hour of build time, which is similar to RL-CNN and RF. Although Transformers have larger architectures than CNNs and LSTMs, the Transformers' attention mechanism considerably reduces the build time. The RL-TF build time is similar to that of RF, which does not require architecture search but only training. RL-LSTM and HO-LSTM require long build times due to the sequential nature of LSTMs, and the extensive training of HO exploration. Figure 3c shows that the RL agent converges to a higher prediction accuracy in a shorter time with Transformer neural network, compared to LSTM and CNN.

Figure 4 shows the build time and the total model size needed by one social model for a cluster of users is lower than training an independent model for each user. As the number of users in a cluster increases, the difference in terms of build time and model size between training a separate model for each individual user (RL-TF) and training a single social-model per cluster of users (INTRAFORCE) dramatically increases. Moreover, we note that as the number of users in a cluster increases, social predictors achieve up to 5% higher accuracy, from 0.73 to 0.78, compared to individual predictors.

V. Conclusions

We presented INTRAFORCE, a system to build a trajectory predictor that learns the social interaction within clusters of similar mobile users. INTRAFORCE uses Reinforcement Learning to build a Social-Transformer architecture based on the intra-cluster user mobility features. We evaluate INTRAFORCE on small and large scale scenarios, based on the ETH+UCY and the Orange datasets, respectively. In the small-scale scenario, INTRAFORCE achieves an ADE of 0.22, which corresponds to a lower positioning error compared to several state-of-the-art models. In the large-scale scenario, we show that Reinforced Transformers outperform LSTM- and CNN-based predictors by achieving up to $+10\%$ accuracy and up to -70% training time, and outperforms non-neural models based on RF and J48 of up to $+20\%$ accuracy. Our experiments show that increasing the number of users in a cluster leads to slightly higher accuracy, while considerably decreasing the time needed to build and train the trajectory predictors, as well as the number of training parameters.

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] 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.

[2] 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.

[3] 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.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] 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.

[6] 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.

[7] 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.

[8] N. Emami, L. Pacheco, A. Di Maio, and T. Braun, "Rc-tl: Reinforcement convolutional transfer learning for large-scale trajectory prediction," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2022.

[9] 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.

[10] 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.

[11] 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.

[12] T. Elsken, J. H. Metzen, F. Hutter, *et al.*, "Neural architecture search: A survey.," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[13] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM (JACM)*, vol. 24, no. 4, pp. 664–675, 1977.

[14] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th international conference on computer vision*, pp. 261–268, IEEE, 2009.

[15] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[16] 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.

[17] 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.