Learning to predict human behaviour in crowded scenes

Alexandre Alahi, Vignesh Ramanathan, Kratarth Goel, Alexandre Robicquet, Amir Abbas Sadeghian, Li Fei-Fei, Silvio Savarese
Stanford University

1. Introduction

Humans are much more predictable in their transit patterns than we expect. In the presence of sufficient observations, it has been shown that our mobility is highly predictable even at a city-scale level [1]. The location of a person at any given time can be predicted with an average accuracy of 93% supposing 3 km$^2$ of uncertainty. How about at finer resolutions such as in shopping malls, in airports, or within train terminals for safety or resource optimization? What are the relevant cues to best predict human behavior within a margin of few centimeters?

Recently, Kitani et al. [2] showed that scene semantics provide strong cues for forecasting pedestrians’ trajectories. Helbing et al. [3, 4] also showed that our mobility is influenced by our neighbors, either consciously, e.g., by relatives or friends, or even unconsciously, e.g., by following an individual to facilitate navigation. More broadly, when humans walk in a crowded public space such as a train terminal, mall, or city centers, they obey a large number of (unwritten) common sense rules and comply with social conventions. For instance, as they consider where to move next, they respect personal space and yield right-of-way. The ability to model these rules and use them to understand and predict human motion in complex real world environments is extremely valuable for a wide range of applications - from the deployment of socially-aware robots [5] to the design of intelligent tracking systems [6] in smart environments.

In this chapter, we present two families of methods to forecast human trajectories in crowded environments. The first one is based on the popular Social Forces model [3] where the causalities behind human navigation is hand-designed by a set of functions that have been carefully chosen based on our understanding of physics underlying social behaviour. The second method is a fully data-driven approach based on Recurrent Neural Networks [7] that does not impose any hand-designed functions or explicit mobility based constraints.

The causality behind human mobility is an interplay between both observable and non-observable cues (e.g., intentions). Humans have the innate ability to “read” one another. When they need to avoid each other, there is an implicit cooperation on where to move next. They have the ability to get along well with each other by preserving a personal distance. These capabilities are often referred to as Social Intelligence [8]. Any forecasting method needs to infer the same behaviors to develop socially-aware intelligent systems. This requires
understanding the complex and often subtle interactions that take place between people in crowded spaces.

In the reminder of this chapter, after presenting relevant works in forecasting human behavior (while sharing more details on the popular Social Forces model [3]), we present a novel characterization of humans that describes the “social sensitivity” at which two humans interact. It captures both the preferred distance an individual wants to preserve with respect to her surrounding as well as the necessity to avoid collision. Low values for the social sensitivity feature implies that individual motion is not affected by other interacting neighbors. High values for the social sensitivity feature means that individual navigation is highly dependent on the position of other people. This characterization allows to define the “navigation style” humans follow while interacting with their surrounding. We obtain different classes of navigation styles by clustering trajectory samples in the social sensitivity space (see Figure 2 for examples). This allows to increase the flexibility in characterizing various modalities of interactions - for instance, some pedestrians who are rushed may appear more aggressive whereas others might exhibit a milder behavior because they are just enjoying their walk. Navigation style classes are used to select the appropriate set of parameters for the Social Forces model to improve prediction of human trajectories.

The ability to model social sensitivity is a key step towards learning common sense conventions based on social etiquette for enhancing forecasting tasks. However, this approach still depends on hand-crafted functions to model “interactions” for specific settings rather than inferring them in a data-driven fashion. This results in favoring models that capture simple interactions (e.g. repulsion/attractions) and might fail to generalize for more complex crowded settings. It also focuses on modeling interactions among people in close proximity to each other (to avoid immediate collisions). It does not anticipate interactions that could occur in the more distant future. Consequently, we end the chapter by presenting a data-driven architecture for predicting human trajectories in the future. Inspired by the success of Long-Short Term Memory networks (LSTM) for different sequence prediction tasks such as handwriting [7] and speech [9] generation, we extend them for human trajectory prediction as well. While LSTMs have the ability to learn and reproduce long sequences, they do not capture dependencies between multiple correlated sequences. We address this issue through a novel architecture which connects the LSTMs corresponding to one by one sequences (see Figure 1). In particular, we introduce a “Social” pooling layer which allows the LSTMs of spatially proximal sequences to share their hidden-states with each other. This architecture, which we refer to as the “Social-LSTM”, can automatically learn typical interactions that take place among trajectories which coincide in time. This model leverages existing human trajectory datasets without the need for any additional annotations to learn common sense rules and conventions that humans observe in social spaces. We conclude the chapter by demonstrating that the Social-LSTM is capable of predicting trajectories of pedestrians much more accurately than state-of-the-art methods on two publicly available datasets: ETH [10], and UCY [11]. We also analyze the trajectory patterns generated by our model to understand the social constraints learned from the trajectory datasets.
The goal of this chapter is to predict the motion dynamics in crowded scenes - This is, however, a challenging task as the motion of each person is typically affected by their neighbors. After presenting relevant methods to solve the forecasting task, we describe in Section 4.1 a new model which we call "Social" LSTM (Social-LSTM) which can jointly predict the paths of all the people in a scene by taking into account the common sense rules and social conventions that humans typically utilize as they navigate in shared environments. The predicted distribution of their future trajectories is shown in the heat-map.

2. Related work

Methods to forecast human navigation can be grouped into two categories: the ones modeling human-human interactions, and the ones focusing on human-space interactions. We briefly present an overview of past works for both approaches. We also discuss relevant Recurrent Neural Network (RNN) models for sequence prediction tasks.

Human-human interactions. Pioneering work from Helbing and Molnar [3] presented a pedestrian motion model with attractive and repulsive forces referred to as the Social Force model. This has been shown to achieve competitive results even on modern pedestrian datasets [11, 10]. This method was later extended to robotics [5] and activity understanding [6, 12, 13, 14, 15, 16, 17].

Similar approaches have been used to model human-human interactions with strong priors for the model. Treuille et. al. [18] use continuum dynamics, Antonini et. al. [19] propose a Discrete Choice framework and Wang et. al. [20], Tay et. al. [21] use Gaussian processes. Such functions have also been used to study stationary groups [22, 23]. These works target smooth motion paths and do not handle the problems associated with discretization.

Another line of work uses well-engineered features and attributes to improve tracking and forecasting. Alahi et. al. [24] presented a social affinity feature by learning from human trajectories in crowd their relative positions, while Yu et. al. [22] proposed the use of
human-attributes to improve forecasting in dense crowds. They also use an agent-based model similar to [25]. Rodriguez et al. [26] analyze videos with high-density crowds to track and count people.

Most of these models provide hand-crafted energy potentials based on relative distances and rules for specific scenes. In contrast, we propose a method to learn human-human interactions in a more generic data-driven fashion.

**Activity forecasting.** Activity forecasting models try to predict the motion and/or action to be carried out by people in a video. A large body of work learns motion patterns through clustering trajectories [27, 28, 29, 30]. More approaches can be found in [31, 32, 33, 34, 35, 36]. Kitani et al. in [37] use Inverse Reinforcement Learning to predict human paths in static scenes. They infer walkable paths in a scene by modeling human-space interactions. Walker et al. in [38] predict the behavior of generic agents (e.g., a vehicle) in a visual scene given a large collection of videos. Ziebart et al. [39, 40] presented a planning based approach.

Turek et al. [41, 42] used a similar idea to identify the functional map of a scene. Other approaches like [43, 44, 45, 46] showed the use of scene semantics to predict goals and paths for human navigation. Scene semantics has also been used to predict multiple object dynamics [47, 46, 33, 48]. These works are mostly restricted to the use of static scene information to predict human motion or activity. In our work, we focus on modeling dynamic crowd interactions for path prediction.

More recent works have also attempted to predict future human actions. In particular, Ryoo et al. [49, 50, 51, 52, 53, 54] forecast actions in streaming videos. More relevant to our work, is the idea of using a RNN model to predict future events in videos [55, 56, 57, 58, 59]. Along similar lines, we predict future trajectories in scenes.

**RNN models for sequence prediction.** Recently Recurrent Neural Networks (RNN) and their variants including Long Short Term Memory (LSTM) [60] and Gated Recurrent Units [61] have proven to be very successful for sequence prediction tasks: speech recognition [9, 62, 63], caption generation [64, 65, 66, 67, 68], machine translation [69], image/video classification [70, 71, 72, 73], human dynamics [74] to name a few. RNN models have also proven to be effective for tasks with densely connected data such as semantic segmentation [75], scene parsing [76] and even as an alternative to Convolutional Neural Networks [77]. These works show that RNN models are capable of learning the dependencies between spatially correlated data such as image pixels. This motivates us to extend the sequence generation model from Graves et al. [7] to our setting. In particular, Graves et al. [7] predict isolated handwriting sequences; while in our work we jointly predict multiple correlated sequences corresponding to human trajectories.

3. Forecasting with Social Forces model

We first present the popular Social Forces model [12] to forecast human trajectory. In this section, we introduce the basic theory behind the model and how to adapt it to multi-
class settings. The model is also our inspiration for our social sensitivity feature described in Sec. 3.2.

3.1. Basic theory

The Social Forces model is commonly used to predict trajectories of pedestrians in a crowded environment. In this model pedestrians are viewed as decision making agents who consider a multitude of personal, social and environmental factors to decide where to go next. Each agent makes a decision on the velocity \( v_i^{(t+\Delta t)} \). At each time step \( t \), the object \( i \) is defined by a state variable \( s_i(t) = \{ p_i^{(t)}, v_i^{(t)}, u_i^{(t)}, g_i^{(t)}, A_i^{(t)} \} \), where \( p_i^{(t)} \) is the position, \( v_i^{(t)} \) the velocity, \( u_i^{(t)} \) the preferred speed (according to the class and the past velocities), \( g_i^{(t)} \) the chosen destination (or goal) and \( A_i^{(t)} \) is the set of objects in the same social group (including \( i \)). Similar to [12], the energy function, \( E_\Theta \), associated to every single agent is defined as:

\[
E_\Theta(v; s_i, s_{-i}) = \lambda_0 E_{\text{damping}}(v; s_i) + \\
\lambda_1 E_{\text{speed}}(v; s_i) + \\
\lambda_2 E_{\text{direction}}(v; s_i) + \\
\lambda_3 E_{\text{attraction}}(v; s_i, s_{A_i}) + \\
\lambda_4 E_{\text{group}}(v; s_i, s_{A_i}) + \\
E_{\text{collision}}(v; s_i, s_{-i} | \sigma_d, \sigma_w, \beta)
\]

where \( \Theta = \{ \lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \sigma_d, \sigma_w, \beta \} \) is the model parameters, \( s_{A_i} \) is the set of state variables of the agent in \( i \)'s social group \( A_i \). \( s_{-i} \) set of states of other agents except \( i \). The parameter \( \lambda_i \) are then weights to balance the importance of each of those energies (\( E \)). More details on the definition of each of the energy can be found in [12]. In our work, we use the collision energy to define our social sensitivity feature in Sec. 3.2. Consequently, we will describe the parameters \( \{ \sigma_d, \sigma_w, \beta \} \) in Sec. 3.2.

Previous works [12, 13, 5] only use one set of parameters for the whole crowd. This approximation implied that everyone would maintain the same safety distance or would grant the exact same weight to each energy function. We can easily see that someone in a hurry would be more likely to bump into or navigate close to others in order to navigate faster, granting more weight to his damping energy in order to go as straight as possible to his destination.

3.2. Modeling Social Sensitivity

We claim that modeling human trajectory with a single navigation style is not suitable for capturing the variety of social behaviors that targets exhibit when interacting in complex scenes. We believe that conditioning such models on navigation style (i.e., the way targets avoid each other) is a better idea and propose a characterization (feature) which we call social sensitivity. Given this characterization, we hence assign a navigation style to each target to better forecast its trajectory and improve tracking.
**Social Sensitivity feature.** Inspired by the Social Forces model (SF) [12], we model targets’ interactions with an energy potential $E_{ss}$. A high potential means that the target is highly sensitive to others. We define $E_{ss}$ as follows:

At each time step $t$, the target $i$ is defined by a state variable $s_i(t) = \{p_i(t), v_i(t)\}$, where $p_i(t)$ is the position, and $v_i(t)$ the velocity. The energy potential encoding the social sensitivity is computed as follows:

$$E_{ss}(v_i(t), s_i, s_{-i}|\sigma_d, \sigma_w, \beta) = \sum_{j \neq i} w(s_i, s_j) \exp \left( -\frac{d^2(v, s_i, s_j)}{2\sigma^2_d} \right),$$

with $w(s_i, s_j)$ defined as:

$$w(s_i, s_j) = \exp \left( -\frac{||\Delta p_{ij}||}{2\sigma_w} \right) \cdot \left( \frac{1}{2} \left( 1 - \frac{\Delta p_{ij}}{||\Delta p_{ij}||} \cdot v_i \right) \right)^\beta,$$

and

$$d^2(v, s_i, s_j) = ||\Delta p_{ij} - \Delta p_{ij}(v - v_j)||_{v - v_j|^2}^2.$$  

The energy $E_{ss}$ is modeled as a product of Gaussians where the variances $\sigma_{w,d}$ represent the distances at which other targets will influence each other. For instance, if two targets $i, j$ are close to each other ($\Delta p_{ij}$ is small), $E_{ss}$ will be large when $\sigma_{w,d}$ are small.

We define the parameter $\Theta_{ss} = \{\sigma_d, \sigma_w, \beta\}$ as the social sensitivity feature and interpret its dimension as follows:

- $\sigma_d$ is the preferred distance a target maintains to avoid collision,
- $\sigma_w$ is the distance at which a target reacts to prevent a collision (distance at which (s)he starts deviating from its linear trajectory),
- and $\beta$ controls the peakiness of the weighting function.

In other words, the parameters $\{\sigma_d, \sigma_w, \beta\}$ aim at describing how targets avoid each others - i.e., their social sensitivity. We now present how we infer the parameters $\Theta_{ss}$ at training and testing time.

**Training.** At training time, since we observe all targets’ velocities, $V^{\text{train}}$, we could learn a unique set of parameters, i.e., a single value for social sensitivity, that minimizes the energy potential as follows (similarly to what previous methods do [12, 13, 15, 16]):

$$\{\sigma_d, \sigma_w, \beta\} = \arg \min_{\{\sigma_d, \sigma_w, \beta\}} \left( \sum_{i=1}^{T-1} E_{ss}(v_i^{\text{train}}, s_i, s_{-i}|\sigma_d, \sigma_w, \beta) \right),$$

(10)
where $T$ is the number of targets in the training data. This minimization is operated with an interior-point method and is set with the following constraint on $\sigma_d$: $\sigma_d > 0.1$ (it specifies that every target can’t have a “vital space” smaller than 10cm).

However, as mentioned previously, we claim that learning a unique set of parameters is not suitable when one needs to deal with complex multi-class target scenarios whereby targets can have different social sensitivity. To validate this claim, we visualize (Figure 2) each target in a social sensitivity space where the x-axis is the $\sigma_d$ values and the y-axis is the $\sigma_w$ ones. This plot is generated using training images from our dataset (see Section 5 for more details). We did not plot the third parameter $\beta$ since it does not change much across targets. Even though our approach can handle an arbitrary number of classes, we cluster the points into only four clusters for the ease of illustration. Each cluster corresponds to what we define as a “navigation style”. A navigation style describes the sensitivity of a target to its surrounding. We illustrate on the sides of Figure 2 how targets follow different strategies in avoiding each other as different navigation styles are used.

Thanks to the above analysis of the social sensitivity space, at training, we solve Equation 10 for each target - without the summation over all targets - to get its social sensitivity feature. We then cluster the points with K-mean clustering to have $N$ number of clusters. Each cluster represents a navigation style.

**Testing.** At test time, we observe the targets until time $t$, and want to assign a navigation style.

In the presence of other targets, we solve Equation 11 for each specific target $i$ at time $t$:

$$\{\sigma_d(i), \sigma_w(i), \beta(i)\} = \arg\min_{\{\sigma_d(i), \sigma_w(i), \beta(i)\}} (E_{ss}(v_i^t, s_i, s_{-i}|\sigma_d(i), \sigma_w(i), \beta(i)))$$

(11)

We obtain the social sensitivity feature $\Theta_{ss}(i) = \{\sigma_d(i), \sigma_w(i), \beta(i)\}$ for each target $i$. Given the clusters found at training, we assign each $\Theta_{ss}(i)$ to its corresponding cluster, i.e., navigation style.

In the absence of interactions, a target takes either a “neutral” navigation style (when entering a scene) or inherit the last inferred class from the previous interaction. The “neutral” navigation style is the most popular one (in green in Figure 2). In figure 3, we show that when the target is surrounded by other targets, its class changes with respect to its social sensitivity.

### 3.3. Forecasting with Social Sensitivity

Thanks to our proposed social sensitivity feature, we have more flexibility in modeling target interactions to forecast future trajectories. In the remaining of this section, we present the details behind our forecasting model driven by social sensitivity.
Problem formulation. Given the observed trajectories of several targets at time $t$, we aim to forecast their future positions over the next $N$ time frames (where $N$ is in seconds).

We adapt the Social Forces model \cite{12} from single class to multiple classes. Each target makes a decision on its velocity $v_{t+1}^i$. The energy function, $E_{\Theta}$, associated to every single target is defined as:

\[ E_{\Theta}(v_{t+1}^i; s_i, s_{-i}) = \lambda_0(c) E_{damp}(v_{t+1}^i; s_i) + \lambda_1(c) E_{speed}(v_{t+1}^i; s_i) \\
+ \lambda_2(c) E_{dir}(v_{t+1}^i; s_i) + \lambda_3(c) E_{att}(v_{t+1}^i; s_i) + \lambda_4(c) E_{group}(v_{t+1}^i; s_i, s_A) \\
+ E_{ss}(v_{t+1}^i; s_i, s_{-1}|\sigma_d(v^t), \sigma_w(v^t), \beta) \]  

(12)

where $\Theta = \{\lambda_0(c), \lambda_1(c), \lambda_2(c), \lambda_3(c), \lambda_4(c), \sigma_d(v^t), \sigma_w(v^t), \beta\}$ and $c$ is the navigation class. More details on the definition of each of the energy terms can be found in \cite{12}.

In our work, we propose to compute $\sigma_d$, and $\sigma_w$ directly from the observed velocity $v^t$ using Equation 11. Both distances $\sigma_d$, and $\sigma_w$ will then be used to identify the navigation class $c$. For each class $c$, the parameter $\Theta$ can be learned from training data by minimizing the energy in Equation 12. We can visualize the impact of the navigation style on the prediction. In figure 4, we show the predicted trajectories when several navigation styles are used to perform the forecasting. This shows the need to assign targets into specific classes.
Figure 3: Illustration of the class assignment for each target. We follow the same color coding as Figure 2 to represent the different navigation styles. Note that for a given target, its class changes across time regardless of its physical class (i.e., whether it is a pedestrian, bike, etc.). When the target is surrounded by other targets, its class changes with respect to its social sensitivity. In this scene, first we can observe a cyclist (shown as label 1 in the images) belonging to a black cluster, i.e., being aggressive in his moves, then belonging to some milder clusters (purple and green). We also can see the evolution of a group of pedestrians (shown as labels 2,3) in the images), initially “mild” (green at $T = 1$), who become red at time $T = 3$ when they accelerate to overtake another group.

4. Forecasting with Recurrent Neural Network

Humans moving in crowded scenes adapt their motion based on the behaviour of other people in their vicinity. For instance, a person could completely alter his/her path or stop momentarily to accommodate a group of people moving towards him. Such deviation in trajectory cannot be predicted by observing the person in isolation. Neither, can it be predicted with simple ”repulsion” or ”attraction” functions (presented in the previous section).

This motivates us to build a model which can account for the behavior of other people within a large neighborhood, while predicting a person’s path. In this section, we describe our pooling based LSTM model (Fig. 5) which jointly predicts the trajectories of all the people in a scene. We refer to this as the “Social” LSTM model.

Problem formulation. We assume that each scene is first preprocessed to obtain the spatial coordinates of the all people at different time-instants. Previous work follow this convention as well [5, 24]. At any time-instant $t$, the $i^{th}$ person in the scene is represented by his/her xy-coordinates $(x^i_t, y^i_t)$. We observe the positions of all the people from time 1 to $T_{obs}$, and predict their positions for time instants $T_{obs+1}$ to $T_{pred}$. This task can also be viewed as a sequence generation problem [7], where the input sequence corresponds to the observed positions of a person and we are interested in generating an output sequence denoting his/her future positions at different time-instants.
Figure 4: We show the predicted trajectory of a given target (red circle) in which four different navigation styles are used to perform the prediction. The corresponding predicted trajectories are overlaid over each other and shown with different color codes (the same as those used for depicting the clusters in figure 2). The ground truth is represented in blue. Predicted trajectories are shown for 6 subsequent frames indicated by $T = 1, \ldots, 6$ respectively. Interestingly, when the target is far away from other targets (no interactions are taking place) the predicted trajectories are very similar to each other (they almost overlap and show a linear trajectory). However, when the red target gets closer to other targets (e.g. the ones indicated in yellow), the predicted trajectories start showing different behaviors depending on the navigation style: a conservative navigation style activates trajectories’ prediction that keep large distances to the yellow targets in order to avoid them (green trajectory) whereas an aggressive navigation style activates trajectories’ prediction that are not too distant from the yellow targets (red trajectory). Notice that our approach is capable to automatically associate the target to one of the 4 clusters based on the characteristics in the social sensitivity space that have been observed until present. In this example, our approach selects the red trajectory which is the closest to the ground truth’s predicted trajectory (in blue).

4.1. Social LSTM

Every person has a different motion pattern: they move with different velocities, acceleration and have different gaits. We need a model which can understand and learn such person-specific motion properties from a limited set of initial observations corresponding to the person.

Long Short-Term Memory (LSTM) networks have been shown to successfully learn and generalize the properties of isolated sequences like handwriting [7] and speech [9]. Inspired by this, we develop a LSTM based model for our trajectory prediction problem as well. In particular, we have one LSTM for each person in a scene. This LSTM learns the state of
Figure 5: Overview of our Social-LSTM method. We use a separate LSTM network for each trajectory in a scene. The LSTMs are then connected to each other through a Social pooling (S-pooling) layer. Unlike the traditional LSTM, this pooling layer allows spatially proximal LSTMs to share information with each other. The variables in the figure are explained in Eq. 14. The bottom row shows the S-pooling for one person in the scene. The hidden-states of all LSTMs within a certain radius are pooled together and used as an input at the next time-step.

the person and predicts their future positions as shown in Fig. 5. The LSTM weights are shared across all the sequences.

However, the naive use of one LSTM model per person does not capture the interaction of people in a neighborhood. The vanilla LSTM is agnostic to the behaviour of other sequences. We address this limitation by connecting neighboring LSTMs through a new pooling strategy visualized in Fig. 5 and Fig. 6.

Social pooling of hidden states. Individuals adjust their paths by implicitly reasoning about the motion of neighboring people. These neighbors in-turn are influenced by others in their immediate surroundings and could alter their behaviour over time. We expect the hidden states of an LSTM to capture these time varying motion-properties. In order to jointly reason across multiple people, we share the states between neighboring LSTMS. This introduces a new challenge: every person has a different number of neighbors and in very dense crowds [24], this number could be prohibitively high.

The “neighborhood” of a person changes dynamically and the LSTM predicting future
position should be able to process this time-varying “neighborhood” state.

Hence, we need a compact representation which combines the information from all neighboring states. We handle this by introducing “Social” pooling layers as shown in Fig. 5. At every time-step, the LSTM cell receives pooled hidden-state information from the LSTM cells of neighbors. While pooling the information, we try to preserve the spatial information through grid based pooling as explained below.

The hidden state \( h^t_i \) of the LSTM at time \( t \) captures the latent representation of the \( i^{th} \) person in the scene at that instant. We share this representation with neighbors by building a “Social” hidden-state tensor \( H^t_i \). Given a hidden-state dimension \( D \), and neighborhood size \( N \), we construct a \( N \times N \times D \) tensor \( H^t_i \) for the \( i^{th} \) trajectory:

\[
H^t_i(m, n, :) = \sum_{j \in \mathcal{N}_i} 1_{mn}[x^j_t - x^i_t, y^j_t - y^i_t]h^j_{t-1},
\]

where \( h^j_{t-1} \) is the hidden state of the LSTM corresponding to the \( j^{th} \) person at \( t-1 \), \( 1_{mn}[x, y] \) is an indicator function to check if \((x, y)\) is in the \((m, n)\) cell of the grid, and \( \mathcal{N}_i \) is the set of neighbors corresponding to person \( i \). This pooling operation is visualized in Fig. 6.

We embed the pooled Social hidden-state tensor into a vector \( a^t_i \) and the co-ordinates into \( e^t_i \). These embeddings are concatenated and used as the input to the LSTM cell of the corresponding trajectory at time \( t \). This introduces the following recurrence:

\[
e^t_i = \phi(x^t_i, y^t_i; W_e) \\
a^t_i = \phi(H^t_i; W_a), \\
h^t_i = \text{LSTM}(h^t_{i-1}, e^t_i, a^t_i; W_l)
\]

where \( \phi(.) \) is an embedding function with ReLU non-linearity, \( W_e \) and \( W_a \) are embedding weights. The LSTM weights are denoted by \( W_l \).

**Position estimation.** The hidden-state at time \( t \) is used to predict the distribution of the trajectory position \((\hat{x}, \hat{y})_{t+1}^i\) at the next time-step \( t+1 \). Similar to Graves et al. [7], we assume a bivariate Gaussian distribution parametrized by the mean \( \mu_{t+1}^i = (\mu_x, \mu_y)_{t+1}^i \), standard deviation \( \sigma_{t+1}^i = (\sigma_x, \sigma_y)_{t+1}^i \) and correlation coefficient \( \rho_{t+1}^i \). These parameters are predicted by a linear layer with a \( 5 \times D \) weight matrix \( W_p \). The predicted coordinates \((\hat{x}^t_i, \hat{y}^t_i)\) at time \( t \) are given by

\[
(\hat{x}, \hat{y})_t^i \sim \mathcal{N}(\mu_t^i, \sigma_t^i, \rho_t^i)
\]

The parameters of the LSTM model are learned by minimizing the negative log-Likelihood
We train the model by minimizing this summation of loss \( L \) for all the trajectories in a training dataset. Note that our “Social” pooling layer does not introduce any additional parameters.

An important distinction from the traditional LSTM is that the hidden states of multiple LSTMs are coupled by our “Social” pooling layer and we jointly back-propagate through multiple LSTMs in a scene at every time-step. In other words, for a given snapshot in time, we evaluate the trajectory LSTMs along with the social pooling layer for all the trajectories in the scene. The sum of the loss from the predicted positions and true positions of these trajectories is jointly minimized through stochastic gradient descent.

**Occupancy map pooling.** The “Social” LSTM model can be used to pool any set of features from neighboring trajectories. As a simplification, we also experiment with a model which only pools the co-ordinates of the neighbors (referred to as O-LSTM in the experiments Sect. 5). This is a reduction of the original model and does not require joint back-propagation across all trajectories during training. This model can still learn to reposition a trajectory to avoid immediate collision with neighbors. However, in the absence of more information from neighboring people, this model would be unable to smoothly change paths to avoid future collisions.

For a person \( i \), we modify the definition of the tensor \( H^i_t \), as a \( N_o \times N_o \) matrix at time \( t \) centered at the person’s position, and call it the occupancy map \( O^i_t \). The positions of all the neighbors are pooled in this map. The \( m, n \) element of the map is simply given by:

\[
O^i_t(m, n) = \sum_{j \in N_i} \mathbf{1}_{mn}[x^i_t - x^j_t, y^i_t - y^j_t],
\]

where \( \mathbf{1}_{mn}[\cdot] \) is an indicator function as defined previously. This can also be viewed as a simplification of the social tensor in Eq. 13 where the hidden state vector is replaced by a constant value indicating the presence or absence of neighbors in the corresponding cell.

The vectorized occupancy map is used in place of \( H^i_t \) in Eq. 14 while learning this simpler model.

**Inference for path prediction.** During test time, we use the trained Social-LSTM models to predict the future position \((\hat{x}^i_t, \hat{y}^i_t)\) of the \( i^{th} \) person. From time \( T_{obs+1} \) to \( T_{pred} \), we use the predicted position \((\hat{x}^i_t, \hat{y}^i_t)\) from the previous Social-LSTM cell in place of the true coordinates

\[
\begin{align*}
\mu^i_t \sigma^i_t \rho^i_t &= W^p h^t, \\
L^i(W_e, W_l, W_p) &= - \sum_{t=T_{obs+1}}^{T_{pred}} \log P(x^i_t, y^i_t | \sigma^i_t, \mu^i_t, \rho^i_t), \\
L(W_e, W_l, W_p) &= \sum_i L^i(W_e, W_l, W_p).
\end{align*}
\]
Figure 6: We show the Social pooling for the person represented by a black-dot. We pool the hidden states of the neighbors (shown in yellow, blue and orange) within a certain spatial distance. The pooling partially preserves the spatial information of neighbors as shown in the last two steps.

$$(x_i^t, y_i^t)$$ in Eq. 14. The predicted positions are also used to replace the actual coordinates while constructing the Social hidden-state tensor $H_i^t$ in Eq. 13 or the occupancy map $O_i^t$ in Eq. 17.

4.2. Implementation details

We use an embedding dimension of 64 for the spatial coordinates before using them as input to the LSTM. We set the spatial pooling size $N_o$ to be 32 and use a 8x8 sum pooling window size without overlaps. We used a fixed hidden state dimension of 128 for all the LSTM models. Additionally, we also use an embedding layer with ReLU (rectified Linear Units) non-linearity on top of the pooled hidden-state features, before using them for calculating the hidden state tensor $H_i^t$. The hyper-parameters were chosen based on cross-validation on a synthetic dataset. This synthetic was generated using a simulation that implemented the social forces model. This synthetic data contained trajectories for hundreds of scenes with an average crowd density of 30 per frame. We used a learning rate of 0.003 and RMS-prop [78] for training the model. The Social-LSTM model was trained on a single GPU with a Theano [79] implementation.

5. Experiments

In this section, we present experiments on two publicly available human-trajectory datasets: ETH [10] and UCY [11]. The ETH dataset contains two scenes each with 750 different pedestrians and is split into two sets (ETH and Hotel). The UCY dataset contains two scenes with 786 people. This dataset has 3-components: ZARA-01, ZARA-02 and UCY. In total, we evaluate our model on 5 sets of data. These datasets represent real world crowded settings with thousands of non-linear trajectories. As shown in [10], these datasets also cover challenging group behaviours such as couples walking together, groups crossing each other and groups forming and dispersing in some scenes.

We report the prediction error with three different metrics. Similar to Pellegrini et al. [10] we use:

1. **Average displacement error** - The mean square error (MSE) over all estimated points of a trajectory and the true points. This was introduced in Pellegrini et al. [10].
2. **Final displacement error** - The distance between the predicted final destination and the true final destination at end of the prediction period $T_{pred}$. 

14
3. **Average non-linear displacement error** - This is the MSE at the non-linear regions of a trajectory. Since most errors in trajectory-prediction occur during non-linear turns arising from human-human interactions, we explicitly evaluate the errors around these regions. We set a heuristic threshold on the norm of the second derivative to identify non-linear regions.

In order to make full use of the datasets while training our models, we use a leave-one-out approach. We train and validate our model on 4 sets and test on the remaining set. We repeat this for all the 5 sets. We also use the same training and testing procedure for other baseline methods used for comparison.

During test time, we observe a trajectory for 3.2 seconds and predict their paths for the next 4.8 seconds. At a frame rate of 0.4, this corresponds to observing 8 frames and predicting for the next 12 frames. This is similar to the setting used by [10, 11]. In Tab. 1, we compare the performance of the following methods:

- **Linear model** (*Lin.*). We use an off-the-shelf Kalman filter to extrapolate trajectories with assumption of linear acceleration.

- **Collision avoidance** (*LTA*). We report the results of a simplified version of the Social Force [12] model which only uses the collision avoidance energy, commonly referred to as linear trajectory avoidance.

- **Social force** (*SF*). We use the implementation of the Social Force model from [12] where several factors such as group affinity and predicted destinations have been modeled.

- **Iterative Gaussian Process** (*IGP*). We use the implementation of the IGP from [80]. Unlike the other baselines, IGP also uses additional information about the final destination of a person.

- **Our multi-class Social Force** (*SF-mc*). The approach presented in Section 3.3.

- **Our Vanilla LSTM** (*LSTM*). This is a simplified setting of our model where we remove the “Social” pooling layers and treat all the trajectories to be independent of each other.

- **Our LSTM with occupancy maps** (*O-LSTM*). We show the performance of a simplified version of our model (presented in Sec. 4.1). As a reminder, the model only pools the coordinates of the neighbors at every time-instance.

- **Our Social LSTM**. The approach presented in Section 4.1.

The naive linear model produces high prediction errors, which are more pronounced around non-linear regions as seen from the average non-linear displacement error. The vanilla LSTM outperforms this linear baseline since it can extrapolate non-linear curves as shown in Graves et al. [7]. However, this simple LSTM is noticeably worse than the Social
Metric | Methods | Lin | LTA | SF [12] | IGP* [81] | SF-mc | LSTM | our O-LSTM | our Social-LSTM
--- | --- | --- | --- | --- | --- | --- | --- | --- | ---
**Avg. disp. error** | ETH [10] | 0.80 | 0.54 | 0.41 | **0.20** | 0.41 | 0.60 | 0.49 | 0.50
Hotel [10] | 0.95 | 0.46 | 0.56 | 0.46 | 0.54 | 0.28 | 0.49 | 0.50 | 0.28
Zara 1 [11] | 0.47 | 0.37 | 0.40 | 0.39 | 0.35 | 0.43 | **0.22** | 0.42 | 0.22
Zara 2 [11] | 0.45 | 0.40 | 0.40 | 0.41 | 0.39 | 0.51 | 0.28 | 0.25 | 0.27
UCY [11] | 0.57 | 0.51 | 0.48 | 0.61 | 0.45 | 0.52 | 0.35 | 0.37 | 0.27
Average | 0.53 | 0.44 | 0.39 | 0.47 | 0.37 | 0.44 | 0.28 | 0.27 | 0.27

**Avg. non-linear disp. error**
| ETH [10] | 0.95 | 0.70 | 0.49 | 0.39 | 0.46 | 0.28 | **0.24** | 0.25 | 0.25
Hotel [10] | 0.55 | 0.49 | 0.38 | 0.34 | 0.32 | 0.09 | **0.06** | 0.07 | 0.07
Zara 1 [11] | 0.56 | 0.39 | 0.41 | 0.34 | 0.41 | 0.24 | **0.13** | 0.13 | 0.13
Zara 2 [11] | 0.44 | 0.41 | 0.39 | 0.43 | 0.39 | 0.30 | 0.30 | **0.16** | 0.16
UCY [11] | 0.62 | 0.57 | 0.54 | 0.62 | 0.51 | 0.31 | 0.20 | **0.16** | 0.16
Average | 0.62 | 0.51 | 0.44 | 0.46 | 0.42 | 0.24 | 0.24 | **0.17** | 0.17

**Final disp. error**
| ETH [10] | 1.31 | 0.77 | 0.59 | **0.43** | 0.59 | 1.31 | 1.06 | 1.07 | 1.07
Hotel [10] | 0.55 | 0.64 | 0.37 | 0.37 | 0.37 | 0.33 | **0.20** | 0.23 | 0.23
Zara 1 [11] | 0.89 | 0.66 | 0.60 | 0.39 | 0.60 | 0.93 | **0.36** | 0.48 | 0.48
Zara 2 [11] | 0.91 | 0.72 | 0.68 | 0.42 | 0.67 | 1.09 | 0.58 | **0.50** | 0.50
UCY [11] | 1.13 | 0.95 | 0.78 | 1.32 | 0.76 | 1.25 | 0.90 | 0.90 | **0.77**
Average | 0.97 | 0.74 | **0.60** | 0.69 | 0.60 | 0.98 | 0.64 | 0.61 | 0.61

Table 1: Quantitative results of all the methods on all the datasets. We present the performance metrics as follows: First 6 rows are the Average displacement error, row 7 to 12 are the Average displacement error for non-linear regions, and the final 6 rows are the Final displacement error. All methods forecast trajectories for a fixed period of 4.8 seconds. (*) Note that IGP uses the intended ground truth destination of a person during test time unlike other methods.

Force and IGP models which explicitly model human-human interactions. This shows the need to account for such interactions.

Our presented SF-mc performs the same as the single class Social Forces model in ETH dataset, and outperforms other methods in UCY datasets. This result can be justified by the fact that the UCY dataset is considerably more crowded, with more collisions, and therefore presenting different types of behaviors. Non-linear behaviors such as people stopping and talking to each other, walking faster, or turning around each others are more common in UCY than in ETH. The SF-mc is able to infer these navigation patterns hence better predict the trajectories of pedestrians. We also report the performance of the IGP model for completeness. While IGP performs better on the less crowded dataset, it does not do well on the crowded ones. Notice that IGP uses the destination and time of arrival as additional inputs (which other methods don’t use).

Our Social pooling based LSTM and O-LSTM outperform the heavily engineered Social Force and IGP models in almost all datasets. In particular, the error reduction is more significant in the case of the UCY datasets as compared to ETH. This can be explained by the different crowd densities in the two datasets: UCY contains more crowded regions with a total of 32K non-linearities as opposed to the more sparsely populated ETH scenes with only 15K non-linear regions.

In the more crowded UCY scenes, the deviation from linear paths is more dominated by human-human interactions. Hence, our model which captures neighborhood interactions achieves a higher gain in UCY datasets. The pedestrians’ intention to reach a certain destination plays a more dominant role in the ETH datasets. Consequently, the IGP model which knows the true final destination during testing achieves lower errors in parts of this
In the case of ETH, we also observe that the occupancy and Social LSTM errors are at par with each other and in general better than the Social force model. Again, our Social-LSTM outperforms O-LSTM in the more crowded UCY datasets. This shows the advantage of pooling the entire hidden state to capture complex interactions in dense crowds.

5.1. Analyzing the predicted paths

Our quantitative evaluation in the Sec. 5 shows that the learned Social-LSTM model outperforms state-of-the-art methods on standard datasets. In this section, we try to gain more insights on the actual behaviour of our model in different crowd settings. We qualitatively study the performance of our Social-LSTM method on social scenes where individuals interact with each others in a specific pattern.

We present an example scene occupied by four individuals in Figure 7. We visualize the distribution of the paths predicted by our model at different time-instants. The first and third rows in Figure 7 show the current position of each person as well as their true trajectory (solid line for the future path and dashed line for the past). The second and fourth rows show our Social-LSTM prediction for the next 12.4 secs. In these scenes, we observe three people(2,3,4) walking close to each other and a fourth person(1) walking farther away from them.

Our model predicts a linear path for person(1) at all times. The distribution for person (1) is similar across time indicating that the speed of the person is constant.

We can observe more interesting patterns in the predicted trajectories for the 3-person group. In particular, our model makes intelligent route choices to yield for others and preempt future collisions. For instance, at time-steps 2, 4, and 5 our model predicts a deviation from the linear paths for person(3) and person(4), even before the start of the actual turn. At time-step 3 and 4, we notice that the Social-LSTM predicts a “halt” for person(3) in order to yield for person(1). Interestingly at time-step 4, the location of the halting point is updated to match the true turning-point in the path. At the next time-step, with more observations, the model is able to correctly predict the full turn anchored at that point.

In Figure 8, we illustrate the prediction results of our Social-LSTM, the SF model [10] and the linear baseline on one of the ETH datasets. When people walk in a group or as e.g. a couple, our model is able to jointly predict their trajectories. It is interesting to note that unlike Social Forces[12] we do not explicitly model group behavior. However, our model is better at predicting grouped trajectories in a holistic fashion. In the last row of Figure 8, we show some failure cases, i.e., when our predictions are worse than previous works. We either predict a a linear path (2nd column) or decelerate earlier (1st and 3rd column) than needed. Although the trajectories do not match the ground-truth in these cases, our Social-LSTM still outputs ”plausible” trajectories, i.e., trajectories that humans could have taken. For instance, in the first and third columns, our model slows down to avoid a potential collision with the person ahead.
Figure 7: We visualize the probability distribution of the predicted paths for 4 people moving in a scene across 6 time steps. The sub-caption describes what our model is predicting. At each time-step: the solid lines in rows 1,3 represents the ground-truth future trajectories, the dashed lines refer to the observed positions till that time-step and the dots denote the position at that time-step. We notice that our model often correctly predicts the future paths in challenging settings with non-linear motions. We analyze these figures in more details in Sec. 5.1. Note that T stands for time and the id (1 to 4) denote person ids. More examples are provided in the supplementary material.

5.2. Discussions and limitations

We are far from predicting all the nuances in human navigation. However, our experiments show encouraging results towards our claim that a data-driven approach has the potential to learn general rules on human navigation as well as nuances behind human be-
Figure 8: Illustration of our Social-LSTM method predicting trajectories. On the first 3 rows, we show examples where our model successfully predicts the trajectories with small errors (in terms of position and speed). We also show other methods such as Social Forces [12] and linear method. The last row represents failure cases, e.g., person slowed down or took a linear path. Nevertheless, our Social-LSTM method predicts a plausible path. The results are shown on ETH dataset [10].

haviour. Heuristic-based approaches that have been tried in the past can only capture the general rules of motion, but won’t be adequate when it comes to capturing the characteristic and subtleties of human motion, which might at times be totally unexpected or random. We believe that a model that has observed human behaviour for quite some time, can come close to account for these irregularities in human motion, or if not that, then at least, react to this sudden anomaly in the most consistent way.

The current set of quantitative experiments assume that there is a single ground truth path to predict. Given the same social context, several plausible paths are possible. As a future work, we will investigate other metrics involving humans in the loop for the evaluation of the predicted paths. We can run experiments to study the number of paths generated by our forecasting model that are "plausible" and "socially-accepted".
6. Conclusions

We have presented two families of methods to forecast human trajectories in crowded scenes. The former is based on Social Forces model and has the capacity to encode the physics behind navigation. The latter is a fully data driven method based on LSTM and has the capacity to encode complex interactions that one might not be aware of. Given a set of experiments on public datasets, the LSTM-based model outperforms other methods. It can jointly reason across multiple individuals to predict human trajectories in a scene. Future work will study the impact of data driven methods in multi-class settings where several objects such as bicycles, skateboards, carts, and pedestrians share the same space. In addition, human-space interaction will also be studied to forecast abrupt non-linear behaviors due to the static scene.

References

URL http://arxiv.org/abs/1506.02216


[70] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, et al., Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks, ICCV.


