Pedestrian Motion Prediction with Deep Learning

Pedro Miguel Gustavo Bilro

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Ana Catarina Fidalgo Barata
Prof. Jorge dos Santos Salvador Marques

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Ana Catarina Fidalgo Barata
Member of the Committee: Prof. Mário Alexandre Teles de Figueiredo

November 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would first like to thank my close family - my parents and older brother - for the unconditional and reliable support and caring throughout my whole life. Everything I have and will accomplish would mean nothing if I did not have them to share it with. I also want to thank my grandparents, uncles, and the rest of my family, for being there for me throughout all these years.

I must give a very special thank you to my dissertation supervisors, Prof. Jorge Marques and Dr. Ana Barata. It was their advice, suggestions, and support that made achieving the thesis objectives possible.

I would also like to thank researcher Parth Kothari from VITA lab at EPFL, for the help and suggestions regarding specific topics of the thesis work.

I would like to thank Ana Ribeiro for being there for me, through thick and thin, day in, day out, for the past two years. For making every small win feel bigger. For forcing me to take breaks and rest, especially when I did not realise I needed it. Thank you.

Finally, I would like to thank all my friends and colleagues, both inside and outside university, that helped me grow academically, professionally, and personally.

To each and every one of you – Thank you.

Abstract

Pedestrian motion prediction is a task that is relevant for many kinds of intelligent systems, such as autonomous vehicles, social robots, and automated surveillance systems. It can be a very challenging task, due to the fact that humans can be influenced by a plethora of factors. In recent years, two types of cues have been getting more relevance: the presence of physical obstacles, and the existence of social interactions between pedestrians. While there have been methods that incorporate both types of cues, most require extra information such as video images, which may not be readily available.

This thesis proposes a novel method, based on Long Short Term Memory, that takes into account obstacles and neighbouring humans to robustly predict pedestrian trajectories. The presence of obstacles is incorporated by integrating the model with Sparse Motion Fields, that learn regions absent of pedestrian motion in an unsupervised way. The existence of interactions is incorporated with a pooling layer that simulates a field of view for each pedestrian.

The proposed method is able to outperform several state-of-the-art models on popular pedestrian datasets. To compare the models, there was the use of standard geometric errors, as well as metrics specifically related to static obstacles and dynamic interactions. The experiments also included visualization of method predictions, to provide a different perspective on the performance of trajectory forecasting models. The model implementation and experiments are publicly available and can be accessed at `https://github.com/pedro-mgb/pedestrian-arc-lstm-smf`.

Keywords

Machine learning; Trajectory prediction; Social interactions; Obstacle awareness

Resumo

A previsão de movimento de pedestres é uma tarefa relevante para muitos tipos de sistemas automáticos, como veículos autónomos, robôs sociais, e sistemas automatizados de videovigilância. Pode ser uma tarefa bastante desafiante, devido ao facto de que os humanos podem ser influenciados por múltiplos fatores. Nos últimos anos, tem sido dada mais importância a dois tipos de fatores: a presença de obstáculos físicos, e a existência de interações sociais entre pedestres. Embora existem métodos que incorporam ambos estes fatores, grande parte destes necessitam de informação adicional como imagens de vídeo, que poderão não estar constantemente disponíveis.

Esta tese propõe um método novo, baseado em redes Long Short Term Memory, que tem em conta obstáculos e humanos vizinhos para prever trajetórias de pedestres corretamente. A presença de obstáculos é incorporada usando campos de movimento esparsos, um método que aprende regiões ausentes de pedestres de forma não supervisionada. A existência de interações sociais é incorporada com um módulo auxiliar que simula um campo de visão para cada pedestre.

O método proposto consegue superar vários trabalhos de estado-de-arte, em conjuntos de dados populares na tarefa de previsão de movimento de pedestres. Para comparar os modelos, utilizou-se métricas geométricas, bem como métricas relacionadas especificamente com obstáculos e interações. As experiências também incluíram visualização de trajetórias previstas, de modo a fornecer uma perspetiva diferente no desempenho de modelos de previsão de trajetórias. A implementação do modelo, bem como as experiências realizadas, estão disponíveis publicamente, e podem ser acedidas visitando `https://github.com/pedro-mgb/pedestrian-arc-lstm-smf` (em inglês).

Palavras Chave

Aprendizagem automática; Previsão de trajectórias; Interações sociais; Percepção de obstáculos

Contents

List of Figures

List of Tables

Acronyms

ADE	Average Displacement Error
ANN	Artificial Neural Network
BPTT	Back Propagation Through Time
CNN	Convolutional Neural Network
CV	Constant Velocity
CSE	Collisions with Scene Environment
DCM	Discrete Choice Model
EM	Expectation-Maximization
FOA	Field of Attention
FOV	Field of View
FDE	Final Displacement Error
GAN	Generative Adversarial Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
GT	Ground Truth
KDE-NLL	Kernel Density Estimate-based Negative Log-Likelihood
LSTM	Long Short Term Memory
NLL	Negative Log-Likelihood
NLP	Natural Language Processing
OSB	Out of Scene’s Bounds
ResNet	Residual Network
RNN	Recurrent Neural Network
SDD	Stanford Drone Dataset

SMF	Sparse Motion Fields
STGCNN	Spatial Temporal Graph Convolutional Neural Network

1 Introduction

Contents

1.1 Motivation

Knowing how pedestrians move and interact with their surrounding environment, which can include pedestrians and other obstacles, is a crucial process for many kinds of intelligent systems. In the same way that humans are constantly adjusting their path according to their surroundings, service robots also need to perform this kind of adjustment in order to coexist with humans [1]. Human drivers anticipate the trajectory of pedestrians, drawing and planning the vehicle's trajectory mentally. This is a process that must be present in autonomous vehicles [2], keeping in mind goals like collision avoidance, and overall safety of both the vehicle's passengers, other drivers, and pedestrians. Surveillance systems may need to analyze motion of people in a video scene and infer what kind of activities are they conducting or will conduct in the near future [3, 4].

A major task present in all of the above settings consists in predicting the actual trajectory of pedestrians, be it in short-term duration, long-term duration, or both. Pedestrian trajectory prediction can be summed up to predicting where a pedestrian will go for a certain foreseeable future.

This is a very challenging task, because the motion of pedestrians can depend on a variety of factors, as seen in fig. 1.1. These factors include, among others, the person's characteristics [5] (like age, health, and emotional state), the person's goals (where they want to go and what they want to do), the presence of physical obstacles and unwalkable areas [6] (*e.g.*, a building or a tree), or of other people [7].

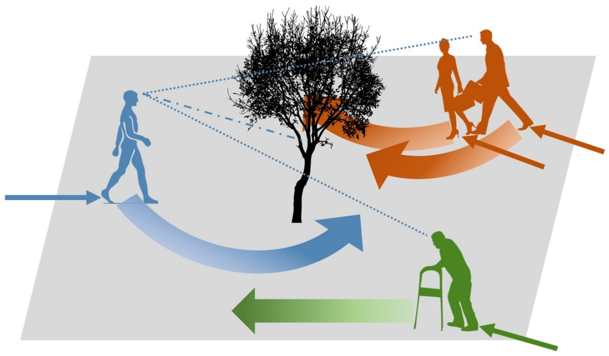


Figure 1.1: Illustration of the trajectory prediction problem and its difficulty. In this example, the man in blue (left side of the figure) will adjust his trajectory to avoid colliding with a tree, and with other people also present in the scene. Similarly, a trajectory prediction method will need to deal with the presence of static obstacles, and of other pedestrians that can interact with each other in a diverse manner of ways. The model's goal is to generate trajectories that seem plausible when compared with what the real future trajectory would likely be.

For a system to accurately predict how a pedestrian will move in the future, it needs to integrate as many of these factors into its predictive model as possible. The difficulty lies on having a model that generalizes well to different kinds of trajectories, while also generating good predictions even if there is high variation in the modes of motion that govern pedestrian trajectories.

For several of the applications listed above, predicting a single trajectory for each pedestrian is not enough - there is the need to generate a sample of trajectories that capture the big variety of paths that a pedestrian can take [8]. This is evident for the case of self-driving cars [9], that need to take into consideration the fact that a pedestrian has several possible alternatives regarding his/her trajectory, and if the predictive model can generate a distribution that robustly captures all of them, then the car will be better apt to adjust its path.

In recent years, there have been advancements in creating data-driven models, mainly Artificial Neural Networks (ANNs), to solve the trajectory prediction problem. There have been data-driven models that incorporate the presence of obstacles and ”forbidden zones” [2, 10], and models that account for social interactions between pedestrians [8, 11], such as group behaviour and collision avoidance (when possible). There have even been models that try to capture both the aforementioned cues [9, 12, 13].

However, these models suffer from some limitations. Those that only consider scene-specific elements will tend to perform poorly in crowded situations, where there are bound to be plenty of interactions between pedestrians. The models that only take into account social interactions may generate trajectories that go against the physical properties of the scene, like colliding with a static obstacle. While existing methods that incorporate both the aforementioned cues tend to combat these limitations, all existing methods require additional information besides the trajectories of pedestrians (*e.g.*, frames from a corresponding video to identify static obstacles in a scene, or to construct a semantic map) to accurately perform prediction. This calls for the need of novel approaches to the trajectory prediction task, that incorporate the aforementioned cues with an approach based solely on past pedestrian trajectories.

1.2 Objective and problem formulation

The objective of this thesis is to use the recent advancements in deep learning to create a data-driven model that takes into account physical aspects of the scene and interactions between pedestrians to generate accurate trajectory predictions. The model should also be robust to new and unseen situations, without requiring extra information such as video frames to extract or learn relevant scene information.

The actual problem of trajectory prediction can be formulated in the following way: given a set of observed trajectories (identified as the past), one for each person, the objective consists in generating trajectories compliant with the past, and in accordance to what the real trajectories would actually be (”close” to the real future trajectories). This prediction should preferably consider as many influencing cues (mainly scene information, and social interactions) as possible.

Mathematically, the problem can be parameterized with the notation similar to [11]:

- For each time instant t, and for each person in a scene (at time t), $i \in \{1, ..., N\}$, where N is the total number of people, there will be a 2D vector identifying the position of person i at time t, $X_i^t \in \mathbb{R}^2$. This means that the person is simply represented by a single point.

- For each person there will be an observed trajectory, $\mathbb{X}_i$, a sequence of X_i^t values, with discrete time step $t \in \{T_{ini}, ..., T_{obs}\}$, where T_{ini} is the time instant that marks the beginning of the observed trajectory, T_{obs} is the time instant for the last observed position . These sequences will serve as input for trajectory prediction.

- The objective is to estimate a future sequence ($\hat{\mathbb{Y}}_i$) of 2D positions for each person, $\hat{Y}_i^t \in \mathbb{R}^2$, $t \in \{T_{obs+1}, ..., T_{pred}\}$, with T_{pred} being the final instant of prediction. This is the desired output, that should be as close to the real sequence $\mathbb{Y}_i$ - sequence of Y_i^t, $t \in \{T_{obs+1}, ..., T_{pred}\}$ -, as possible. While X_i^t and Y_i^t have the same content, this distinction in notation will be helpful in chapter 3.

Data-driven models for trajectory prediction usually estimate parameters via a training set. This training set is made up of past pedestrian trajectories, as mentioned above, and also the future 2D sequences, *i.e.*, what the model should attempt to output. To assess the quality of a model, a test set is used, containing a set of future 2D sequences - Ground Truth (GT) - to compare with the predictions.

Besides 2D coordinates, there are also extra options that can be used as input. One example consists in using velocity and position, and so one element of a trajectory would consist of 4 elements rather than 2: (X_i^t, V_i^t), where $V_i^t \in \mathbb{R}^2$ is corresponds to the 2D instantaneous velocity at time t. Instead of velocity, a sequence of displacements ΔX_i^t could be considered, which is equivalent to multiplying the velocities by the associated time interval, Δt. Note that, despite the different kind of inputs, the same objective is always present, to predict sequences of positions for all people in the scene.

1.3 Thesis structure

This document is organized as follows:

- Chapter 2 contains an overview of related work in these topics, with short description and discussion of some ground-breaking methods in the task of trajectory prediction.

- Chapter 3 describes the proposed model to solve the trajectory prediction problem with the incorporation of social interactions between pedestrians and relevant scene information. This will also involve a detailed description of how the model incorporates each of these cues separately, and how they were combined in the final model.

- Chapter 4 contains the experiments made with the proposed methods, also comparing with some methods stated in Chapter 2, both quantitatively and qualitatively. This chapter also includes a discussion of the obtained results.

- Finally, in Chapter 5 some short conclusions are made regarding the overall work done, and future work is proposed along with possible directions to explore in the task of trajectory forecasting.

2

Related work

Contents

Human trajectory prediction methods can be categorized in several ways. This Chapter adopts the taxonomies from [5]. Many recent motion prediction methods have been categorized as ”pattern-based”, aiming to learn motion patterns by observing trajectory data. Most of the methods covered in this chapter have also been categorized as ”pattern-based”, and the proposed method in Chapter 3 is also of this type. The other categories are ”physics-based” models, that predict motion based on dynamical equations such as Newton’s laws of motion, and ”planning-based” models, that attempt to reason about a pedestrian’s goals and generate the path that the pedestrian would take to reach that goal.

The methods can also be divided into the type of contextual cues they incorporate [5], *i.e.*, what kind of characteristics that influence human motion are going to be considered in the model. The considered cues were already introduced, to some extent, in Chapter 1. The dynamic environment cues consider the presence and interactions between people and/or groups of people. There are also static environment cues, which incorporate the presence of physical obstacles, and overall physical aspects of a scene. Finally, pedestrian-specific cues[1] comprise the attributes used to characterize a person. Sections 2.1 to 2.3 will be about each of the aforementioned cues.

As an extension to this taxonomy proposed in [5], Section 2.4 is dedicated to works that incorporate both social and scene information in the predictive model. Finally, Section 2.5 contains a discussion on popular datasets used to train and test models, as well as recent efforts to create new ways of evaluating the quality of the predicted trajectories, in all of their inherent characteristics.

2.1 Pedestrian characterization

There are many possibilities when it comes to characterizing a pedestrian. A simple consideration is to model a person by a single point (*e.g.*, mass center). Several pattern-based methods follow this approach, representing a person by a 2D point (x and y coordinates) in a scene, usually storing a history or sequence of positions [6, 8, 9, 11], as presented in Section 1.2. Others may use the position of a person along with the velocity [14, 15]. Some works have not taken a point representation of a person, and instead have a bounding box surrounding a person in a video scene [16].

More complex forms of modelling take into account head orientation of the person, or even a fully articulated pose, considering several body joints [17–19], to have more information in regard to generating the future trajectory. These approaches are more promising in robustly capturing the characteristics of human motion. However, they are less common than the simpler representations mentioned above. This is mainly due to the fact that access to such detailed characteristics may be difficult [20, 21]. Moreover, the complexity of the underlying model increases considerably.

[1]The original term from [5] was actually agent-specific cues, with agent being a more general term that could potentially refer not only to pedestrains, but also cyclists, vehicles, and other entities to which it can be relevant to perform trajectory prediction.

2.2 Methods that incorporate social interactions

Classical works are based on physics laws to model human motion considering the presence of other people. Such is the case of the Social Force model [7]. This model, inspired in physics, considers standard repulsion and attraction forces to incorporate the influence of the neighbours of each pedestrian in his/her own trajectory. This approach, or variants of it, are still being used in recent works [18, 21, 22]. It usually requires the use of handcrafted energy potentials, which makes model generalization difficult.

In recent years, the best performance has been obtained with the use of ANNs. The most popular type has been Recurrent Neural Networks (RNNs), that excel at processing sequential data - with a trajectory being a sequence of positions set recurrently through time, An architecture commonly used in this task has been Long Short Term Memory (LSTM) [23, 24]. LSTM networks are more apt at capturing long term dependencies in sequences than plain RNNs.

Pioneering work proposed in [11], named Social LSTM, made use of LSTM networks to predict pedestrian trajectories. For each pedestrian, there is one LSTM network to observe one past trajectory, using it to condition the predicted trajectory, *i.e.*, the desired output. All LSTM networks share the same parameters, so that the model does not depend on the number of pedestrians. These parameters are estimated in the training phase of the model, using 2D sequences coming from a training set. The incorporation of social constraints is done via a Social Pooling Layer - a handcrafted grid is created around each of the pedestrians, to consider the influences of other pedestrians in his/her local neighbourhood. The Social pooling mechanism is illustrated fig. 2.1.

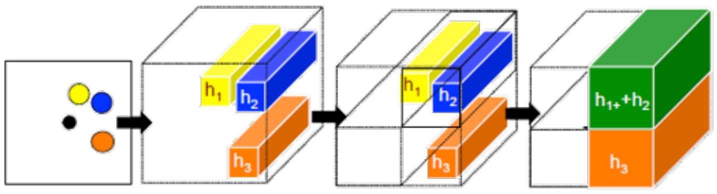


Figure 2.1: Example of how Social LSTM model considers influence of other pedestrians. A point wise representation is used for each pedestrian. Considering the person represented by a black dot, a handcrafted square grid is built surrounding that pedestrian. The neighbour pedestrians have their LSTM hidden states (see section 3.1 for more details on what it means) accumulated across each cell of the grid. Credits to the original image go to [11].

This model has some limitations. For instance, it is not able to encompass all possible interactions between people, but only those in the immediate vicinity (inside the grid), although it may be argued that pedestrians farther away will not have influence in the real traversal. The actual grid size has to be manually defined, and different contexts or even pedestrians may benefit from a different grid size, thus hampering generalization. Nonetheless, the popularity and performance improvement of this architecture when compared with more classical models [7] has led to the development of several variants and extensions [25–28]. Directional LSTM [26] is a simpler but improved adaption of the Social-LSTM model that uses the relative velocities of pedestrians in place of the LSTM states, which are hard to interpret.

Generative Adversarial Networks (GANs) [29], and variants of GANs [30, 31], have also been used in the task of trajectory prediction [8, 14], usually in conjunction with LSTM. The first incorporation of GANs for trajectory prediction was in [8], with Social GAN. The goal of the model is to generate multiple trajectories for the same pedestrian (instead of just one like previous works [11,22]), to better capture the multimodal nature of human motion. The incorporation of interactions was done via a Pooling Module that considers relative distances between all pedestrians, instead of just those in close proximity like in [11], and through a variety loss that encourages diverse but geometrically accurate sample generation.

The variety loss has been proven to not lead to the actual ground truth distribution [32]. This motivated the use of different ways of training GANs for trajectory forecast. For instance, Info-GAN [31] has been used in [14]. This also included additional social featuresto compute a set of attention weights that encapsulate how much attention each pedestrian gives to every other.

Some works have used other kinds of networks to solve this task. For instance, Mohamed et al. [33] have obtained competitive results with their model based on Spatial Temporal Graph Convolutional Neural Networks (STGCNNs) [34].

Recently, more focus has been given to making data-driven models interpretable [26,28]. The Social Anchors model from [28] incorporates Discrete Choice Models (DCMs) [35] along with LSTM, to give more interpretability on the choices the network makes for the pedestrian's path, given the presence of other pedestrians. Understanding what makes the network perform a specific prediction is important for several kinds of safety-critical intelligent systems (*e.g.*, autonomous vehicles). Thus, the interpretability of data-driven models for trajectory forecast is a promising and important area of research in the future.

Methods that only take into account interactions between pedestrians have the limitation of not being able to generate trajectories that are compliant with the scene in which they are inserted. This means that the methods discussed in this Section are likely to generate trajectories where pedestrians collide with static obstacles or even go outside the scene's walkable areas. This is due to the fact that these methods are not taking into account scene-specific information for trajectory prediction.

2.3 Methods that capture scene-specific information

The context of the scene may, in some cases, pose as much if not more weight than social interactions for the trajectory prediction task [2,6,10]. However, the usage of these types of cues in motion prediction models has been much smaller when compared to cues related to pedestrian interactions [5].

Some works have tried to incorporate the exact pose of obstacles into the predictive model [36], which can be problematic in an environment with many objects [5]. Pattern-based approaches commonly rely on semantic maps [2, 10, 37] that have additional information such as the kind of obstacles that are present (roads, buildings, *etc.*). These semantic maps aren't usually supplied as input, but extracted from video frames [2, 10], assuming they are accessible along with trajectory data [20, 21, 38].

The CAR-NET [2] model takes as input pedestrian trajectories and a top-view image of the scene. The latter is supplied to a Convolutional Neural Network (CNN) [39] in order to extract feature maps through scene segmentation techniques [40]. These feature maps allow the identification of scene semantics (*e.g.*, obstacles). They are supplied to an attention module to determine the relevant regions to consider feeding into a LSTM network for trajectory prediction.

Ridel et al. [10] also extracted features from the scene, but using Residual Networks (ResNets) [41]. Additionally, they used U-Net [42] to process the pedestrians' trajectories. The result of these networks is merged to form a probability grid, with each cell indicating the probability of a pedestrian being at that cell at a certain time step. This grid is used as input to a trajectory generator module.

The use of feature extractors can make the model computationally heavier, and one can think of retrieving scene information solely from trajectory data. A recent work makes use of Sparse Motion Fields (SMF) [6] to learn physical aspects of the scene without the use of video frames, therefore not needing feature extractors like CNNs - which also usually require pre-training with extra data. The use of SMF restricts the learnt representation of pedestrians' motion to areas where motion is actually possible, implicitly considering other areas as potential obstacles or zones where traversal is not possible. An example of the predicted motion fields and their sparsity is shown in fig. 2.2.

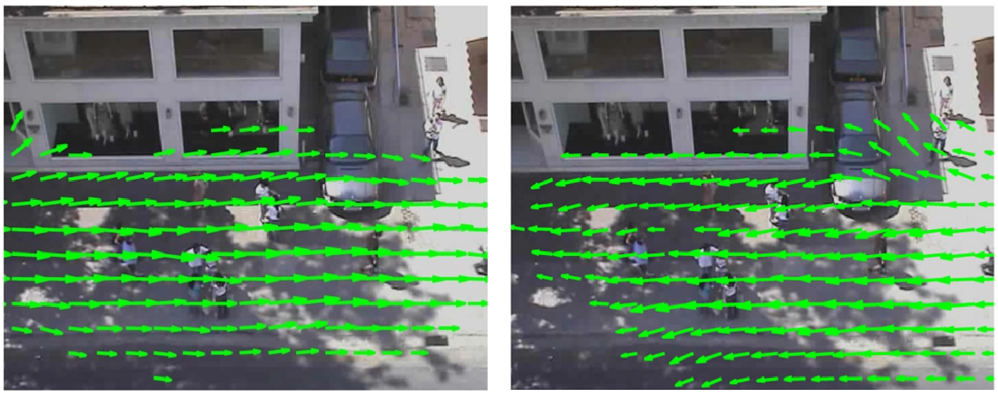

(a) (b)

Figure 2.2: Examples of predicted sparse motion fields on a scene apart of a real trajectory forecasting dataset [20]. It can be seen how most of the area of the building, and also the vehicles, is discarded. The movement of pedestrians is restricted to the walkable areas of the scene. Credits to the original images go to [6]

While scene-specific elements should be considered for trajectory prediction, using it on its own means that the predictive models do not consider the influence of other pedestrians. Hence, the models may have worse results in crowded situations, where social interactions can play a very relevant part.

2.4 Methods that use scene information and social interactions

In the past couple of years, there has been a considerable amount of research into incorporating both social and scene cues into a predictive model [9, 12, 13, 16, 43]. Incorporating both these kinds of cues has actually been more common than just incorporating scene-specific cues [5].

The social force model originally proposed in [7] can account for both social interactions and interactions with obstacles (which can be modelled as entities of constant position). This classical model has been adapted in more recent works [21, 44, 45].

Some works took as basis Social LSTM [11], extending it to incorporate the scene's geometry, either by explicitly defining the positions of static obstacles [15], or by extracting some information from scene images [46, 47]. Others have also used GANs similarly to [8], achieving very good results [9, 12, 48].

Sophie [9], a model based on CAR-NET [2] (see Section 2.3), uses the same kind of CNNs [39,40] to extract features from scene images. This model also includes social interactions in its attention module, to give a context vector regarding each pedestrian, highlighting the influence of the neighbours. Social-BiGAT [43] also made use of a CNN to extract features, but used a Bicycle GAN [49] to improve the sample generation, and Graph Attention Network [50] to incorporate interactions between pedestrians.

The NEXT model from [16] uses bounding boxes surrounding people and objects in a video scene, together with semantic maps. These maps are extracted from the scenes, identifying areas like roads, grass and sidewalks. The several inputs are combined in a module used to supply information for trajectory generation. This model was also used to solve a different task - activity recognition, *i.e.*, identifying what the person is doing or going to do (*e.g.*, walking, running, entering a car, *etc.*).

A recent model that has had the best performance in trajectory forecast is the Trajectron++ [13,51]. It incorporates LSTMs networks with kinds of RNNs. It also uses CNNs for scene processing (if available). Interactions between pedestrians - and potentially other types of agents such as cyclists or skaters - are encoded as edges of a spatio-temporal graph.

Table 2.1 summarizes some of the presented trajectory prediction methods, comparing them in several ways: whether they take into account social constraints, scene constrains, or both; what inputs the models expect to receive; what kind of networks/algorithms are they based on; are they open source, *i.e.*, do they have a publicly available implementation, which is relevant for the sake of performing some experiments with the models (refer to Chapter 4).

All methods in this section, as well most in Section 2.3, require some additional information regarding the scene to perform accurate trajectory prediction, which increases the computational weight of the model, while also adding an additional dependency on videos that may not always be available (*e.g.*, access to Global Positioning System (GPS) data alone [55]). To the best of our knowledge, there has not been a model to incorporate both social and scene constraints, using only trajectory data as input.

Table 2.1: Summary of some trajectory prediction methods, organized by date of publication, in ascending order.

Method	Social	Scene	Inputs	Model(s) and Algorithm(s) used	Open Source
Social LSTM [11]	Yes	No	Trajectory Data	LSTMs. Handcrafted grids	Yes
CAR-NET [2]	No	Yes	Trajectory data Top-view images	CNNs [39, 40]. LSTMs. Attention Module	No
Social GAN [8]	Yes	No	Trajectory Data	LSTMs. GANs. Pooling with relative distances	Yes
SoPhie [9]	Yes	Yes	Trajectory data Top-view images	CNNs [39, 40]. LSTMs. Social and physical attention	No
NEXT [16]	Yes	Yes	Trajectory data Images with bounding boxes	CNNs. LSTMs. Person interaction and behaviour modules	Yes
Ridel et al. [10]	No	Yes	Trajectory data Top-view images	ResNet [41]. U-Net [42]. ConvLSTM [52, 53]	No
SMF [6]	No	Yes	Trajectory data	Motion Fields. EM algorithm [54]	No
Trajectron++ [13]	Yes	Yes	Trajectory Data	LSTMs. Other RNNs. CNNs. Spatio-temporal graphs	Yes
Social-STGCNN [33]	Yes	No	Trajectory Data	STGCNN [34]	Yes

2.5 Datasets and evaluation metrics

The increase of methods for trajectory prediction would not have occurred without the existence of proper datasets to be able to not only test methods and evaluate the outputs, but also, in the case of almost every method, to train models in order to learn their parameters.

Two of the most popular datasets are BIWI Walking Pedestrians dataset [21] (commonly known as ETH) and Crowds dataset [20] (commonly known as UCY). Both these datasets consist of videos taken from a higher point of view for certain locations (2 for BIWI, 2 for Crowds, although for the case of Crowds there is data for the same scene/location at more than one time interval). The actual pedestrian trajectories have been extracted using a constant sampling rate of $2.5\,\mathrm{Hz}$. BIWI dataset has a lower average crowd density than Crowds dataset, meaning the latter is usually more interesting and impactful for methods that incorporate social interactions.

A more recent and large dataset is the Stanford Drone Dataset (SDD) [38], with bird's eye view of eight different scenes. These scenes not only contain trajectories of pedestrians, but also of cyclists and cars, among others. Another large-scale dataset, applied to the context of autonomous driving systems, is the nuScenes dataset [56]. This dataset has much more data than just plain 2D trajectories, to encompass the plethora of sensors and data that real autonomous vehicles have access to.

Evaluation in the context of trajectory prediction has been somewhat standardized [5]. The most common types of metrics are geometry-based. The Average Displacement Error (ADE) is the most common metric used to evaluate the quality of the predictions [5]. Essentially, it involves computing the euclidean distance between each predicted position and the real position (coming from GT data), and performing an average over it. Another very common metric is the Final Displacement Error (FDE), the euclidean distance between prediction and GT for the final position (last time step) in the trajectory.

For a probabilistic model that can generate multiple samples of trajectories, several works [8, 9, 12] have selected the minimum value of ADE and FDE from a set of s sampled (bestOf-s-ADE/FDE) trajectories, $s > 1$. However, a top-$n\%$ ADE and FDE has shown to provide more stable results [5]. Recent works [13, 26, 32, 51] have employed a Kernel Density Estimate-based Negative Log-Likelihood (KDE-NLL), which involves using the entirety of all trajectory samples to create a distribution and compute the metric. This makes it a more robust way of quantitatively evaluating the performance of multimodal methods than the prior metrics mentioned in this paragraph.

The plain use of distance-based or probabilistic metrics does not give any indication of compliance with physical and social cues that influence human motion, such as scene-specific elements or the interactions between pedestrians. Few metrics have been developed with respect to trajectories being in accordance with both those types of cues [5].

Ridel et al. [10] and others [13] have employed (different) metrics to evaluate compliance with the scene, considering percentages of trajectories going into invalid areas where traversal is not possible, or that result in collisions with obstacles. However, these metrics were only used for an ablation study, therefore not comparing between different state-of-the-art methods.

The authors of Sophie [9], and also others [48] used an interaction-relevant metric - an average percentage of near collisions between pedestrians in the same video frame. A near collision was identified when two pedestrians in the same frame were at a distance lower than a certain threshold ($0.1\,\mathrm{m}$ was used). Lower values would indicate the model generates more socially accurate trajectories.

There have been few, if any, methods that use a robust quantitative evaluation considering both scene and interaction-relevant cues. This will be something that will also be explored and tackled in chapter 4, providing a more complete evaluation of trajectory forecasting models.

In recent years, some benchmarks for trajectory prediction have been defined. One example of that is Trajnet [57], providing popular datasets [20, 21, 38] and metrics (ADE and FDE). A new and more complete benchmark, Trajnet++ [26], to compare trajectory forecasting for crowded places, was recently released, which has recent datasets [58–60] and metrics (like percentages of colliding pedestrians), as well as some baseline methods for comparison [8, 11]. The fact that Trajnet++ considers socially relevant data motivated the choice to use it for the experiments of this thesis (more information can be found in section 4.1), while also going beyond the benchmark to consider a scene-specific evaluation.

3

Proposed solution

Contents

The main objective of this thesis is to develop a pedestrian trajectory prediction method, which can learn scene-specific information like obstacles and unwalkable zones, while also taking into account social interactions between pedestrians. The proposed solution does not require video frames or scene images, dropping the need to use computationally heavy networks (*e.g.*, CNNs) for feature extraction.

At the core of the proposed model are LSTM networks [23, 24], that have proved to excel at processing sequential data, including sequences of pedestrian positions [8, 11]. To help incorporating cues that influence human motion, one can build upon some of the concepts introduced in Chapter 2. Namely, this work will integrate the SMF model, developed in [6, 61], with LSTM networks for scene-compliant predictions, to be detailed further in this chapter. The influence of pedestrians on each other will be based upon the relative velocities between them, as used by Directional LSTM [26].

The proposed solution will be described in parts The first few sections will describe the networks and methods that serve as the building parts for the full model. Afterwards, the incorporation scene and social cues will be detailed (in this order). The last section will feature the final model, which encapsulates scene and social cues for accurate pedestrian motion forecast. New developments and contributions are made in sections 3.4, 3.5, and 3.6.

3.1 LSTM networks

The prediction of human motion can be seen as a sequence generation task [11, 62]. A trajectory can be formulated as a sequence of positions throughout a certain span of time. The explicit presence of time makes the trajectories inherently recurrent - the future trajectory of a pedestrian can depend on the overall past path taken by the same pedestrian.

One class of data-driven architecture specially designed to process sequential data is RNNs. For this class of networks, the data is correlated across one or more dimensions (*e.g.*, space, time, or both). Prior works in the area of trajectory forecast [8, 11, 13] have obtained success with LSTM networks [23, 24], a more specialized type of RNN, and the proposed work will be based on this kind of network too. RNNs, in general, can be identified by the following dynamic recurrent equation[1]:

$$h^t = F\left(h^{t-1}; x^t; W\right), \tag{3.1}$$

where h^t is the state of the recurrent network at time t, also known as the hidden unit of the network. The h^t can also be seen as the memory of the network. F is a generic function that receives the state at the previous time step, h^{t-1}, and an input x^t, and depends on a set of parameters W of the network (weights and bias vectors). Note that W does not depend on the time step[2], meaning function F is

[1]In control theory, the system defined by 3.1 is known as a non-linear dynamic system.
[2]In control theory, such a system can be classified as time-invariant.

always applied with the same weights. This is commonly known as parameter sharing. The state h^t can be decoded into a useful output via an output layer. Other layers can be added and appended to the RNN identified by (3.1), in order to apply them to a variety of problems, including trajectory forecasting.

Training a RNN sums down to performing Back Propagation Through Time (BPTT) [63], starting on the final time step of the sequence, and moving backwards to compute gradients and estimating the parameters that comprise the network, W. It is known that the BPTT procedure on a standard RNN can have issues with the propagation of gradients through time. It is common for the gradients to vanish or explode when dealing with longer sequences. Multiple solutions have been proposed in the past to solve this issue. The approach to be adopted in this work involves using LSTM networks.

Each LSTM cell, as originally proposed in [23], relates to a single time step - with parameter sharing still enforced, *i.e.* all cells share the same parameters W_{LSTM} -, and is built upon a recurrent equation similar to (3.1). However, the cell has an additional internal recurrence, denoted by the presence of an internal state s^t, which helps solve the poor gradient propagation. There are also extra units: an input gate unit and an output gate unit. The input gate unit conditions x^t going into (3.1), and the output gate unit conditions the eventual processing of the external state, h^t (equivalent to the h^t in (3.1)).

An important extension proposed in [24] was to add a forget gate unit, conditioning the dependency of s^{t-1} on s^t, having the possibility to ”forget” what is in the state, when it is no longer necessary. This extension is also important for the task of trajectory forecasting. In certain situations, it may be beneficial to ”forget” part of the past of a trajectory, since it may not have relevance for the prediction task.

All these gate units have parameters that will be learned at the training stage. The overall configuration of an LSTM cell is depicted in fig. 3.1, for a single time step.

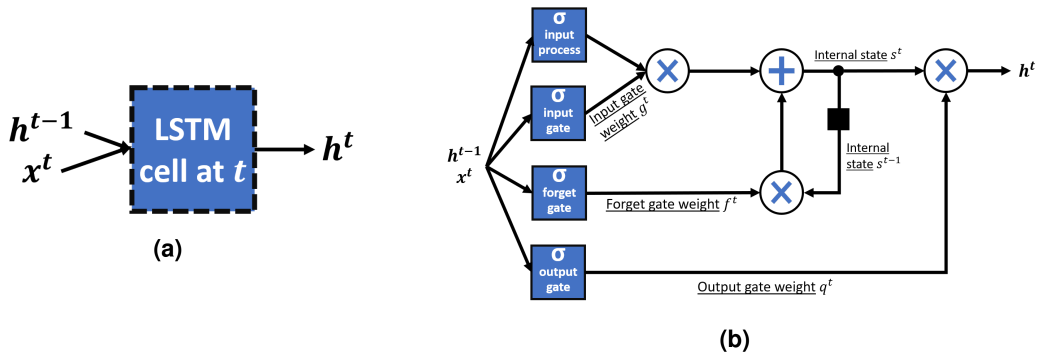


Figure 3.1: Diagram of a generic LSTM cell from [24], for a single time step t. **(a)** The cell receives an input vector x^t and the hidden layer state vector in the previous time step h^{t-1}, and outputs the vector at the next, h^t. **(b)**. The several gate units, in blue squares, have separate parameters that don't depend on the current time, and have either sigmoid or $\tanh$ activation function. The gate units output weights that are used in several products. The LSTM cell possesses an internal state s^t, that is used recurrently, *i.e.*, the s^t is computed with the state s^{t-1}. The black square indicates a delay of one time step.

Each LSTM cell performs the operations in (3.2) - as described in [64] - using the input x^t, and the state h^{t-1}, to compute the state at the next time step, h^t:

$$\begin{aligned}
i^t &= \sigma\left(W_{ii}x^t + b_{ii} + W_{hi}h^{t-1} + b_{hi}\right), \\
g^t &= \tanh\left(W_{ig}x^t + b_{ig} + W_{hg}h^{t-1} + b_{hg}\right), \\
f^t &= \sigma\left(W_{if}x^t + b_{if} + W_{hf}h^{t-1} + b_{hf}\right), \\
o^t &= \sigma\left(W_{io}x^t + b_{io} + W_{ho}h^{t-1} + b_{ho}\right), \\
s^t &= f^t \odot s^{t-1} + i^t \odot g^t, \\
h^t &= o^t \odot \tanh\left(s^t\right),
\end{aligned} \tag{3.2}$$

where i^t, g^t, f^t, and o^t, are respectively the vectors at time t for the processed input, input gate, forget gate, and output gate. Note that each of the four units uses a separate set of weights and biases - with W_{i*} and b_{i*} being for the input to the cell, and W_{h*} and b_{h*} for the previous state ($* \in \{i, g, f, o\}$) -, all to be learned by the model. Two different activation functions are used: σ is the sigmoid function, $tanh$ is the hyperbolic tangent. The $\odot$ represents an element-wise product.

3.2 Trajectory prediction using LSTM networks as base

In the context of the trajectory prediction task, RNNs, and specifically, LSTM can and have been used to forecast trajectories. The general model, as shown in fig. 3.2, would receive a past trajectory - a sequence of 2D positions - to observe and process, and would output the predicted trajectory.

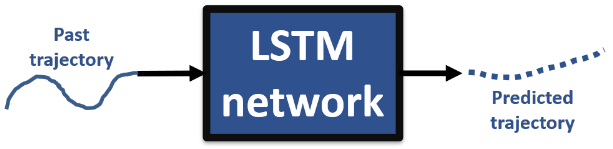


Figure 3.2: A general LSTM model for trajectory forecast. Its input is a past trajectory (to observe or process), and it will predict a future trajectory.

If the model were to directly predict the exact position of every pedestrian at each instant, its performance would be poor. When considering an absolute coordinate system, it can be hard to predict where the pedestrian will exactly be in that system. Furthermore, when dealing with multiple scenes, the coordinate systems of each scene may not coincide. As such, a better alternative is to have the model predict how much the pedestrian moves from the previous instant. It is easier to process and predict relative motion when compared to predicting absolute locations [8, 14, 26, 33]. Therefore, the model in this section, and all LSTM-based models in this chapter, will not process absolute positions, but will instead process the displacements between two consecutive instants. The displacement of pedestrian i at time step t will be $\Delta X_i^t = X_i^t - X_i^{t-1}$ (X is replaced by Y or $\hat{Y}$ for the case of GT or prediction). The predicted trajectory can be obtained by cumulatively summing the predicted displacements:

$$\hat{Y}_i^t = X_i^{T_{obs}} + \sum_{\tau=T_obs+1}^{t} \Delta\hat{Y}_i^\tau, \quad t \in \{T_{obs}+1, ..., T_{pred}\} \tag{3.3}$$

The behaviour of the network can be split in two phases: observation (processing the past, $t \in \{T_{ini}, ..., T_{obs}\}$) and prediction (forecasting the future trajectory, $t \in \{T_{obs}+1, ..., T_{pred}\}$). To simplify the notation, and since this base model will not consider pedestrian interactions, the subscript i (identifying the pedestrian), present in section 1.2, will be omitted from equations. The LSTM model described in this section does not consider any influence of neighbours (refer to section 3.5 for such models).

3.2.1 Phase 1: Observe

In this first phase, the network uses the past trajectory as input to compute the states associated to the past - h^t, $t \in \{T_{ini}, ..., T_{obs}\}$. The network will learn to store the relevant portions of the past motion in the state, propagating it to the prediction phase (section 3.2.2). The model processes a past sequence of displacements, as seen in fig. 3.3, one instant at a time.

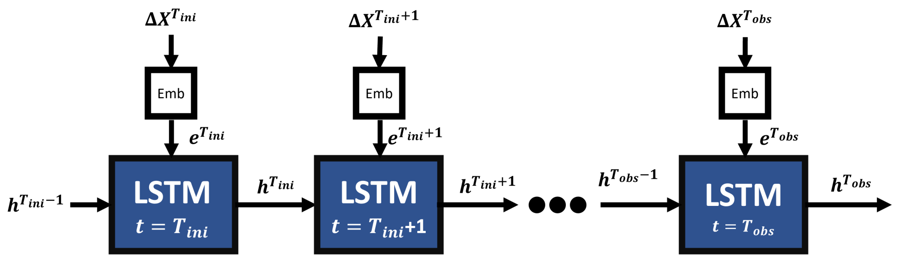


Figure 3.3: The observation phase of the model, unwrapped across time. Each displacement from the past trajectory is fed to the LSTM cell at time t, as long with the state of the previous instant, h^{t-1} to compute h^t. The output will be a state containing a summary of the entire past trajectory

At each instant, the past displacement goes through an embedding layer, converting it into a vector e^t with dimensions on the same order of magnitude as the LSTM state (usually much higher than $\mathbb{R}^2$).

$$\begin{aligned} e^t &= Emb\left(\Delta X^t; W_{emb}\right) \\ &= PReLU\left(W_{emb_{lin}} \Delta X^t + b_{emb_{lin}}\right), \end{aligned} \tag{3.4}$$

where W_{emb} are the parameters of the input embedding, including the weights and biases of the linear (affine) transformation, $W_{emb_{lin}}$ and $b_{emb_{lin}}$, plus a learnable parameter of the $PReLu$ activation function:

$$PReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{otherwise} \end{cases}, \tag{3.5}$$

The $PReLu$ is similar to a $LeakyReLu$, but the slope a is a parameter that is also learned by the model. It has proven to be an appropriate activation function for the trajectory forecasting task in prior works [33].

The state is then obtained via a recurrence analogous to the one present in (3.1):

$$h^t = LSTM\left(h^{t-1}, e^t; W_{LSTM}\right), \tag{3.6}$$

where W_{LSTM} corresponds to the several weights and bias vectors for the standard LSTM network, visible in (3.2). These weights are not dependent on each pedestrian, nor of the current instant - they are the same for all trajectories. Note also that this process requires an initial state, $h^{T_{ini}-1}$. Usually, this initial state is simply filled with zeroes, implying that there is no prior history.

The output of this phase will be the state at the last instant of the observed trajectory, $h^{T_{obs}}$, which will contain the summary and relevant portions of the pedestrian's motion until that instant. This state is only used internally by the model, in the next phase.

3.2.2 Phase 2: Predict

After $h^{T_{obs}}$ has been computed, it can be used by the network for computing not only the next state $h^{T_{obs}+1}$, but also a prediction for the future displacement. This is where the second phase of the model begins. A visual depiction of this phase is present in fig. 3.4.

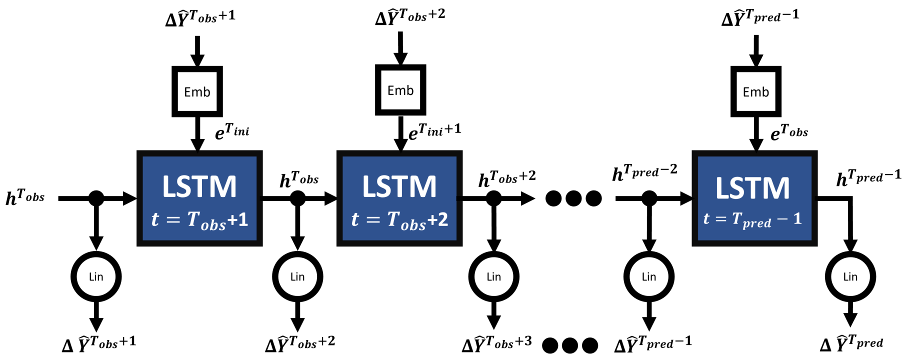

$\Delta\hat{Y}^{T_{obs}+1}$
$\Delta\hat{Y}^{T_{obs}+2}$
$\Delta\hat{Y}^{T_{pred}-1}$
Emb
Emb
Emb
$e^{T_{ini}}$
$e^{T_{ini}+1}$
$e^{T_{obs}}$
$h^{T_{obs}}$
LSTM $t = T_{obs}+1$
$h^{T_{obs}}$
LSTM $t = T_{obs}+2$
$h^{T_{obs}+2}$
$h^{T_{pred}-2}$
LSTM $t = T_{pred}-1$
$h^{T_{pred}-1}$
Lin
Lin
Lin
Lin
Lin
$\Delta\hat{Y}^{T_{obs}+1}$
$\Delta\hat{Y}^{T_{obs}+2}$
$\Delta\hat{Y}^{T_{obs}+3}$
$\Delta\hat{Y}^{T_{pred}-1}$
$\Delta\hat{Y}^{T_{pred}}$

Figure 3.4: The prediction phase of the model, unwrapped across time. The LSTM cells are the same as in observation phase, as well as the embedding layers, but in this phase the model also "decodes" the state (via Lin layer) to obtain the predicted displacement at the next instant. From $t = T_{obs} + 1$ onwards, there is no past to feed to the network, so the prediction $\Delta\hat{Y}^t$ is used instead. The displacements can be cumulatively added to obtain the predicted trajectory

There are two big differences when compared to the observation phase. First, the prediction is obtained by passing the state through a linear layer, that wasn't present in the observation phase. Second, since the past trajectory took place for $t \in \{T_{ini}, ..., T_{obs}\}$, the model has no more data for $t > T_{obs}$. Instead, the model is fed the prediction at that time step, $\hat{Y}^t$.

$$\hat{Y}^{t+1} = W_{out_{lin}} h^t + b_{out_{lin}} \tag{3.7}$$

$$h^t = LSTM\left(h^{t-1}, Emb\left(\hat{Y}^t; W_{emb}\right); W_{LSTM}\right) \tag{3.8}$$

where $W_{out_{lin}}$ and $b_{out_{lin}}$ are respectively, the weights and bias vector for the output linear layer. The weights and biases of the LSTM cell and Emb layer - W_{LSTM} and W_{emb}, respectively - are the same as in (3.4) and (3.6). The output of the model, *i.e.*, the predicted sequence of future positions, can be directly obtained from the displacements, by applying (3.3).

3.2.3 Training

The model is trained to minimize a loss that gets smaller as the predicted trajectory gets closer to the real trajectory coming from the training set. The chosen loss (usually called L2 loss) is given by:

$$L(W_{net}) = \sum_{t=T_{obs}+1}^{T_{pred}} \left\|\hat{Y}^t - Y^t\right\|_2^2 = \sum_{t=T_{obs}+1}^{T_{pred}} \left(\hat{Y}^t - Y^t\right)^2, \tag{3.9}$$

where W_{net} are the weights of the network that will be learned in the training process, $\hat{Y}^t$ and Y^t are respectively the predicted - obtained by applying (3.3) - and real positions at time t. The goal of the model will therefore be to generate trajectories that are on average as close to the GT as possible. The back-propagation and weight update is done in batches of trajectories from a training set, with a predetermined batch size. Moreover, a single model can be trained on multiple scenes.

3.2.4 Multimodal LSTM baseline: Generating multiple paths per pedestrian

Several works [6, 8, 9, 12, 13] have demonstrated the usefulness of exploring the multimodal nature of pedestrian motion. In general, a pedestrian may have more than one optimal path to choose from, and models that are able to output multiple trajectory samples for the same pedestrian are better at capturing this multitude of paths.

The LSTM network addressed throughout this section can be altered to sample several trajectories per pedestrian. The major change is in the output of the model: instead of outputting a sequence of predicted displacements, $\Delta\hat{Y}^t$, the model will output the parameters of a probability distribution.

The model will estimate the parameters of a Bi-Variate Gaussian distribution for the displacements (across x and y components of the 2D trajectory), similar to other works [11, 26, 33]. This consists of 5 parameters, estimated at each timestep: 2D mean, 2D standard deviation, and a correlation factor - $\hat{\mu}^{t+1} = \left[\hat{\mu}^t_{\Delta x}, \hat{\mu}^t_{\Delta y}\right]$, $\hat{\sigma}^{t+1} = \left[\hat{\sigma}^t_{\Delta y}, \hat{\sigma}^t_{\Delta y}\right]$, $\hat{\rho}^t$. The parameters are estimated via the previous LSTM state:

$$\left[\hat{\mu}^{t+1}, \hat{\sigma}^{t+1}, \hat{\rho}^{t+1}\right] = W_{out_{lin}} h^t + b_{out_{lin}}, \tag{3.10}$$

Note that this is analogous to (3.7), except that the weights and bias vectors will have more elements, due to the fact that there are more parameters to estimate at each time step.

Since this model is outputting a probabilistic distribution for the sake of capturing multimodality in the trajectories, using the same L2 loss in (3.9) would not model uncertainty. Thus, the training objective changes to minimize a probabilistic loss. Specifically, it will involve minimizing the following Negative Log-Likelihood (NLL) loss:

$$L(W_{net}) = -\sum_{t=T_{obs}+1}^{T_{pred}} \log\left(\mathbb{P}\left(\Delta Y^t \mid \hat{\mu}^t, \hat{\sigma}^t, \hat{\rho}^t\right)\right), \tag{3.11}$$

where W_{net} are the weights of the network that will be learned in the training process, and ΔY^t is the real relative displacement at the same instant (coming from the training set). Just like in section 3.2.3, the overall loss is averaged for all pedestrian trajectories, and this model can also be trained on data from different scenes.

It is worth remembering that the baseline LSTM model of this section does not consider any influencing cues (scene or social) to improve the predicted trajectories. The next few sections will detail the procedure for integrating the scene-specific information. Following that will be the interaction-aware extensions to the LSTM baseline.

3.3 Trajectory prediction using SMF

Unlike other works that consider scene-specific information [2, 10], SMF model does not require additional inputs like top-view images of the scene in order to learn scene-specific cues. It was the only work found that learns scene-specific cues using only trajectory data, *i.e.*, sequences of positions. This work has achieved competitive results when compared to methods that require extra information [9, 12]. This section will summarize the main aspects of the model, for the reader to better understand the integration of SMF with LSTM networks, in section 3.4.

Similarly to section 3.2, there is no notion of interacting pedestrians in this model, and so the i subscript in positions will also be dropped for this section. Motion fields are applied with the previous 2D position and output a 2D displacement. The following relation between the current position of a pedestrian, $X^t \in [0,1]^2$, and the previous one, $X^{t-1} \in [0,1]^2$ (note that here the positions are normalized, with $[0,1]^2$ corresponding to the image lattice), can be defined as:

$$X^t = X^{t-1} + \Gamma_{k^t}\left(X^{t-1}\right) + w^t, \quad t = T_{ini}+1, \ldots, T_{pred}, \tag{3.12}$$

where T_{ini} and T_{pred} are respectively the first and final time instants of the trajectory, $k^t \in \{1, \ldots, K\}$ (with K being total number of motion fields) identifies the active motion field $\Gamma_{k^t} : [0,1]^2 \to \mathbb{R}^2$ governing the displacement at current time step t, and $w^t \in \mathbb{R}^2$ is white noise, $w^t \sim N\left(0, \Sigma_{k^t}(X^t)\right)$, with the covariance matrix Σ_{k^t} depending on the 2D position of the pedestrian. There is a probability of switching from one motion field k^{t-1} to another k^t in two consecutive instants. The several transition probabilities between motion fields will be elements of a stochastic transition matrix, $B(X^t)$, evaluated at position X^t.

The motion fields, Γ_k, covariance matrices Σ_k, and transition matrix B are defined in a regular grid with N nodes. These parameters are estimated using the Expectation-Maximization (EM) [54] algorithm. The parameters also need to be estimated individually for each of the available scenes. The priors of the motion fields were subject to some restrictions, in order to make them smooth - penalize large changes between motion fields in neighbour nodes - and sparse - small displacement values for most motion fields in nodes of the grid. It is this consideration of sparsity that allows the SMF model to learn scene-specific information, since regions of a scene where no motion exists (due to there being obstacles or that region not being walkable) will have near-zero displacement values.

To use these parameters in the task of trajectory prediction, it is still necessary to decide the best sequence of motion fields to apply in (3.12) for each prediction instant. Each of the past positions X^t, $t \in \{T_{ini}, ..., T_{obs}\}$ has associated a vector of probabilities, $\alpha(t) \in \mathbb{R}^K$. Each of the elements of the α vector, identified by the motion field to apply $j \in \{1, ..., K\}$, is given by:

$$\alpha_j(t) = \mathrm{P}\left\{X^{T_{ini}}, \ldots, X^t, k^t = j \mid \hat{\theta}\right\}, \tag{3.13}$$

where $\hat{\theta}$ corresponds to the estimated parameters of the model. The actual vector of probabilities can be obtained through recursion, using $\alpha(t-1)$ and transition probabilities given by matrix $B(X^{t-1})$. The choice of motion field to apply at time step t is given by the motion field that has associated the highest probability: $k_t^* = \underset{k}{\operatorname{argmax}}\ \alpha(t)$.

In [6], there are two possible representations of the predicted 2D position. One corresponds to a deterministic version of (3.12), *i.e.*, without the inclusion of noise:

$$\hat{X}^t = \hat{X}^{t-1} + \Gamma_{k_t^*}\left(\hat{X}^{t-1}\right), \tag{3.14}$$

Alternatively, one can consider a stochastic representation for the predicted position, given by the following Gaussian distribution:

$$\hat{X}^t \sim \mathcal{N}\left(\hat{X}^{t-1} + \Gamma_{k_t^*}\left(\hat{X}^{t-1}\right), \Sigma_{k_t^*}\left(\hat{X}^{t-1}\right)\right), \tag{3.15}$$

With this representation, one can sample the predicted position several times, making it possible to generate several distinct trajectories for the same pedestrian, like the multimodal LSTM of section 3.2.4.

3.4 Trajectory prediction with LSTM and SMF

Human motion is governed by many different factors, most notably the physical aspects of each scene and the social interactions between pedestrians, none of which are being taken into consideration in the model described in section 3.2. This section will be about incorporating the scene-specific cues into LSTM. The proposed model will use not only displacements coming from data, but also displacements computed with SMF [6] - the ones added to the previous position in section 3.3 as shown in (3.12).

The goal of this integration is to improve the generated trajectories in accordance with the scene's geometry. Similar to section 3.2 and 3.3, the subscript i identifying the pedestrian will still be omitted.

3.4.1 LSTM-SMF-I: Supply the most likely motion field

This first extension involves feeding the network only the most likely displacement provided from the SMF model. The network learns to weigh both its own displacement and the one provided by the motion fields at the prediction phase. While it would be possible to obtain the motion fields displacements at the observation phase, it would not improve performance. It is preferable to have memory of the past using only the actual data.

Unlike the original LSTM baseline, there are differences in terms of the input and both observation and prediction phases. At observation phase, only a displacement coming from data, ΔX^t, is fed to the network. At prediction phase, two displacements will be fed. This leads to the conclusion that there is a benefit in having separate LSTM networks to handle observation and prediction phase. This type of configuration is commonly known as an Encoder-Decoder architecture, and has also been used in prior works for this task [8, 28, 43]. A simple diagram representing the global model to integrate motion fields predictions can be seen in fig. 3.5. The encoder will be in charge of the observation phase, sending information that the decoder will use to generate the future trajectory, in the prediction phase.

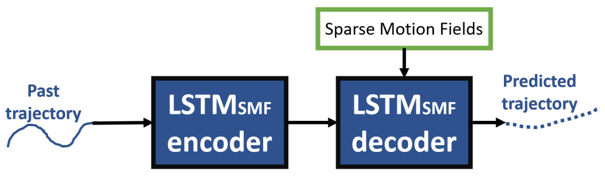


Figure 3.5: Overall LSTM encoder-decoder architecture for trajectory forecast, integrating with SMF model. There are two separate LSTMs - the encoder will be used in the observation phase, to process the past trajectory, similarly to what was described in section 3.2.1; the decoder will use and weigh future predictions of its own and motion fields predictions to generate trajectories that are more scene-compliant.

The observation phase is similar to the same phase of the baseline model, described in section 3.2.1. The only difference is that in this case, the LSTM that handles the observation phase (encoder) is different that the one that handles the prediction phase (decoder). The output of this phase will still be a state containing a summary of the past, $h^{T_{obs}}$, that will be used for the prediction phase. This state is the knowledge that the encoder passes to the decoder. In some prior works [8, 14], this knowledge may not represent the state directly. However, for this model, the encoder and decoder LSTM cells will have identical characteristics (although separate parameters), and so the state can be directly used as input.

In the prediction phase, the LSTM decoder - named $LSTM_d$ - will receive and use the two predicted displacements, along with the previous state, to output a new state. This new state can be used to obtain a new predicted displacement, using (3.7). The prediction phase of the model is depicted in fig. 3.6.

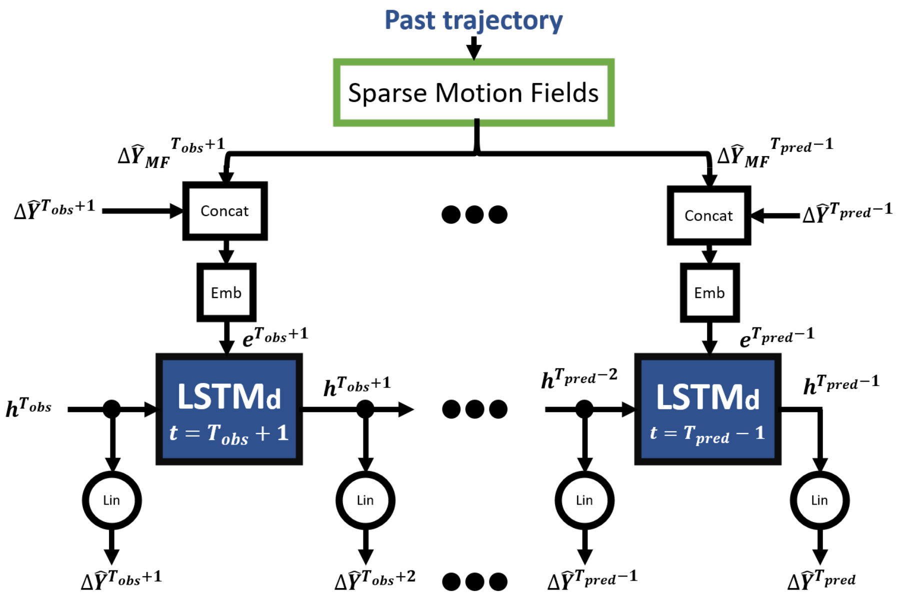

Past trajectory
Sparse Motion Fields
$\Delta\hat{Y}_{MF}^{T_{obs}+1}$
$\Delta\hat{Y}_{MF}^{T_{pred}-1}$
$\Delta\hat{Y}^{T_{obs}+1}$
Concat
Concat
$\Delta\hat{Y}^{T_{pred}-1}$
Emb
Emb
$e^{T_{obs}+1}$
$e^{T_{pred}-1}$
$h^{T_{obs}}$
LSTMd
$t = T_{obs}+1$
$h^{T_{obs}+1}$
$h^{T_{pred}-2}$
LSTMd
$t = T_{pred}-1$
$h^{T_{pred}-1}$
Lin
Lin
Lin
Lin
$\Delta\hat{Y}^{T_{obs}+1}$
$\Delta\hat{Y}^{T_{obs}+2}$
$\Delta\hat{Y}^{T_{pred}-1}$
$\Delta\hat{Y}^{T_{pred}}$

Figure 3.6: The prediction phase of the model, done via LSTM decoder, where at each instant both the previous LSTM and SMF predictions (where the latter only uses the past trajectory) are fed into the cell, and together they are used to obtain the predicted displacement for the next instant.

The first predicted displacement from the SMF method is obtained by processing the entire past trajectory. Only the most likely of K motion fields is chosen, applying it to the last observed position. This displacement, $\Delta\hat{Y}_{MF}^{T_{obs+1}} = \Gamma_{k^*_{T_{obs}+1}}\left(X^{T_{obs}}\right)$, as well as all other future motion fields displacements $\Delta\hat{Y}_{MF}^{t+1} = \Gamma_{k^*_t}\left(\hat{Y}_{MF}^t\right)$ (note that SMF uses the previous SMF prediction; it does not receive any LSTM prediction) for $t \in \{T_{obs}+1, T_{pred}-2\}$ are concatenated with the predicted LSTM displacement at the same time instant, $\Delta\hat{Y}^t$. Both of these are fed to a new embedding layer:

$$\begin{aligned} e^t &= Emb\left(\Delta\hat{Y}^t, \Delta\hat{Y}_{MF}^t; W_{emb_d}\right) \\ &= PReLU\left(W_{emb_{lin_d}}\left[\Delta\hat{Y}^t, \Delta\hat{Y}_{MF}^t\right]^T + b_{emb_{lin_d}}\right), \end{aligned} \tag{3.16}$$

where W_{emb_d} corresponds to the parameters of this embedding layer, specific to the decoder. The actual predicted displacements from SMF, come in a normalized interval $[0, 1]$. As such, they first must be converted to the correct units being employed (metric, pixel, *etc.*) before they are able to be sent as input to the embedding layer in (3.16).

The embedded vector, e^t is the direct input to the LSTM cell, fed in similar fashion to (3.6), but with LSTM parameters specific to the decoder:

$$h^t = LSTM_d\left(h^{t-1}, e^t; W_{LSTM_d}\right), \tag{3.17}$$

The model outputs a set of displacements $\Delta\hat{Y}^t$, for $t \in \{T_{obs}+1, T_{pred}\}$, obtained via a linear layer:

$$\Delta\hat{Y}^{t+1} = W_{out_{lin}} h^t + b_{out_{lin}}, \tag{3.18}$$

with $W_{out_{lin}}$ and $b_{out_{lin}}$ being the weights and biases of the linear layer, present only in the decoder. The actual trajectory can be obtained from the displacements, via (3.3).

3.4.2 LSTM-SMF-II: LSTM integrates the several motion fields displacements

The first extension can result in some improvements over the base LSTM model. However, the LSTM only has access to what the SMF model deems to be the most likely prediction. There may be cases where it would be best to apply a different motion field, or even a combination of multiple fields. This drops the assumption made in [6] - that at a given instant, the trajectory of a pedestrian is governed by a single motion regime.

The SMF prediction is also completely independent of the LSTM model, and in some cases that can lead to a larger trajectory error when compared to the real trajectory. It can be useful to feed the motion fields model with the prior LSTM prediction, so it can better adjust its position. This calls for a second version for integrating SMF with LSTM: to let the network learn what motion field to apply at each time step, with the possibility of choosing more than one motion field for the same instant, with different contributions.

This new variant still has some similarities with the one described in the previous section. First, it maintains the Encoder-Decoder architecture. Second, the processing of the past trajectory (observation phase) by the encoder is the same.

The differences between this new variant and the previous one are all in the prediction phase. The differences are highlighted in fig. 3.7, with a different colour (compare with fig. 3.6). As seen in fig. 3.7, there are two separate layers to embed the input that is fed to the LSTM decoder cells. One, Emb_L, is responsible for embedding the prior prediction of the LSTM network. The other, Emb_{MF}, embeds the K different displacement predictions that the SMF model generates. The embeddings are given by:

$$e_L^t = Emb_L\left(\Delta\hat{Y}^t; W_{emb_L}\right), \tag{3.19}$$

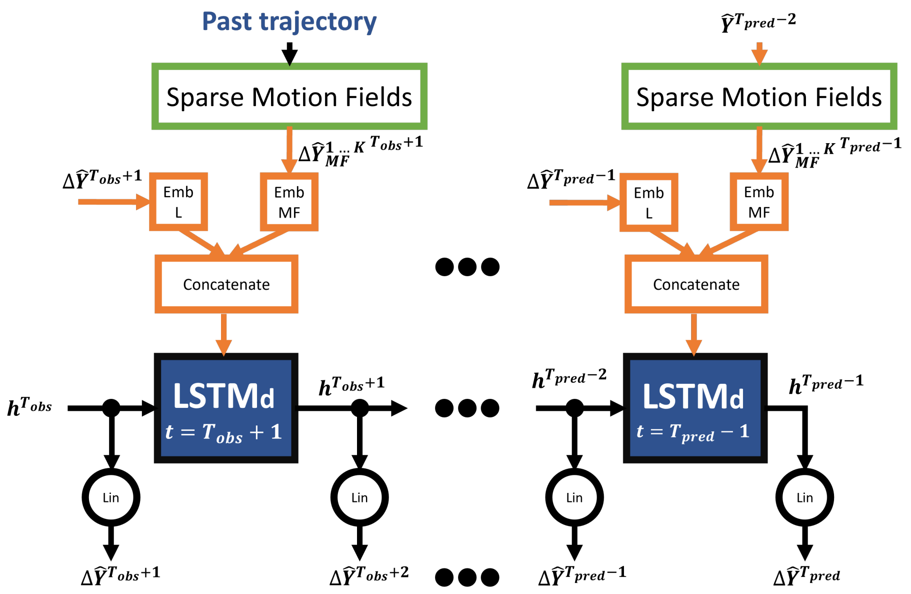

Past trajectory
Sparse Motion Fields
$\Delta\hat{Y}^{1\dots K}_{MF}{}^{T_{obs}+1}$
$\Delta\hat{Y}^{T_{obs}+1}$
Emb L
Emb MF
Concatenate
LSTMd
$t = T_{obs}+1$
$h^{T_{obs}}$
$h^{T_{obs}+1}$
Lin
$\Delta\hat{Y}^{T_{obs}+1}$
$\Delta\hat{Y}^{T_{obs}+2}$
$\hat{Y}^{T_{pred}-2}$
Sparse Motion Fields
$\Delta\hat{Y}^{1\dots K}_{MF}{}^{T_{pred}-1}$
$\Delta\hat{Y}^{T_{pred}-1}$
Emb L
Emb MF
Concatenate
LSTMd
$t = T_{pred}-1$
$h^{T_{pred}-2}$
$h^{T_{pred}-1}$
Lin
$\Delta\hat{Y}^{T_{pred}-1}$
$\Delta\hat{Y}^{T_{pred}}$

Figure 3.7: The prediction phase of the new variant that integrates SMF and LSTM for scene-compliant trajectory forecasting. The differences from the previous model are shown in orange. There are two embedding layers, to process the predictions of LSTM and motion fields separately. Also, the SMF model receives the LSTM prediction instead of using its own.

$$e^t_{MF} = Emb_{MF}\left(\Delta\hat{Y}^1_{MF}{}^t, \dots, \Delta\hat{Y}^K_{MF}{}^t; W_{emb_{MF}}\right), \tag{3.20}$$

where e^t_L and e^t_{MF} are have the same size, equal to half the input size expected by the LSTM cell. This is mostly equivalent to saying the K displacements of SMF method are given the same initial relevance as the displacement obtained from the LSTM. The Emb_{MF} layer will "choose" the motion field(s) that are active at each instant, and also giving a contribution to each of the displacements via its parameters, $W_{emb_{MF}}$. Each embedding consists in a linear layer with *PReLU* activation function, just like in (3.16).

The two embeddings will be concatenated and used as the direct input to the LSTM decoder cell:

$$h^t = LSTM_d\left(h^{t-1}, \left[e^t_L, e_{MF}{}^t\right]; W_{LSTM_d}\right), \tag{3.21}$$

where h^t will still be used to obtained the next predicted displacement, $\hat{Y}^{t+1}$, using (3.18). The SMF model also receives the LSTM predicted position using (3.3), instead of using its own to perform the predictions at the next instant - $\Delta\hat{Y}^k_{MF}{}^{t+1} = \Gamma_k\left(\hat{Y}^t\right)$ for $k \in \{1, ..., K\}$ and $t \in \{T_{obs}+1, T_{pred}-2\}$.

3.4.3 Training

Both variants of LSTM incorporating SMF for trajectory forecast are trained with the same L2 loss used by the baseline model, in (3.9). That is because the goal is still for these models to generate trajectories compliant with the past and as close to the real trajectories as possible.

It is worth noting that the SMF model needs to be trained first so they can be used on the proposed LSTM models. The SMF model needs to be trained per scene, so that scene-specific elements can be learned. As such, there will be need to have training (to learn context of the scene) and testing data (to evaluate the quality of what was learn on unseen data) for each scene.

The LSTM models described in this section, as well as the baseline in section 3.2, can be trained on multiple scenes. For the models of this section, there would be the need to have several SMF models, equal to the number of distinct scenes, and the LSTM model would need to know to which scene does the trajectory belong to, so it knows what SMF model to apply.

While not strictly necessary, it was chosen to also train one LSTM model per scene, having a $1:1$ ratio in terms of number of LSTM and SMF models. This enforces the learning of application of the motion fields to be scene-specific, improving the performance on each of the scenes at test time.

3.5 Trajectory prediction with interaction-aware LSTM

Pedestrian trajectory forecast with the incorporation of social interactions is still an open problem. This is evidenced by the large body of works released in recent years (refer to Section 2.2). There are many kinds of possible interactions between pedestrians and/or groups of pedestrians. Some examples of possible interactions are shown in fig. 3.8. Creating a model that can robustly deal with the variety of interactions is one of the biggest challenges associated with the task of trajectory prediction.

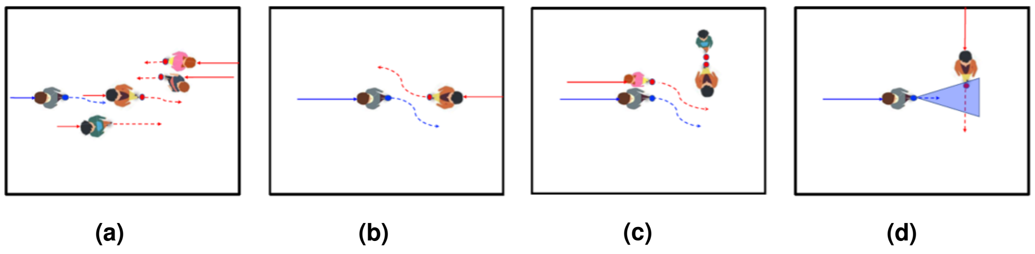


Figure 3.8: Examples of possible interactions that pedestrians may have. **(a)** Leader Follower: A pedestrian following another that is in front **(b)** Collision Avoidance: Two pedestrians adjusting course to avoid clashing with one another **(c)** Group travelling: Two or more pedestrians travelling very close to each other, and in the same direction. **(d)** Other, more uncommon forms of interaction. Credits to the images go to [26].

A summary of the proposed model for interaction-aware trajectory forecast, which is equivalent - at a higher level - to architectures of prior works [8,26,27] is available in fig. 3.9. Similarly to the scene-aware model of section 3.4, it adopts an encoder-decoder architecture, where the encoder processes the past trajectory ($t \in \{T_{ini}, ..., T_{obs}\}$), and the decoder outputs the predicted trajectory ($t \in \{T_{obs}+1, ..., T_{pred}\}$). The addition is an extra interaction layer that is placed through consecutive instants of LSTM cells, be it of encoder or decoder. The interaction layer, to be discussed further in this section, will generate a summary of the social context surrounding each pedestrian.

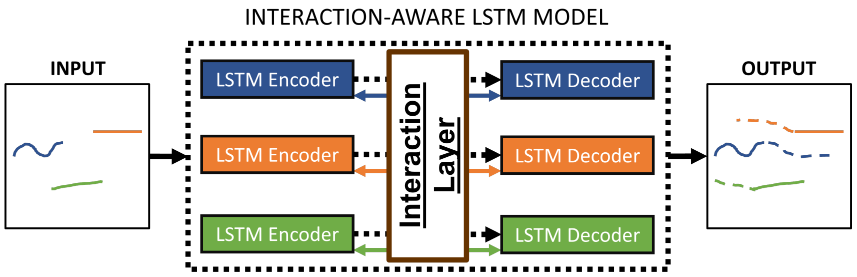


Figure 3.9: Overall LSTM encoder-decoder architecture for interaction-aware trajectory forecast. The input of the model is no longer a single trajectory, but a set of trajectories for all pedestrians simultaneously present in a scene. Each pedestrian's motion is processed via a separate LSTM, but the the cells of the encoder have the same weights for all pedestrians (same for decoder). The social context surrounding each pedestrian is obtained every instant via an interaction layer.

As mentioned at the start of this chapter, and similarly to what was discussed in section 3.4, the proposed interaction-aware model will be based on an existing work introduced in chapter 2: Directional LSTM [26]. This LSTM model considers the presence and influence of neighbouring pedestrians via a grid-shaped pooling technique. Every instant, the contribution of each neighbour inside the grid will be the relative velocity when compared to the pedestrian in question. Neighbours that are too far away to be inside the grid will not influence the motion of the pedestrian. The output of this pooling is a $N_g \times N_g \times 2$ tensor for each pedestrian, where N_g is the number of rows of the grid - or columns, since the grid is square -, and 2 symbolizes the relative velocity, which is in $\mathbb{R}^2$. An alternative basis to use could be Social LSTM [11], which uses the neighbouring pedestrians LSTM states. There are two reasons that lead to picking Directional LSTM instead of Social LSTM:

1. Having the relative velocities of neighbours is easier to interpret than the LSTM hidden state, since the latter is an encoded summary of the pedestrians past.
2. It is more efficient to process a tensor of relative velocities, of size $N_g \times N_g \times 2$, then of size $N_g \times N_g \times N_h$, where N_h is the dimension of the hidden state, which in most cases is at least an order of magnitude higher than $\mathbb{R}^2$ [26].

The usage of a grid-based pooling can have some limitations. One, already mentioned in section 2.2, lies with the fact that the dimensions of the grid (N_g and length of the grid) are fixed for all pedestrians. Another issue is that the grid, being centered on the pedestrian, can also include neighbours behind the pedestrian. Those neighbours will have little importance when compared to the neighbours in front, whom the person may be following (fig. 3.8(a)) or may have to avoid colliding with (fig. 3.8(b)).

The following sections will detail a new LSTM based model that considers the relative motion of neighbouring pedestrians, while also attempting to combat the limitations of a fixed grid shape. The first limitation to tackle will be the consideration of neighbours behind the pedestrian, by simulating a Field of View (FOV) for each pedestrian. After, multiple techniques will be presented as possible options to vary the dimensions of the FOV, as a function of the social context surrounding the pedestrian. Since the models in this section process the trajectories of multiple pedestrians simultaneously, the subscript $i = \{1, ..., N\}$ (N is the current number of pedestrians present in the scene) identifying a pedestrian will be added to the notation (equivalent to the notation of section 1.2).

3.5.1 Arc-LSTM: Improving Directional LSTM by using a FOV instead of a grid

Using a grid-shaped tensor to include the influence of neighbours in the pedestrian's motion is not the most intuitive way of making a model socially-aware. Humans do not have a structured knowledge of what is behind them. While it can be argued that one must have some implicit knowledge of people immediately behind, most of a pedestrian's focus will lie in what is in front, *i.e.*, what the person can see.

This notion of focusing on what the pedestrian can see has been approached on prior works [17, 21, 65]. It is also what inspires the following improvement to Directional LSTM: Instead of considering a grid centered on the pedestrian, the proposed interaction-aware model will use an arc shape. The arc can be thought of as the FOV of the pedestrian, approximating what the pedestrian is seeing, or even a Field of Attention (FOA), to consider what the person is paying attention to at a particular moment. The terms arc, FOV and FOA may be used interchangeably throughout this section.

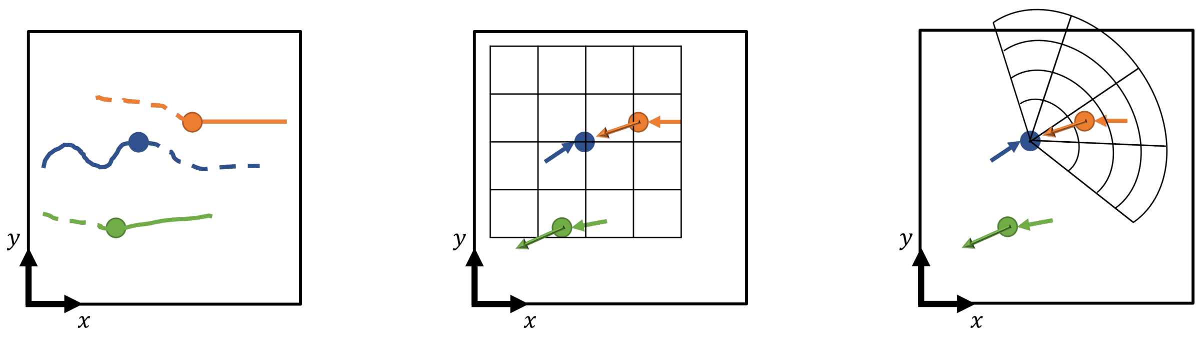
A square grid can be characterized by 2 parameters - the length of the side l and the number of nodes in a row N_g. In contrast, an arc shape requires 4 parameters - its radius, r; its width or angle, α; the number of cells (or divisions) across the radius, N_r, and across the angle N_α. The extra parameters make an arc shape a more flexible choice to use in a pooling technique. The parameters r and α make up the size of the region that the pedestrian is seeing and/or focusing on. A comparison between the use of grid and arc shapes for interaction-specific pooling can be seen in fig. 3.10.

For this arc to resemble an actual FOV, its orientation should be given by the pedestrian's gaze direction. However, as stated on the problem formulation of section 1.2, the proposed models only have access to the actual trajectories of pedestrians, in $\mathbb{R}^2$. To circumvent this, it will be approximated by the direction of motion:

$$\theta_i^t \approx \arctan\left(\frac{\Delta y_i^t}{\Delta x_i^t}\right) \tag{3.22}$$

where θ_i^t is the angle of the gaze direction, in the scene's coordinate system, and Δy_i^t and Δx_i^t are the current displacement components along the y and x axis, respectively, for pedestrian i and time t. It is worth reinforcing that this is an approximation - there may be some cases where the pedestrian is moving in one direction and its gaze may be directed in a considerably different direction. In many cases, however, a pedestrian can be focusing on the direction of motion, even if not directly gazing at it.

The tensor given by the arc-shaped directional pooling, for a certain pedestrian i with neighbourhood



(a) (b) (c)

Figure 3.10: Comparing directional pooling (originally from [26]) using different shapes. **(a)** Trajectories of 3 pedestrians, with full line representing the past and dashed representing the future. The pooling will be focused on the blue pedestrian. **(b)** The grid shape considers a pedestrian (in green) that is behind and moving away from the blue pedestrian. **(c)** The arc based shape only includes the orange pedestrian, that the blue pedestrian is most likely focusing on.

set $\mathcal{N}_i$ can be formalized as:

$$A_i^t(m_r, n_\alpha, :) = \frac{1}{\sum_{j\in\mathcal{N}_i} \mathbf{1}_{m_r,n_\alpha}\left[d_{ij}^t, \beta_{ij}^t\right]} \sum_{j\in\mathcal{N}_i} \mathbf{1}_{m_r,n_\alpha}\left[d_{ij}^t, \beta_{ij}^t\right]\left(\Delta X_j^t - \Delta X_i^t\right) \tag{3.23}$$

where A_i^t is the $N_r \times N_\alpha \times 2$ tensor containing the relative velocities of neighbours in the FOV of the pedestrian. The difference of displacements $\left(\Delta X_j^t - \Delta X_i^t\right)$ is used in place of the actual relative velocity (as in section 3.2 and 3.4). The distance between a pedestrian and its neighbour, $d_{ij}^t = \sqrt{(x_j^t - x_i^t)^2 + (y_j^t - y_i^t)^2}$, as well as their relative orientation in reference to the pedestrian's gaze, $\beta_{ij}^t = \alpha_{ij}^t - \theta_i^t = \arctan\left(\frac{y_j^t - y_i^t}{x_j^t - x_i^t}\right) - \theta_i^t$, are used in indicator function $\mathbf{1}_{m_r,n_\alpha}[d_{ij}^t, \beta_{ij}^t]$. This function checks if d_{ij}^t and β_{ij}^t are such that the neighbour is inside the (m_r, n_α) cell of the arc shape, with $m_r \in \{1, ..., N_r\}$ and $n_\alpha \in \{1, ..., N_\alpha\}$. Formally, the indicator function is given by (3.24):

$$\mathbf{1}_{m_r,n_\alpha}\left[d, \beta\right] = \begin{cases} 1 & \text{, if } \frac{m_r - 1}{N_r} r \leq d < \frac{m_r}{N_r} r \bigwedge \left(\frac{n_\alpha - 1}{N_\alpha} - \frac{1}{2}\right)\alpha \leq \beta < \left(\frac{n_\alpha}{N_\alpha} - \frac{1}{2}\right)\alpha \\ 0 & \text{, otherwise} \end{cases} \tag{3.24}$$

with r and $0 < \alpha \leq 2\pi$ being the whole radius and angle of the arc, and $-\pi \leq \beta < \pi$. In the example of fig. 3.10(c), the neighbour in orange (the only one inside the arc) is inside the cell with $m_r = 2$ and $m_\alpha = 3$. Below are some examples of common social interactions and their translation in relative velocities, to better understand the directional pooling layer and some underlying details:

- **Two pedestrians walking in opposite directions**: similar situation to fig. 3.10(c), and also in fig. 3.8(b). Since the directions are opposite, the relative velocity is high. The directional pooling will highlight these cases, making it useful for the model to learn to avoid collisions.

- **Two pedestrians walking in the same direction with similar speeds**: common in a leader-follower scenario (fig. 3.8(a)). The directional pooling initializes all cells with zeroes. The relative velocities in this case are small, meaning there is not much difference between an empty cell and having a neighbour with identical motion. Therefore, this model prioritizes collision avoidance situations over leader-follower or group behaviour (fig. 3.8(c)).

- **A pedestrian with two close neighbours going towards the person**: it is quite possible in crowded scenes to have neighbours so close to each other that they are in the same cell of the arc. The original grid pooling [26] discarded all neighbours in the same cell except for a randomly selected one[3]. The final A_i^t will have a mean of all relative velocities in the same cell, as seen in (3.23). An alternative could be simply summing, but for high relative velocities, it could be confused as a pedestrian moving very fast, rather than a group moving at normal speed.

- **A pedestrian with two close neighbours that have opposite relative velocities**: also possible in crowded scenes, although less common. Summing or averaging the relative velocities in the same cell cancels the contributions of the two neighbours. This is one case where picking only one of the neighbours could be a beneficial. However, this limitation, as well as the above one, can be tackled by having accurate dimensions for the FOV, or by having a variable FOV as a function of the current crowd density - a topic that will be addressed in section 3.5.2.

These examples have highlighted some strengths and limitations of pooling with use of directional velocities, but overall it is a compact and effective way of storing the social context for each pedestrian.

This LSTM model with arc-shaped directional pooling to include the influence of pedestrians on each other, denoted Arc-LSTM, will process each pedestrian's motion individually. It will now have a second kind of input, derived from the original past trajectories, which is the set of pooled tensors containing the neighbours' relative velocities. A detailed scheme of the Arc-LSTM can be found in fig. 3.11. It concerns the processing of trajectories from a set of N interacting pedestrians.

Both the pedestrian displacement and their arc pooling go through an embedding layer (not present in fig. 3.11), which similarly to (3.4), consist of a linear layer with a $PReLu$ activation function:

$$e_i^t = Emb\left(\Delta X_i^t; W_{emb}\right), \tag{3.25}$$

$$p_i^t = Emb\left(A_i^t; W_{emb_A}\right), \tag{3.26}$$

where the embedded inputs e_i^t and p_i^t, for each pedestrian i at instant t, will be computed using the predicted displacements $\Delta\hat{Y}_i^t$, for $t > T_{obs}$. The arc pooling will be computed with the predicted positions

[3]While not explicitly discussed in the paper, it is a detail that can be observed in their publicly available implementation

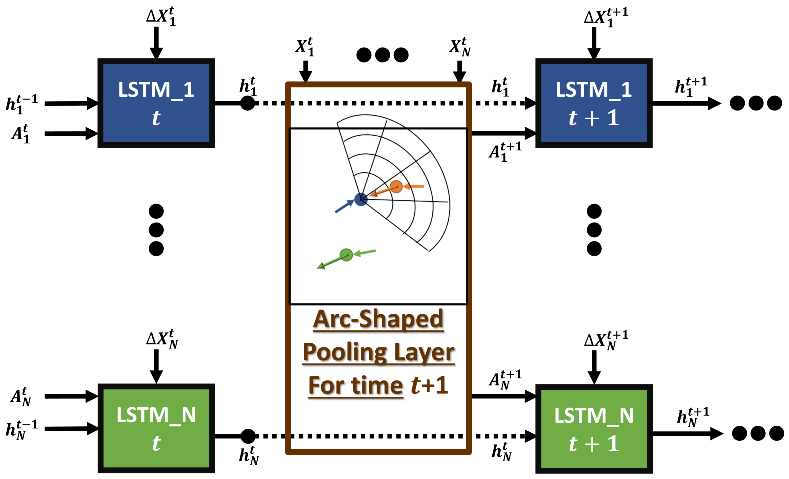


Figure 3.11: The Arc-LSTM model processing all N pedestrians' motion and relevant interactions for two consecutive instants. There are multiple cells stacked vertically, with different colours distinguishing different pedestrians. The LSTM cells receive the hidden state at the previous time step, as well as the prior motion and arc-shaped directional pooling, to compute the hidden state at the next time-step.

for $t > T_{obs}$. The embeddings e_i^t and p_i^t are fed to the LSTM cells, of both the encoder and decoder:

$$\begin{aligned} h_i^t &= LSTM_e\left(h_i^{t-1}, \left[e_i^t, p_i^t\right]; W_{LSTM_e}\right), \quad t \in \{T_{ini}, T_{ini}+1, ..., T_{obs}\} \\ h_i^t &= LSTM_d\left(h_i^{t-1}, \left[e_i^t, p_i^t\right]; W_{LSTM_d}\right), \quad t \in \{T_{obs}+1, T_{obs}+2, ..., T_{pred}\}, \end{aligned} \tag{3.27}$$

where the encoder and decoder parameters, W_{LSTM_e} and W_{LSTM_d}, have the same dimensions.

The generation of a predicted trajectory is going to be slightly different from the scene-compliant model of section 3.4. Instead of outputting a single displacement per pedestrian per instant, the model will generate parameters of a bi-variate Gaussian distribution for the displacement (as in section 3.2.4):

$$\left[\hat{\mu}_i^{t+1}, \hat{\sigma}_i^{t+1}, \hat{\rho}_i^{t+1}\right] = W_{out_{lin}} h_i^t + b_{out_{lin}}, \tag{3.28}$$

where $[\mu_i^t, \sigma_i^t, \rho_i^t] \in \mathbb{R}^5$ contains a mean, $\mu_i^t \in \mathbb{R}^2$, 2D standard deviation, $\sigma_i^t \in \mathbb{R}^2$, and correlation factor $\rho_i^t \in \mathbb{R}$ for the displacement of pedestrian i at time t. This was already employed in a multi-modal version of the baseline LSTM, in section 3.2.4. It is important to note that this does not force the model to generate more than one trajectory sample per pedestrian - one can discard the uncertainty at inference time and just use the mean displacement, $\mu_i^t \equiv \Delta\hat{Y}_i^t$. The mean displacement is fed as input to the LSTM decoder for $t \in \{T_{obs}+1, ..., T_{pred}\}$, and can be used to build the single trajectory with (3.3).

The explicit consideration of uncertainty helps the model to generate more socially acceptable trajectories [11,26,33]. If this model were trained with the L2 loss of eq. (3.9), the predictions would be forced to "follow" the average behaviour of the GT, without considering the possibility of straying from it to, *e.g.*, avoid colliding with another pedestrian. To train the the model - W_{net} - with the use of uncertainty, a NLL loss (identical to section 3.2.4) is used. This loss will also be used in the models of the next section.

3.5.2 V-Arc-LSTM: Variable arc size in accordance to the social context

The model from section 3.5.1 tackles one of the limitations found with grid-based interaction pooling - the consideration of neighbours behind the pedestrian, that in many cases do not influence the pedestrian's motion. The second limitation found is more challenging to tackle - the same FOV dimensions (or grid in the case of [26]) are used for all pedestrians.

The social environment may vary from scene to scene, and even in the same scene across time instants. Different social settings are shown in fig. 3.12, to motivate the use of variable FOV. In fig. 3.12(a), a pedestrian benefits from having a smaller FOV (or FOA), especially when compared with fig. 3.12(b). This conclusion can mainly be drawn from the difference in crowd density - the overall number of pedestrians for a certain area - in the two scenes. Since there is a higher number of neighbours close to the pedestrian in fig. 3.12(a), the focus and view of the pedestrian will be only for those in close proximity. In contrast, the pedestrian in fig. 3.12(b) has less neighbours, and is able to have a larger FOV to consider neighbours that are farther away and that can alter the person's trajectory.

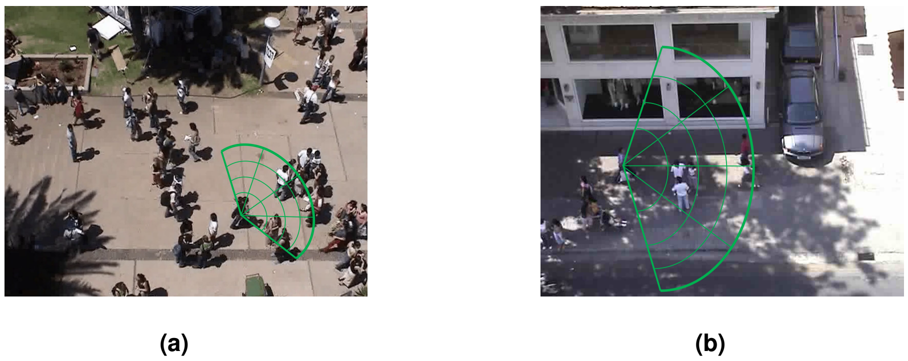


Figure 3.12: Examples from a real pedestrian trajectories dataset [20] to illustrate the motivation behind a variable FOV for interaction pooling. **(a)** In a scene with a relatively high crowd density. A pedestrian will, on average, focus on the neighbours that are closer, especially since those farther away may not even be visible. **(b)** In a scene with smaller number of pedestrians, the FOV can be bigger to include neighbours not exactly in close proximity, but that may influence the pedestrian's motion.

This section will provide several hypotheses to consider a variable size arc (or FOV) pooling shape. These hypotheses go from simple heuristics based on social characteristics taken from the set of pedestrain trajectories, to a data-driven model that learns to configure the size of the FOVs. Note that more hypotheses were thought of and implemented, but this section will feature only the most relevant.

3.5.2.A V-Jn-Arc-LSTM: Variable arc radius to include up to J neighbours

When in crowded scenarios, there is a limited number of things that a pedestrian can focus on. When adjusting his/her trajectory, the pedestrian's focus may lie on a subset of neighbours from the whole set of visible neighbours. This model will consider an arc pooling shape that only considers the J closest neighbours to the pedestrian. Note that this is different from a concatenation strategy such as that of [13], since the concatenation is done with each relative neighbour motion, while the method proposed here uses an actual arc shape that will also include empty areas (without any neighbours).

The radius r_i^t of the arc will grow or shrink to fit J neighbours (can change for each pedestrian i and every instant t), while the angle α will remain fixed. A strategy could also be devised to vary the angle of the arc (*e.g.*, smaller in more dense areas), but the former concept is simpler and more intuitive. Neighbours that are not included in the arc of spread α will never be used in the interaction pooling. An example of how this procedure works for different social situations is shown in fig. 3.13. The pedestrian will focus on those closer in crowded situations (fig. 3.13(a)), but in sparse situations (fig. 3.13(b)), the arc's radius may grow to include pedestrians farther away.

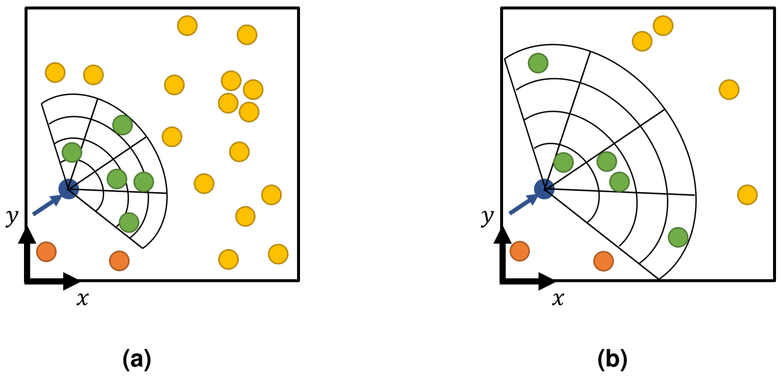


Figure 3.13: V-Jn-Arc-LSTM model includes at most J=5 neighbours in its arc pooling shape. Pedestrian in question in blue. Neighbours inside arc in green. Neighbours that could be inside arc (if radius r was larger) in yellow. Neighbours that could never be in arc (outside spread α) in orange. **(a)** Crowded scenes will have, generally, neighbours closer to the pedestrian. That will result in a smaller arc radius. **(b)** Less crowded situations will have more liberty for the radius, since the J neighbours can be farther away.

To make sure the arc radius is between reasonable values, a minimum value, r_{min}, and maximum value, r_{max}, are set. For instance, if there are less than $k < X$ neighbours in the arc of spread α and radius r_{max}, the pooling arc will only include those k neighbours. The rest of this interaction-aware model is exactly as the fixed arc shape model described in section 3.5.1.

3.5.2.B V-ND-Arc-LSTM: Arc radius that depends on the mean distance of neighbours

Considering that a pedestrian gives importance to a fixed number of neighbours can have its limitations. Depending on the social context, there may be more (or less) neighbours that are influencing the trajectory of the pedestrian. An example of that is shown in fig. 3.14. If the arc shape were to include J=5 neighbours (fig. 3.14(a)), it would consider neighbours that are very far away from the pedestrian, and unlikely to be in the actual FOA of the pedestrian. A better alternative would be a FOA that varies with the mean distance of the neighbours in front of that pedestrian. Those neighbours would be apart of an arc with angle α and infinite radius. For the case of fig. 3.14(b), the arc would not include the distant neighbours as in fig. 3.14(a), but they would still contribute to the mean distance.

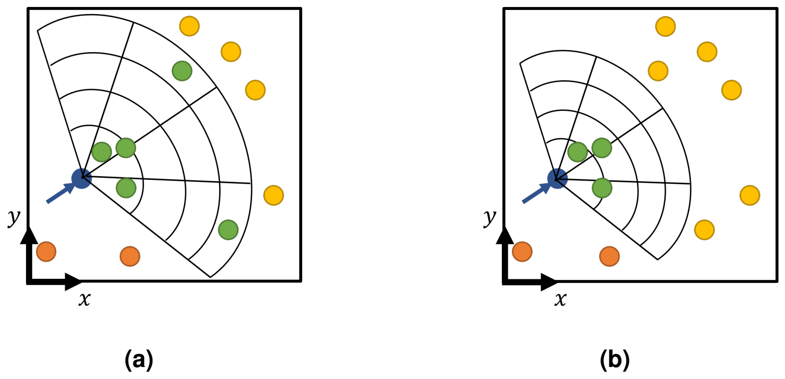


Figure 3.14: Comparing V-Jn-Arc-LSTM (J=5) and V-ND-Arc-LSTM. Neighbours inside arc in green. Neighbours that could be inside arc in yellow. Neighbours that could never be in arc in orange. **(a)** For V-Jn-Arc-LSTM, some of the neighbours that may not pose much influence to the pedestrian are included in the arc. **(b)** V-ND-Arc-LSTM, the radius is proportional to the mean distance of neighbours (in green and yellow). The arc radius is smaller because a subset of neighbours is close to the pedestrian.

Analytically, the radius of the arc for pedestrian i at time step t is computed via (3.29):

$$r_i^t = \left(C \frac{1}{\sum_{j \in \mathcal{N}_i} \mathbf{1}_\alpha \left[\beta_{ij}^t\right]} \sum_{j \in \mathcal{N}_i} \mathbf{1}_\alpha \left[\beta_{ij}^t\right] \left\| X_j^t - X_i^t \right\|_2 + B \right) \Bigg|_{[r_{min}, r_{max}]} \tag{3.29}$$

where $\mathbf{1}_\alpha \left[\beta_{ij}^t\right]$ is an indicator function that returns 1 if the neighbour j is inside the arc of spread α and infinite radius, and 0 if not. Visually, as represented in fig. 3.14, returning 1 means the neighbour is green or yellow, and 0 means the neighbour is orange. The neighbours will only contribute to varying the arc radius if the indicator function returns 1. C and B are coefficients of an affine transformation, which allows the mapping from the mean distance of neighbours to arc radius to be more flexible, instead of a direct 1:1 mapping to the $[r_{min}, r_{max}]$ interval.

Similarly to the model of the previous section, an arc radius based on the distance of neighbours can also have limitations. In certain situations, some neighbours that are farther away may still pose influence [66]. The pedestrian's FOA may grow to consider more distant neighbours if, for instance, some are moving fast towards the pedestrian. A pedestrian can give more importance to the relative motion of the surrounding neighbours than the relative distance. This concept was experimented with to create different models to vary each pedestrian's arc dimensions. The reason such models are not detailed in this chapter is that they still suffer from a lack of generalization to the several kinds of motion regimes and interactions between pedestrians. The next model is a proposition to combat this.

3.5.2.C V-LSTM-Arc-LSTM: Variable arc shape via LSTM network

The previous solutions consisted in simple ways to vary the arc pooling for each pedestrian, given some characteristics of the neighbourhood and the underlying social context. This solution goes beyond that - choose the dimensions of the arc shape with a separate ANN. This network, to be referred as LSTM$_A$, will sequentially process the arc dimensions that are being employed. The V-LSTM-Arc-LSTM will have a fixed arc angle, with the radius able to grow or shrink according to the social context. The purpose of LSTM$_A$ will be to decide, at every instant, what is the best radius of the FOV for each pedestrian. The scheme of V-LSTM-Arc-LSTM can be found in fig. 3.15. It corresponds to the same Arc-LSTM model in fig. 3.11, but extended with the new LSTM$_A$ (in orange), which will be detailed below.

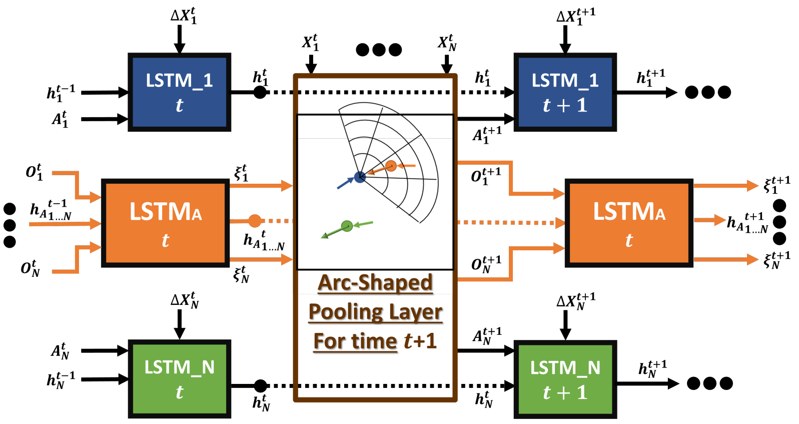


Figure 3.15: The V-LSTM-Arc-LSTM model. It is an extension to Arc-LSTM, with the new content shown in orange. The new network, LSTM$_A$, predicts the most likely radius for each of the pedestrians. Such prediction will be done using a simplified social context O_i^t. The predicted radii will be used to configure the size of the arc pooling layer in the next time step.

The predicted radius could be any value in a reasonable interval $[r_{min}, r_{max}]$, exactly like in sections 3.5.2.A and 3.5.2.B. However, the approach taken with this model is going to differ: the output will be a radius from a discrete set of Q possible values $\mathcal{R} = \{\varrho_1, \varrho_2, ..., \varrho_Q\}$. The reason for that lies in the limitations of the available data. Throughout this chapter, there has been access to past data - pedestrian displacements ΔX_i^t and relative displacements of neighbours. This does not apply to predicting the best FOV dimensions - there is no past data indicating what was the FOV of each pedestrian i at each time step t. Therefore, the subtask of choosing the most appropriate arc dimensions is weakly supervised [67]. Such tasks are generally more difficult than supervised ones, hence the simplified approach of discretizing the possible values for the arc radius.

The LSTM$_A$ network will not have direct access to the radius values from set $\mathcal{R}$. Instead, it will use an index $\xi_i^t \in \{1, ..., Q\}$, which identifies the radius from set $\mathcal{R}$, $r_i^t \equiv \varrho_{\xi_i^t}$. This choice is inspired from Natural Language Processing (NLP) models, including those based on LSTM networks [62], where the input space is a discrete set of Q possible words. These RNN-based methods predict the most likely word index at time step t. Similarly, the LSTM$_\text{A}$ network will predict the most likely radius index for each pedestrian i at instant t, ξ_i^t.

As seen on fig. 3.15, the LSTM$_\text{A}$ has access to a different social context generated by the arc-shaped pooling layer. The LSTM$_\text{A}$ does not process motion, and therefore would benefit little from having knowledge of the motion of the neighbours given by A_i^t. As such, the LSTM$_\text{A}$ cells receive a simpler arc pooling, containing the relative occupancy of neighbours, O_i^t - each cell of the arc will have a value between 0 and 1, where 0 means there are no neighbours and 1 means all neighbours inside the arc are contained in that cell. This is the same technique used by Occupancy-LSTM [11], but normalized in $[0, 1]$ interval. Like the discretization of possible radii, it is a simplifying approach made because of the low supervision of this subtask. The tensor O_i^t is given by:

$$O_i^t(m_r, n_\alpha, :) = \frac{1}{\sum_{j \in \mathcal{N}_i} \mathbf{1}_{r_i^t, \alpha}\left[d_{ij}^t, \beta_{ij}^t\right]} \sum_{j \in \mathcal{N}_i} \mathbf{1}_{m_r, n_\alpha}\left[d_{ij}^t, \beta_{ij}^t\right] \tag{3.30}$$

where $\mathbf{1}_{m_r, n_\alpha}\left[d_{ij}^t, \beta_{ij}^t\right]$ is the indicator function to see if neighbour j is in (m_r, n_α) cell of the FOV of pedestrian i - with angle α and radius r_i^t -, and is computed using (eq. (3.24)). The other indicator function $\mathbf{1}_{r_i^t, \alpha}\left[d_{ij}^t, \beta_{ij}^t\right]$ checks if neighbour is inside the FOV of pedestrian i, regardless of the cell.

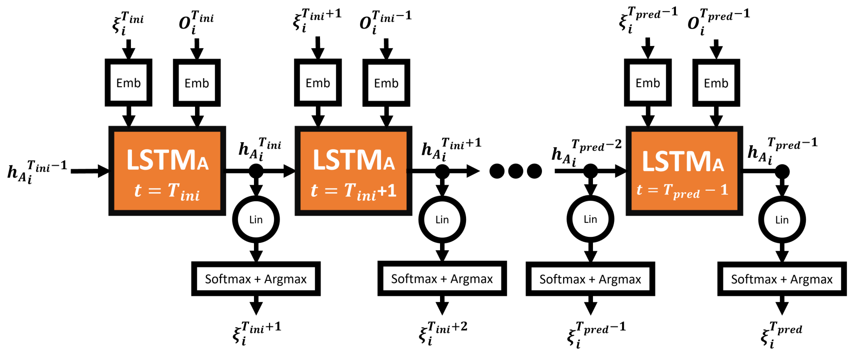


Figure 3.16: Detailed depiction of the LSTM$_\text{A}$. It outputs, for each pedestrian i and instant t, the most likely index ξ_i^{t+1} for the radii for set $\mathcal{R}$. This cell receives an embedding the index from previous instant ξ_i^t, and an embedding of an arc-shape occupancy tensor O_i^t. The embeddings, along with the previous hidden state, $h_{A_i}^{t-1}$, are used to compute the next state. The state goes through a Linear layer and a softmax to obtain the likelihood associated to each radius.

The scheme in fig. 3.16 allows to better understand how the LSTM$_\text{A}$ model works internally. The arc-shaped tensor with relative occupancy, O_i^t, is first embedded before being fed to the LSTM$_\text{A}$. The same is done for the previous arc radius index ξ_i^t, which indicates the radius used to compute O_i^t:

$$\lambda_i^t = Emb_\xi\left(\xi_i^t; W_{emb_\xi}\right), \tag{3.31}$$

$$\gamma_i^t = Emb_O\left(O_i^t; W_{emb_O}\right), \tag{3.32}$$

where both Emb_ξ and Emb_O both have a $PReLu$ activation function. The embeddings λ_i^t and γ_i^t have the same dimensions as e_i^t - computed via (3.25) - and p_i^t - computed via (3.26) -, respectively. Their concatenation will form the direct input to the LSTM$_A$ cell:

$$h_{A_i}^{\;t} = LSTM_A\left(h_{A_i}^{\;t-1}, \left[\lambda_i^t, \gamma_i^t\right]; W_{LSTM_A}\right), \tag{3.33}$$

where the dimensions of the parameters of LSTM$_A$ are the same as of the other LSTM cells. The state goes through a linear layer to obtain an array in $\mathbb{R}^Q$, which can be normalized in $[0,1]$ interval - via softmax layer σ - to obtain the likelihood associated to each radius.

$$\eta_i^{t+1} = W_{out_A} h_{A_i}^{\;t} + b_{out_A} \in \mathbb{R}^Q, \tag{3.34}$$

$$P\left(\xi_i^{t+1} = q\right) = \sigma\left(\eta_i^{t+1}\right)_q = \frac{\exp {\eta_i^{t+1}}_q}{\sum_{k=1}^{Q} \exp {\eta_i^{t+1}}_k} \in [0,1], \tag{3.35}$$

$$\hat{\xi}_i^{t+1} = \underset{q}{\operatorname{argmax}}\, P\left(\xi_i^{t+1} = q\right) \in \{1, ..., Q\}, \tag{3.36}$$

where $\hat{\xi}_i^{t+1}$ identifies the radius to be used by the arc-shaped interaction layer for pedestrian i in the next instant, $r_i^{t+1} = \varrho_{\hat{\xi}_i^{t+1}} \in \mathcal{R}$. The V-LSTM-Arc-LSTM is trained using the same NLL loss function, defined in eq. (3.11). It is important to recall that the prediction of the most appropriate FOV radius is not directly included in the training of this model, since there is no labelled data with the real FOV dimensions of each pedestrian. However, the model can, to some extent, learn the most appropriate FOV dimensions while also learning to predict pedestrian motion.

3.6 Trajectory prediction with scene and interaction-aware LSTM model: Arc-LSTM-SMF

The model to be detailed in this section fulfils the main goal of the thesis: creating a single model that forecasts pedestrian trajectories with the influence of scene-specific elements, and social interactions between pedestrians. This will involve combining the contributions from sections 3.4 and 3.5. For scene-complaint forecast, this model will feature the combination of the second extension of LSTM with SMF, detailed in section 3.4.2, since the LSTM receives more information from the K motion regimes. The consideration of social interactions will come from the fixed arc shape directional pooling, from section 3.5.1. A variable arc configuration from section 3.5.2 could have been employed, but that could imply having different configurations for each scene, and so the fixed shape model was chosen. A schematic for the complete Arc-LSTM-SMF architecture is shown in fig. 3.17.

The Arc-LSTM-SMF model has several similarities with its scene and social standalone counterparts. The most notable of which, common in both Arc-LSTM and LSTM-SMF, is the use of an encoder-

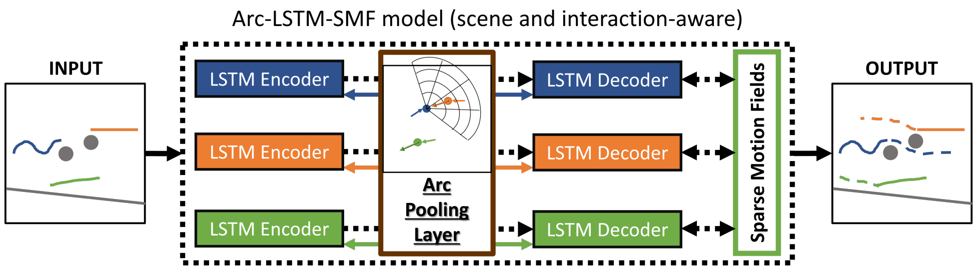


Figure 3.17: Arc-LSTM-SMF architecture for scene and interaction-aware trajectory forecast. The input of the model is a set of trajectories for all pedestrians simultaneously present in a scene. The pedestrians may have obstacles in the way (in gray, not apart of input), and the predictions should avoid them. The social context surrounding each pedestrian is obtained every instant via an interaction layer. The SMF predictions are sent to the LSTM decoder to have a scene-specific consideration of the environment.

decoder architecture. The encoder processes each past pedestrian motion and an arc-shaped pooling of the relative velocities of neighbours. The encoder of this model is identical to the encoder of section 3.5.1, and so it will not be detailed here. The relevant scene information is fed to the decoder via the SMF predictions (along with the same type of information fed to the decoder).

For the Arc-LSTM-SMF decoder, and similarly to the LSTM-SMF-II of section 3.4.2, there are two embeddings of motion, one for the overall model's prediction, another for the K SMF predictions:

$$e^t_{i\,L} = Emb_L\left(\Delta\hat{Y}^t_i; W_{emb_L}\right), \tag{3.37}$$

$$e^t_{i\,MF} = Emb_{MF}\left(\Delta\hat{Y}^{1\ \ t}_{MF\,i}, \ldots, \Delta\hat{Y}^{K\ \ t}_{MF\,i}; W_{emb_{MF}}\right), \tag{3.38}$$

where the embeddings $e^t_{i\,L}$ and $e^t_{i\,MF}$ have the same dimensions, and are computed using layers with separate parameters, W_{emb_L} and $W_{emb_{MF}}$, respectively. These embeddings are combined with the embedding of the arc-shaped social context, p^t_i - computed using (3.26) -, and fed to the LSTM decoder cell:

$$h^t_i = LSTM_d\left(h^{t-1}_i, \left[e^t_{i\,L}, e^t_{i\,MF}, p^t_i\right]; W_{LSTM_d}\right), \tag{3.39}$$

with the state h^t_i being used to obtain the next prediction, which is a bi-variate Gaussian distribution for the pedestrian displacement - $\left[\hat{\mu}^{t+1}_i, \hat{\sigma}^{t+1}_i, \hat{\rho}^{t+1}_i\right] \in \mathbb{R}^5$, computed using (3.28). The deterministic prediction is obtained from the mean of the distribution - $\Delta\hat{Y}^{t+1}_i = \hat{\mu}^{t+1}_i$.

To consider the uncertainty when training, the same NLL loss function as eq. (3.11) will be employed. However, this model will be trained in a per-scene format, as described in section 3.4.3. This means that there should be data from each scene available for both training and testing, to learn relevant scene and social cues, and to evaluate the quality of the predictions with such cues.

The Arc-LSTM-SMF corresponds to the final proposed model, that considers scene-specific information and social interactions, where the only type of input data used is pedestrian trajectories.

4

Experimental results and discussion

Contents

This chapter contains experiments with the model variants proposed in chapter 3, like the original LSTM baseline, and with the incorporation of scene and social information, both individually and combined. To understand and compare the performance of these models, the results for models from chapter 2, as well as simple non-LSTM baselines, are shown. The implementation of the solution from section 5.1 and experiments detailed in this chapter is public and can be accessed at `https://github.com/pedro-mgb/pedestrian-arc-lstm-smf`. There are two types of results to present:

- Quantitative results are made using actual evaluation metrics, and one can compare different methods by comparing the values for those metrics. See section 4.2 for more detailed information on what evaluation metrics have been used.

- Qualitative results are also important to assess the quality of different models, in a less formal way than quantitative ones. For this chapter, these results will usually come in the form of visualization of predictions, comparing between several models and the real trajectory.

4.1 Datasets used for training and testing

The experiments are done on two public datasets, introduced in section 2.5: The BIWI Walking Pedestrians [21] dataset - commonly known as ETH -, and the Crowds by example [20] dataset - commonly known as UCY. There are a total of 4 different scenes - ETH and Hotel belong to BIWI dataset, and Univ and Zara belong to Crowds dataset. Figures 4.1 to 4.4 show visual depictions of each of the scenes.

The original trajectories from these two datasets have variable lengths, as well as variable velocities

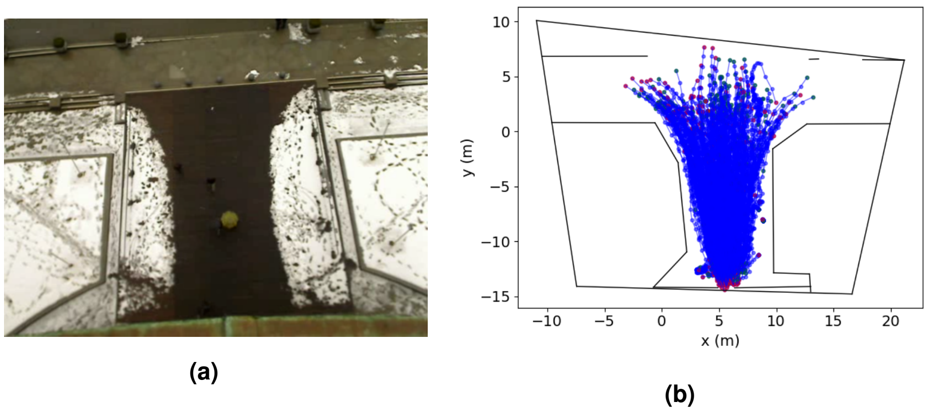
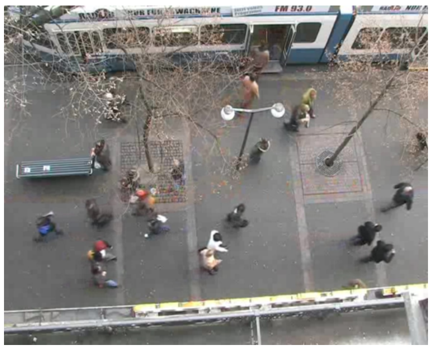
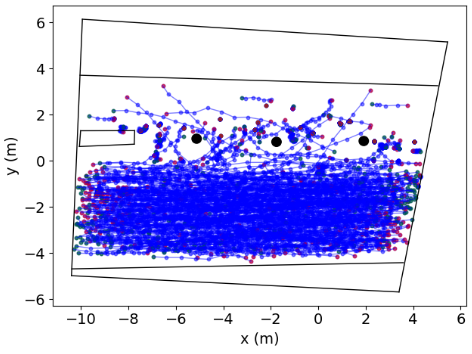


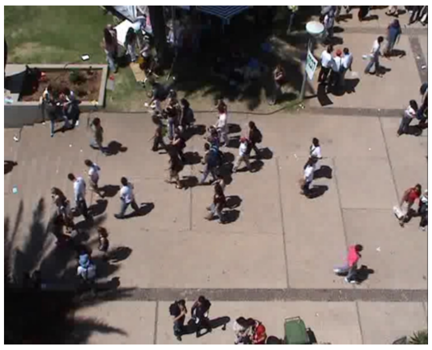
Figure 4.1: Visual representations of the ETH scene, from BIWI dataset. **(a)** A snapshot of a particular frame of the original video. **(b)** Display of the entirety of the scene trajectories, in blue, and in metric units, with delimiting of the scene's limits and unwalkable areas, in black. Note that the snow is considered an unwalkable area for these experiments, even though footsteps can be noticed in (a). For the available data, no pedestrian goes through the snow, but this representation cannot be used in the general case.

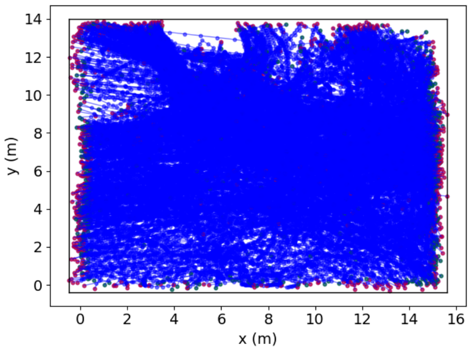

(a)



(b)

Figure 4.2: Visual representations of the Hotel scene, from BIWI dataset. **(a)** A snapshot of a particular frame of the original video. **(b)** Display of the entirety of the scene trajectories, in blue, and in metric units, with the scene's limits and obstacles, in black. Examples of obstacles are the bench, and the lamp post.


(a)



(b)

Figure 4.3: Visual representations of the Univ scene, from Crowds dataset. **(a)** A snapshot of a particular frame of one of the original videos. **(b)** Display of the entirety of the scene trajectories, in blue, and in metric units, with the scene's limits, in black. There are no physical obstacles represented in this scene.

(some pedestrians walk faster, others walk slower). Table 4.1 contains statistical information regarding the aforementioned aspects, for each of the four scenes.

There are more trajectories for the scenes from Crowds dataset (Zara and Univ) than the scenes from BIWI dataset (ETH and Hotel). Moreover, the trajectories of Crowds dataset are usually longer, going on average for over 40 instants ($16\,\mathrm{s}$). The longest trajectories usually involve pedestrians that are in the same position for long periods of time, *e.g.*, when pedestrians are talking with each other.

The Hotel scene is where pedestrians travel a smaller distance. This is because the camera that captures this scene is at a closer distance to the pedestrians (compare the "size" of pedestrians for ETH - fig. 4.1(a) - and Hotel fig. 4.2(a)). For every scene, there are pedestrians near-zero velocity, meaning they are mostly still. The ETH scene has a larger range of velocities and average velocity per pedestrian,

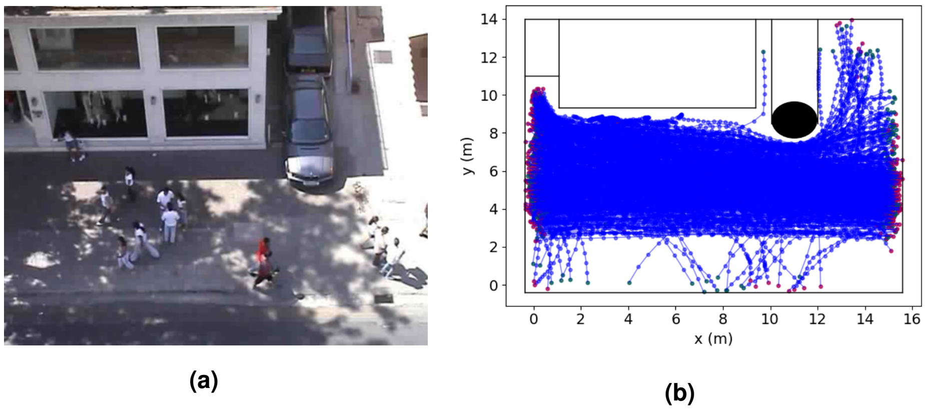


Figure 4.4: Visual representations of the Zara scene, from Crowds dataset. **(a)** A snapshot of a particular frame of one of the original videos. **(b)** Display of the entirety of the scene trajectories, in blue, and in metric units, with delimiting of the scene's limits and unwalkable areas, in black. Note that the cars to the right of the building are represented as an obstacle, being always present in the scene. Obviously they are not a permanent obstacle, which is why this representation cannot be used in the general case.

Table 4.1: Statistical information regarding trajectories in BIWI (ETH) and Crowds (UCY) dataset. The first two scenes belong to BIWI, the last two to Crowds. This table contains per-scene information regarding the number of trajectories, how much do the lengths vary, the distance travelled and the relative displacement (per-instant) of the pedestrians. Note that the velocity could be easily obtained from the displacement, since the time interval is constant ($0.4\,\mathrm{s}$)

Scene	Total number of trajectories	Trajectory length range (Min - Max)	Trajectory length (Avg. / Std.)	Average distance travelled (m)	Pedestrian displacement per instant (m) (Min - Max)	Pedestrian displacement per instant (m) (Avg. / Std.)
ETH	354	$3-114$	$15.48/6.06$	13.29	$0.00-2.37$	$0.96/0.29$
Hotel	378	$3-100$	$17.25/12.14$	6.75	$0.00-0.89$	$0.47/0.26$
Univ	967	$4-352$	$43.96/19.93$	11.65	$0.03-0.92$	$0.34/0.16$
Zara	489	$7-584$	$40.65/10.64$	14.54	$0.03-0.86$	$0.46/0.13$

because the original video from which the trajectories were sampled has been sped up.

4.1.1 Trajnet++ data configuration

Most works [6, 8, 9, 11–14, 33] that have used BIWI [21] and Crowds [20] datasets for training and comparing with other methods, do not use full-length trajectories. There is a very large range for trajectory lengths, and for the case of interacting pedestrians, there are not many pedestrians with complete trajectories of the same length, in the same time frame.

In the past year, several works [8, 11, 27, 28] have been compared using the Trajnet++ benchmark [26]. Trajnet++ structures trajectory data into fixed length portions, with a focus on capturing as many interactions in each portion as possible. It is this explicit prioritization of social cues in the data and evaluation metrics that motivated the choice to use the Trajnet++ configuration for all experiments of this chapter.

The Trajnet++ data configuration defines the concept of primary pedestrian - in a set of trajectories, some may be more unique and rich in interactions. It is those trajectories that are more interesting from a forecasting point of view. For the choice of primary pedestrian, Trajnet++ prioritizes pedestrians with more relevant social interactions (see fig. 3.8), and limits the number of primary pedestrians that have less relevant trajectories (*e.g.*, motionless or quasi-motionless, linear paths). The remaining pedestrians that do not classify as primary are also available for trajectory forecast, but are only used as social context, *i.e.*, as neighbours to the primary pedestrian. The resulting situations[1] will be richer and more interaction-centric, enabling trajectory forecasting methods to better learn social cues. For more information on the choice of primary pedestrians, see [26].

The original Trajnet++ data[2] considers a fixed length of $L = 21$ instants, with the first 9 instants for the past trajectory (input to the model), and the last 12 instants for GT (to compare with prediction). In reality, the first instant is only used to compute the displacement (the models of [26] and chapter 3 work with displacements) from the second instant, so one can say that the data length is 20 instants, which is the same value that was used for Trajnet [57], and by many prior works [8, 9, 11, 33]. Pedestrian trajectories with length smaller than L can never be used for as a primary pedestrian; instead they are just used as social context. Pedestrian trajectories with length larger than L are split across time, and so a single pedestrian with a long trajectory may have several trajectory splits that are primary. Furthermore, for a set of trajectories with length $L = 21$, there is the possibility to construct more than one minibatch, if more than one pedestrian classifies as a primary pedestrian.

A benefit in using trajectories of fixed length L is model efficiency. It is easier to aggregate trajectories of fixed length in the same batch, rather than trajectories of variable length. The use of uniform-length batches of trajectories means the model training and inference can be better mapped to processing architectures such as Graphics Processing Unit (GPU), reducing training and inference times.

There are differences between the data used in this chapter and the original Trajnet++ data:

- The original Trajnet++ benchmark has scenes that are only available training (*e.g.* Hotel). To properly test scene-specific models (like Arc-LSTM-SMF), each scene should have portion for training and a portion for testing. Therefore, the data from the BIWI/Crowds dataset was split, with around 50% duration of each scene going to the training and testing sets. Afterwards, it was converted to the Trajnet++ format, using their publicly available conversion code.

- Other datasets available in Trajnet++ [58–60] were not used. It is worth noting that these all belonged to the training set and were not being used for evaluation in this benchmark.

- Besides $L = 21$, some experiments will also feature trajectories with length $L = 11$ due to the lack

[1] In [26], a set of pedestrian trajectories in a certain time frame was labelled as a scene. Here, it will be labelled as minibatch or situation, to avoid confusing with the already employed concept of scene (buildings, terrain, etc.)

[2] Trajenet++ Benchmark data available here, except for ground truth test data, which is server private.

of trajectories of length $L = 21$ for some scenes.

- The original trajectories of Trajnet++ were obtained with a stride of 2 instants, meaning a pedestrian could become primary pedestrian if it has not been primary in the previous 2 instants. This technique resulted in an increase of primary pedestrians, but at the cost of having a lot of repeated information. This can hinder the ability for the model to learn diverse social interactions, since it is fed interactions that are almost the same. As such, the stride to convert the data was changed to the actual length L. No pedestrian will ever have repeated portions of their original trajectory.

Table 4.2 has information on the number of trajectories for each scene, for both training and testing sets. There are also statistics on the Trajnet++ conversion, regarding the number of minibatches and distinct primary pedestrians - remember that a pedestrian may become primary more than once if their trajectory is long and relevant enough -, using the configuration of Trajnet++ described above.

Table 4.2: Original number of trajectories for training and testing sets, as well as the number of resulting minibatches in Trajnet++ [26] format with fixed lengths up to $L = 21$ and $L = 11$ (primary pedestrians must have length L, neighbours can have less). The number of distinct primary pedestrians (less or equal to the number of minibatches) for each fixed length is also shown.

Scene	Original number of trajectories Train / Test	Trajectories up to $L = 21$		Trajectories up to $L = 11$	
		Total number of minibatches Train / Test	Distinct primary pedestrians Train / Test	Total number of minibatches Train / Test	Distinct primary pedestrians Train / Test
ETH	$194/124$	$34/3$	$28/3$	$197/86$	$170/86$
Hotel	$103/150$	$25/31$	$25/26$	$90/95$	$69/72$
Univ	$435/362$	$624/532$	$340/310$	$1232/1069$	$416/351$
Zara	$191/204$	$201/243$	$171/183$	$454/531$	$187/198$
All Scenes	$\underline{\mathbf{923/840}}$	$\underline{\mathbf{884/809}}$	$\underline{\mathbf{564/522}}$	$\underline{\mathbf{1973/1781}}$	$\underline{\mathbf{842/706}}$

As could be previously expected from the statistics in table 4.1, there are much fewer minibatches for ETH and Hotel, due to the fact that there are fewer trajectories for these scenes, and their average length is smaller than for Univ or Zara. For ETH and Hotel, the number of minibatches with $L = 21$ is much smaller than the original number of trajectories. This is what motivated the use of a smaller split length, $L = 11$, so that those scenes had a substantial number of trajectories for training and evaluation.

4.2 Evaluation metrics

For evaluation, several metrics will be used to compare trajectory forecasting methods, including their ability to capture the influencing cues of human motion and behaviour.

The simplest metrics used are also the most popular [5]. These are the distance-based metrics that indicate numerically, how close is the prediction to the real trajectory. The two distance-based metrics used for the experiments are ADE and FDE, already addressed in section 2.5. Using the notation introduced in section 1.2, ADE and FDE are computed using the following equations:

$$ADE = \frac{1}{T_{pred} - T_{obs}} \sum_{t=T_{obs}+1}^{T_{pred}} \left\| \hat{Y}_i^t - Y_i^t \right\|_2, \tag{4.1}$$

$$FDE = \left\| \hat{Y}_i^{T_{pred}} - Y_i^{T_{pred}} \right\|_2, \tag{4.2}$$

where $\| \cdot \|_2$ denotes the euclidean distance, $\hat{Y}_i^t \in \mathbb{R}^2$ and $Y_i^t \in \mathbb{R}^2$ are respectively the predicted and GT positions for pedestrian i and time step t. Using Trajnet++ data configuration, these metrics are only computed for the primary pedestrian $i = 1$ of each minibatch. This will also apply to the other metrics detailed in section 4.2.1 and 4.2.2. When considering predictions for several minibatches in a scene, the results will be in form of a compact ADE/FDE, averaged across all trajectories of primary pedestrians. For both these metrics, the lower their value is, the closer the prediction is to the GT, and the prediction can be seen as better, in that regard.

As mentioned in section 2.5, plain distance-based metrics are not enough to fully compare the quality of predictions. The next sections dive into the efforts made to combine, adapt, or create new evaluation metrics that take into account two kinds of cues that influence human motion - the presence of physical obstacles in the scene, and the presence of multiple pedestrians interacting with each other.

4.2.1 Scene-specific metrics

This section explores how to evaluate if the model predictions comply (or not) with the scene's structure. From section 2.3 and 2.4, some models in the past have used scene context to perform more scene compliant trajectory predictions [2, 9, 16]. However, none have formally evaluated how well were the scene-specific cues being captured. Evaluation was merely done in a qualitative sense.

The proposed scene-specific metric involves considering the existence of obstacles and unwalkable areas (*e.g.*, poor terrain) in the scene. A map was manually built for each of the available scenes in BIWI [21] and Crowds [20] datasets (visible in figs. 4.1(b) to 4.4(b), in black). Formally, the evaluation metric is the average number of times a pedestrian collides with an obstacle or goes to an unwalkable area - labelled as Collisions with Scene Environment (CSE). A pedestrian is said to collide with an obstacle or a forbidden zone if the predicted trajectory intersects the obstacle or limits of that zone. A similar concept was present in [10], although just in an ablation study for SDD [38]. To the best of our knowledge, this is the first work to use such a metric in the BIWI [21] and Crowds [20] datasets.

It is worth noting that some scenes (*e.g.*, Univ of Crowds dataset) don't really contain any obstacles or areas that pedestrians cannot transverse. To also have some consideration of scene compliance for this type of scenes, a different metric was used, which can be formalized as the average number of trajectories that go outside of the scene's bounds - labelled as Out of Scene's Bounds (OSB). The scene's bounds can be seen as a box, and all the trajectories (originated from the datasets) are inside that box. These bounds can also be seen for the BIWI and Crowds scenes, in figures fig. 4.1(b), 4.2(b), 4.3(b), and 4.4(b) (boxes in black, not necessarily square, that include all the trajectories).

This concept of out of the scene for an evaluation metric is less robust than the consideration of actual obstacles. In most cases, the scene's bounds do not constitute a real obstacle that the pedestrians cannot cross. For instance, in the case of a sidewalk (*e.g.*, in fig. 4.4(a)), the trajectories can continue beyond the scene's bounds, since they are still on the sidewalk. So it is not as detrimental to have a high value for OSB, when compared to having a high value of CSE.

To visually understand these two scene-specific evaluation metrics, there is an example of trajectory with CSE in fig. 4.5(a), and an example of a trajectory that would increase the OSB metric in fig. 4.5(b).

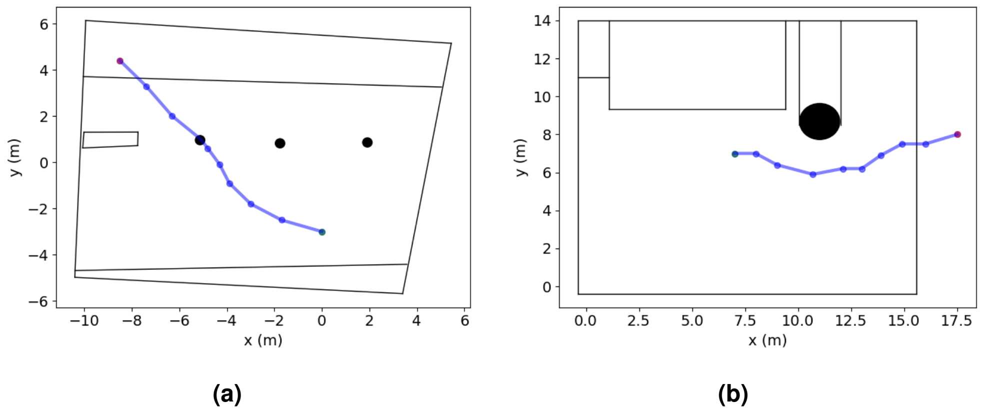


Figure 4.5: Fictitious examples of trajectories that are not scene-compliant **(a)** Trajectory in Hotel scene. There is a collision with an obstacle (black circle), while also going to an unwalkable area (beyond black line). The CSE value would increase by 2. **(b)** Trajectory in Zara scene. This trajectory goes beyond the scene's bounds (black rectangle).

4.2.2 Interaction-specific metrics

Interaction-centric evaluation has already been explored in recent works [9,26,48]. In all of the aforecited works, one concept was always present: collision avoidance. A common pedestrian behaviour involves maintaining a certain distance (if possible) from others to avoid clashing. A collision metric can be built by defining a safety distance - collision threshold, T - that all pedestrians should maintain.

In the conducted experiments, the evaluation of social accuracy will be based on metrics from the Trajnet++ benchmark. These metrics are Col-I - percentage of times a primary pedestrian collides with neighbours, using the predicted trajectories for all pedestrians -, and Col-II - percentage of times a primary pedestrian collides with neighbours, using GT trajectories for the neighbours. To better distinguish the two, Col-I will be referred as Col-P and Col-II as Col-GT. The Col-P metric is more significant than Col-GT, since when inferring, the model cannot access the GT. However, in certain situations, it may be important to analyze the compliance of the primary pedestrian using the real social context.

Examples of situations where collisions would occur can be found in fig. 4.6. It is important to note that the Col-P and Col-GT metrics restrict the number of collisions between the same two pedestrians to 1. This is to avoid situations like that of fig. 4.6(b), where two pedestrians in a group are travelling very close to each other, resulting in multiple "collisions", but with all of them are derived from the fact they are travelling in group. If one were to include all those collisions, it would inflate the results, when compared to single collisions like that of fig. 4.6(a), which should have just as much relevance.

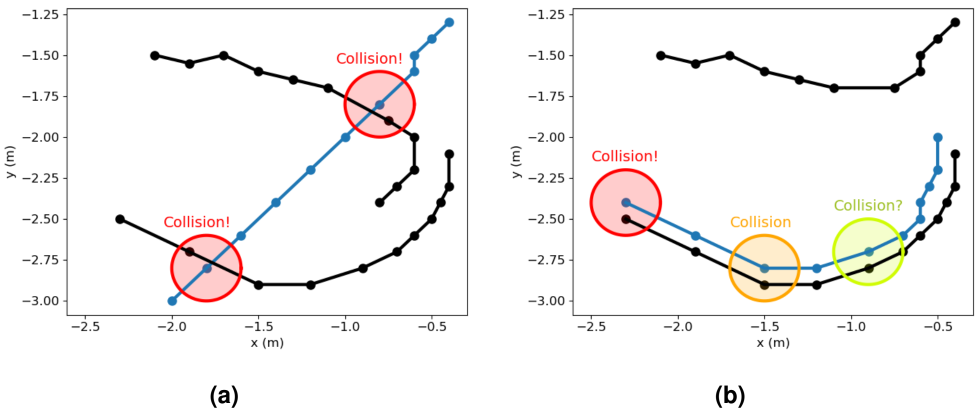


Figure 4.6: Fictitious examples of two socially-relevant situations, that result in collisions ($T = 0.2\,\mathrm{m}$). Visually, a circle is drawn on each position of the primary pedestrian (blue trajectory), and a collision occurs if any of the neighbours (black trajectories) in that instant intersect the circle. **(a)** Primary pedestrian has two collisions, one with each neighbour. **(b)** The primary pedestrian has several "collisions" with the same neighbour (group travelling), but only the first one will count, to avoid inflating the overall results.

Formally, using the notation of chapter 3, the Col-P metric (in absolute values) can be defined as:

$$\text{Col-P} = \sum_{j \in \mathcal{N}_1} \sum_{t=T_{obs}+1}^{T_{pred}} \begin{cases} 1 & \text{, if } \|\hat{Y}_1^t - \hat{Y}_j^t\|_2 < T \\ 0 & \text{, otherwise} \end{cases}, \tag{4.3}$$

where the subscript $_1$ indicates the primary pedestrian, and $\mathcal{N}_1$ the neighbourhood of that pedestrian. The Col-P metric is only computed for the primary pedestrian, but using the predictions of neighbours. For the Col-GT metric, the only difference involves replacing $\hat{Y}_j^t$ with Y_j^t in (4.3). The results will be displayed in percentage, which will involve dividing by the total number of primary pedestrians.

The collision threshold used for interaction-centric evaluation was $T = 0.1\,\mathrm{m}$. It was the maximum value for which no collisions occurred in the test set (and almost no collisions in the training set). It is also a value commonly chosen in recent works [9, 26]. With $T = 0.1\,\mathrm{m}$, any collision that results from model predictions is socially inaccurate, since it is not present in the test set.

4.3 Model implementation details and baselines

All LSTM models addressed in chapter 3 were implemented with the PyTorch framework[3]. All models were trained using Adam optimizer [68] with a starting learning rate of $\alpha = 0.001$, and decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, with the number of training epochs never exceeding 100, and a batch size of 8.

The LSTM models used a hidden state dimension of 128, with position embedding size of 64 (for the models that integrated SMF predictions, it consists in 32 for LSTM displacement embedding, and 32 for SMF displacement embedding). The embedding dimension used for the interaction tensor (3.26) was 256. The arc shaped pooling has $N_r = 4$ cells for radius and $N_\alpha = 5$ cells for angle, with $r = 4\,\mathrm{m}$ and $\alpha = 140°$. These hyper-parameters were the ones that yielded the best results on a separate validation set, with trajectories from the BIWI and Crowds datasets.

The SMF models, one for each of the four scenes, were built using similar configuration to that of [6]. Unlike other models, they were trained on the original dataset trajectories of variable length. This is because the goal of SMF in this thesis is to help in improving scene compliance, therefore it is not desirable to lose trajectories of pedestrians because they do not have a predefined length.

There are a few optimization choices that have been presented in recent works that use the Trajnet++ benchmark [26, 28], that will also be used for these models. The loss functions used for training the models of chapter 3 are computed only for the primary pedestrians [26]. This means that the trajectories of the neighbours are not used for methods like the base LSTM (section 3.2) and its integration with SMF (section 3.4). For the models sections 3.5 and 3.6, the GT of the neighbour trajectories is used instead of the predictions. This is a concept known as teacher forcing, already employed in [11, 26].

Several models from chapter 3 will be used in the experiments of this chapter. To distinguish between them, the following labels will be defined:

- LSTM: the base model from section 3.2 that predicts motion without social or scene information.
- LSTM-SMF-I: first integration of SMF with LSTM, detailed in section 3.4.1.
- LSTM-SMF-II: second integration of SMF with LSTM, detailed in section 3.4.2.
- Arc-LSTM: socially-aware LSTM model with fixed arc (or FOV) shape for directional pooling of neighbours. It is described in section 3.5.1.
- V-Jn-Arc-LSTM: variable arc shape that includes up to J=20 neighbours (section 3.5.2.A). Arc radius between $r_{min} = 2\,\mathrm{m}$ and $r_{max} = 8\,\mathrm{m}$.

[3]Pytorch framework obtained from `https://pytorch.org`

- V-ND-Arc-LSTM: arc shape that varies on the mean neighbour distance (section 3.5.2.B). Arc radius between $r_{min} = 2\,\mathrm{m}$ and $r_{max} = 8\,\mathrm{m}$, and coefficients $B = C = 1$ (1:1 mapping).
- V-LSTM-Arc-LSTM: variable arc shape via LSTM network (section 3.5.2.C), with a discrete set of arc radius values - $\{2, 3, 4, 5, 6, 7, 8\}\,\mathrm{m}$. Radius value at T_{ini} is randomly picked from the set. All quantitative results are averaged from 5 independent inferences with initial random radii.
- Arc-LSTM-SMF: social and scene-aware LSTM model with arc shaped directional pooling and SMF predictions. It is described in section 3.6.

To better assess the performance of the aforementioned models, some experimental results will also be shown for state-of-the-art models mentioned in chapter 2, as well as simpler baselines:

- Constant Velocity (CV): a baseline that uses the last observed velocity for the whole prediction.
- SMF [6, 61]: the standalone SMF models, trained on each scene referred in section 4.1.
- S-LSTM [11]: Social LSTM - the first socially-aware LSTM model used for this task. The implementation from the Trajnet++ benchmark will be used, since it was built for its data configuration.
- D-LSTM [26]: Directional LSTM, which can be seen as an efficient improvement to S-LSTM. Implementation supplied by the authors of Trajnet++, here.
- S-GAN [8]: Social-GAN, which allows multiple samples of socially aware trajectories. For the same reason as S-LSTM, the implementation from Trajnet++ is used.
- S-STGCNN [33]: Socially-aware multimodal method that uses STGCNN. Implementation adapted from original to support Trajnet++ data, available here.

Unfortunately, there are not any methods that explicitly consider the scene. except for SMF. This is due to two reasons. First, almost none of them have publicly available implementations, as seen in table 2.1. Second, these methods do not support data in the Trajnet++ format, and would require considerable effort to make then support it.

4.4 Scene-specific evaluation using LSTM integrated with SMF

The first major contribution to the trajectory forecast problem was the integration of SMF [6] with LSTM networks (section 3.4), creating an environment-aware model. The network implicitly learns how to weigh the predictions of SMF and LSTM to generate more accurate and scene-compliant trajectories.

This section covers the experiments conducted with these models (labelled in section 4.3 as LSTM-SMF-I,II), when compared to their standalone parts - just LSTM and just SMF -, and a simple CV method.

First, quantitative results will be shown, using not only ADE and FDE, but also scene-specific evaluation metrics introduced in section 4.2.1. Then, some qualitative results will be used to provide a different perspective on the predictions of several methods.

4.4.1 Quantitative results

The first objective is to understand if integrating SMF with LSTM results in improving the predictions, from a standpoint of the trajectory errors. This will involve comparing the extensions LSTM-SMF-I and LSTM-SMF-II with plain LSTM and SMF, as well as CV, using the test set described in section 4.1. The ADEs and FDEs, defined in (4.1) and (4.2), respectively, are shown in table 4.3 for trajectories of length $L = 21$, and in table 4.4 for trajectories of length $L = 11$.

Table 4.3: Comparison of baseline models (first 2) with models that consider scene-specific elements for trajectory forecasting (last 3). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Displacement errors are in *metres*. Lowest error (best model) is shown in **bold and underline**.

Scene	CV ADE / FDE	LSTM ADE / FDE	SMF ADE / FDE	LSTM-SMF-I ADE / FDE	LSTM-SMF-II ADE / FDE
ETH	$0.73/1.04$	$0.87/1.22$	$0.97/\underline{\mathbf{0.72}}$	$0.81/0.99$	$\underline{\mathbf{0.54}}/0.84$
Hotel	$0.52/1.03$	$0.53/1.08$	$0.75/1.41$	$0.51/1.05$	$\underline{\mathbf{0.43}}/\underline{\mathbf{0.72}}$
Univ	$0.68/1.49$	$0.69/1.46$	$1.30/2.41$	$0.69/1.48$	$\underline{\mathbf{0.67}}/\underline{\mathbf{1.43}}$
Zara	$\underline{\mathbf{0.53}}/1.21$	$0.54/1.18$	$0.95/1.67$	$0.57/1.16$	$0.54/\underline{\mathbf{1.12}}$
Average	$0.62/1.19$	$0.66/1.24$	$0.99/1.55$	$0.64/1.17$	$\underline{\mathbf{0.55}}/\underline{\mathbf{1.03}}$
Weighted Average	$\underline{\mathbf{0.63}}/1.38$	$0.65/1.40$	$1.17/2.14$	$0.65/1.37$	$\underline{\mathbf{0.63}}/\underline{\mathbf{1.31}}$

Table 4.4: Comparison of baseline models (first 2) with models that consider scene-specific elements for trajectory forecasting (last 3). Use of data configuration described in section 4.1, with trajectory splits of fixed length $L = 11$. Displacement errors are in *metres*. Lowest error (best model) is shown in **bold and underline**.

Scene	CV ADE / FDE	LSTM ADE / FDE	SMF ADE / FDE	LSTM-SMF-I ADE / FDE	LSTM-SMF-II ADE / FDE
ETH	$0.42/0.76$	$0.40/0.72$	$0.62/0.99$	$0.40/0.71$	$\underline{\mathbf{0.38}}/\underline{\mathbf{0.64}}$
Hotel	$0.25/0.45$	$0.25/0.46$	$0.54/0.92$	$0.26/0.46$	$\underline{\mathbf{0.24}}/\underline{\mathbf{0.41}}$
Univ	$\underline{\mathbf{0.28}}/0.57$	$\underline{\mathbf{0.28}}/0.57$	$0.66/1.14$	$0.29/0.59$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.56}}$
Zara	$0.22/0.46$	$0.23/0.43$	$0.53/0.90$	$0.24/0.48$	$\underline{\mathbf{0.21}}/\underline{\mathbf{0.42}}$
Average	$0.29/0.56$	$0.29/0.55$	$0.59/1.00$	$0.30/0.56$	$\underline{\mathbf{0.25}}/\underline{\mathbf{0.46}}$
Weighted Average	$0.27/0.54$	$0.27/0.54$	$0.61/1.05$	$0.28/0.55$	$\underline{\mathbf{0.26}}/\underline{\mathbf{0.52}}$

All models have smaller values of ADE and FDE for Hotel and Zara scenes, due to having simpler scenarios and more linear trajectories. In contrast, the range of velocities for ETH is quite high (the original video is sped up), and Univ scene is more crowded, with lots of trajectories having irregularities, to avoid clashing with neighbours. In ETH, for $L = 21$, the ratio between ADE and FDE is smaller than for $L = 11$. This is due to the fact that there are only 3 primary pedestrians for the former, and so a change in one prediction impacts the results more than for $L = 11$ (total of 86 primary pedestrians).

One can notice that the base LSTM performs very similar to the simple CV method in most scenes. This has already been discovered and discussed in a recent work [69], having shown that CV can outperform several complicated data-driven models [8, 9, 11]. The base LSTM is trained to minimize on average the error *w.r.t* the GT, and without any other information at its disposal, its predictions converge to a straight line in most cases (which will also be noticeable in the qualitative evaluation of section 4.4.2).

The extensions of LSTM with SMF slightly improve the results. In particular, LSTM-SMF-II (section 3.4.2) outperforms CV and base LSTM, especially in the FDE metric. It even has better performance in the Univ scene, where the environment does not pose much influence (see fig. 4.3). The LSTM network benefits from deciding how to apply the SMF predictions (applying just one, or possibly multiple of them), as opposed to just receiving the most likely and uncorrected prediction from the SMF model.

Geometrically, the SMF has the worst performance of all models. However, integrating SMF with LSTM decreases the displacement errors, because the LSTM architecture learns when (and when not) to use SMF predictions. The main use of the SMF in the proposed models is to enforce the LSTM to generate scene-complaint trajectories. To evaluate this, a different metric is required.

The quantitative scene-specific metric to be used is the average number of CSE (be it obstacles, or going through unwalkable areas) per pedestrian, as introduced in section 4.2. To also evaluate in scenes with no obstacles, a secondary metric was introduced - average number of OSB. Table 4.5 contains the results of those metrics for the same 5 types of models of tables 4.3 and 4.4. The results are only for $L = 21$, since almost no collisions occurred for $L = 11$. CSE and OSB were 0 for both ETH and Hotel scenes, therefore they do not appear in table 4.5.

Table 4.5: Scene-specific evaluation - Average number of CSE per primary pedestrian, and average number of OSB trajectories. Same models as in table 4.3. Use of Trajnet++ data configuration described in section 4.1, with trajectories of length $L = 21$. Results are displayed in percentages (%). Lowest values (best model) shown in **bold and underline**. ETH and Hotel scenes omitted since there were no collisions for any of the methods. Univ scene has '-' in CSE values since no obstacles were registered for the scene.

Scene	CV CSE / OSB	LSTM CSE / OSB	SMF CSE / OSB	LSTM-SMF-I CSE / OSB	LSTM-SMF-II CSE / OSB
Univ	$-/4.1$	$-/2.4$	$-/\underline{\mathbf{0}}$	$-/2.6$	$-/2.8$
Zara	$2.5/2.1$	$1.6/2.1$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$0.8/4.5$	$0.4/1.6$

Even for the scenes where collisions occurred, their number is still relatively low. For instance, for LSTM, 1.6% of collisions on Zara scene (with 243 primary pedestrians, from third column of table 4.2) corresponds to 4 collisions. This calls for the need to also perform similar experiments on datasets with more scene restrictions, such as the SDD [38] dataset, The CSE and OSB values were also computed for primary pedestrians and neighbours (used as social context for Trajnet++), to have additional trajectories when evaluating the ability of the LSTM models with SMF to learn scene context. Those results are shown in table 4.6.

Table 4.6: Summary of results for scene-specific evaluation - Average number of CSE for all pedestrians (including the neighbours), and average number of trajectories that are OSB. Use of Trajnet++ data configuration described in section 4.1, with trajectories of length $L = 21$ and $L = 11$. Results are displayed in percentages (%). Lowest values (best model) shown in **bold and underline**.

Length	CV CSE / OSB	LSTM CSE / OSB	SMF CSE / OSB	LSTM-SMF-I CSE / OSB	LSTM-SMF-II CSE / OSB
$L = 21$	$1.5/14.5$	$0.7/11.9$	$\underline{\mathbf{0.4}}/\underline{\mathbf{0}}$	$0.7/11.4$	$\underline{\mathbf{0.4}}/12.4$
$L = 11$	$0.7/6.7$	$0.4/6.7$	$\underline{\mathbf{0.1}}/\underline{\mathbf{0}}$	$0.5/5.9$	$0.2/6.5$

The model that is distinctively less scene-compliant is the CV model, having the largest number of CSE and OSB. Judging from displacement errors alone on table 4.3 and 4.4, the CV would be the preferable method to use, since performance is similar to LSTM models, and does not require training. But this model has no notion of obstacles and cannot predict changes of direction to stay within the walkable regions of the scene.

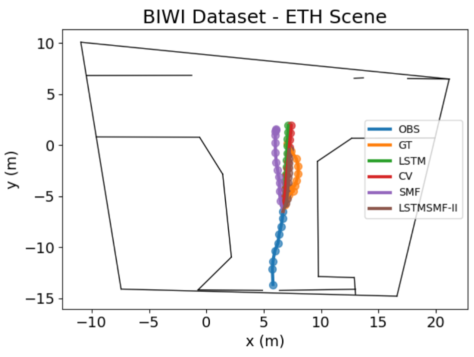
The base LSTM of section 3.2 has better performance than the CV model, especially for lengths $L = 21$. This shows it can implicitly learn, to some extent, how to avoid going into obstacles and unwalkable areas. Of course, this LSTM is not explicitly using any scene-relevant information.

While the SMF method was the one with highest displacement errors, here it has the lowest collisions with static obstacles, and zero trajectories going out of the scene's bounds. The restrictions on the motion fields to be sparse have allowed the model to learn how to avoid areas where pedestrian traversal is not possible. The absence of trajectories out of scene bounds is a result of another characteristic of the model: the normalization and limit of the positions in $[0, 1]^2$ (image lattice) interval. Because of that, the trajectories theoretically cannot go beyond the limits of the original scene images.

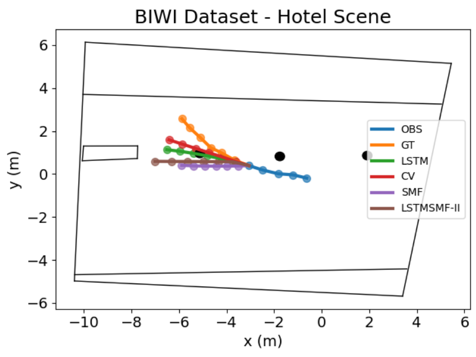
The LSTM-SMF-II of section 3.4.2, when compared to the base LSTM, has less collisions with the environment, both for primary pedestrians and their neighbours. It is even almost on par with SMF in table 4.6. This shows that the LSTM-SMF-II model can be scene-aware, while also lowering geometric errors. However, the value of OSB is not distinctively better (in some cases being worse) when compared to the base LSTM, and LSTM-SMF-I. The LSTM predictions are free to go beyond the scene limits, while the SMF predictions are not. In some cases those SMF predictions can be seen as ”unnatural”, leading to the LSTM network not considering them. The next section will provide further insights on this aspect.

4.4.2 Qualitative results

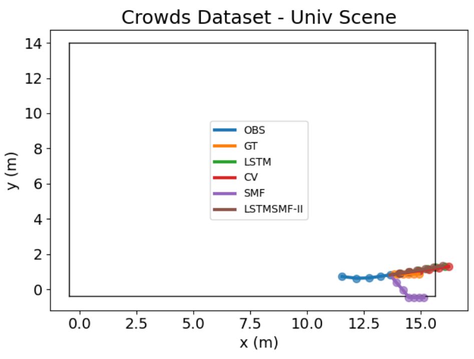
This section aims to provide a different way of evaluating the quality of the model's predictions. This will be done by displaying some predictions of the test set trajectories using the several models evaluated on the previous section. Predictions from different scenes are shown in fig. 4.7, for 4 methods, each identified by a different colour: CV (green); base LSTM (red); SMF (purple); LSTM-SMF-II (brown). The predictions shown are a small but relevant sample of the test set, with different situations to highlight the strengths and limitations of the proposed model for scene-compliant trajectory prediction.



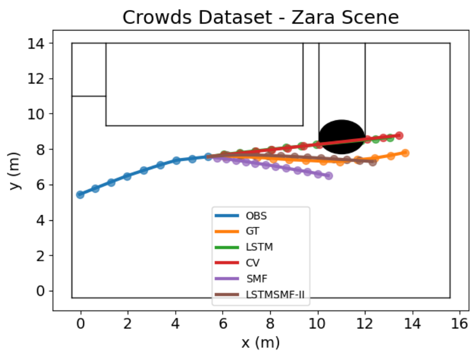
(a)



(b)



(c)



(d)

Figure 4.7: Predictions of several trajectories (some with length $L = 11$, others $L = 21$) for 4 scenes: ETH in **(a)**; Hotel in **(b)**; Univ in **(c)**; Zara in **(d)**. There are predictions of 4 models: LSTM in green, CV in red, SMF in purple, and LSTM integrated with SMF in brown. Past trajectory (OBS) in blue, and GT in orange.

The first situation, in fig. 4.7(a), is fairly simple - a trajectory from ETH scene, with the past trajectory being mostly linear. The LSTM and CV predictions are practically identical, as hinted by the similar results in section 4.4.1. The LSTM-SMF-II prediction has the same direction, but with reduced speed, which makes this prediction the one with less ADE. No model, however, predicts the curve that the pedestrian would take, since there is no obstacle in the way, and the past does not indicate any curve.

The trajectory in fig. 4.7(b) enlightens the contribution of SMF predictions to the LSTM network. At the end of the past trajectory, the pedestrian is moving towards an obstacle. The CV and LSTM models, not being scene aware, predict the pedestrian will continue in the same direction, thus colliding with the obstacle. The SMF method learned from the training set that an obstacle was present, avoiding it. The LSTM-SMF-II does not learn this directly, but has access to K SMF predictions, using them to predict a change of direction in the pedestrian to avoid the obstacle. However, the change of direction does not correspond to the real one. The prediction is scene-compliant, but worse from a geometric standpoint.

A scene-specific concept that LSTM-SMF-II does not properly learn is scene bounds. As mentioned on section 4.4.1, even though the SMF has no predictions out of scene bounds, they may seem as unnatural to the LSTM network. An illustration of this is present in fig. 4.7(c). All methods except SMF predict the pedestrian will keep moving with the same velocity and go beyond the scene's bounds (the bounds of the original video). The SMF limit the prediction to the bounds, but if it were not for that theoretical limitation, the SMF prediction would go outside the scene's bounds as well. This sudden curve (and that of other predictions that were not SMF's most likely one, not visible in fig. 4.7(c)) makes the LSTM-SMF-II prioritize the LSTM decoder's prediction.

The concept of scene bounds is not as important as actual collisions with the environment, since in these scenes most of the bounds do not pose an actual physical barrier. As such, situations like that of fig. 4.7(d) are more important. The base LSTM and CV predictions do not account for the presence of the obstacle, resulting in a collision. The LSTM-SMF-II has scene-aware predictions from SMF, predicting an accurate direction for the pedestrian. This prediction can be seen as a mixture of the base LSTM and the most likely SMF prediction, showing that LSTM-SMF-II can correctly weigh predictions from both separate methods to generate more accurate predictions.

Overall, it can be concluded that the integration of LSTM with SMF provides a boost in generating more accurate predictions, as well as decreasing the amount of pedestrians that violate the context of the scene environment. However, there are still limitations on the LSTM network's ability to generate curved trajectories. One way to better predict when a pedestrian will change direction is if the pedestrians in the proximity are considered. Such experiments will be discussed in the next section.

4.5 Social evaluation of interaction-aware LSTM models

Apart from the scene environment, the other type of influencing cue considered in the proposed solution of chapter 3 is the existence of several pedestrians in the vicinity and underlying social interactions. This section will explore the quality of the predictions of the interaction-aware models of section 3.5, with use of the metrics introduced in section 4.2.2. Baseline models such as CV and LSTM will be compared with Arc-LSTM models, both with fixed arc dimensions (plain Arc-LSTM - section 3.5.1) and variable dimensions (V-Jn-Arc-LSTM, V-ND-Arc-LSTM and V-LSTM-Arc-LSTM - section 3.5.2).

4.5.1 Quantitative results

Similarly to section 4.4.1, the first goal is to see if including social interactions improves the overall quality of the predictions. For that, the distance-based metrics introduced in section 4.2 - ADE and FDE - are computed for the same data of the previous section. The results are shown in table 4.7 and table 4.8 for primary pedestrians with trajectory length $L = 21$ and $L = 11$, respectively.

Table 4.7: Comparison of baselines (first 2) with interaction-aware models (last 4). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Displacement errors are in *metres*. Lowest error (best model) is shown in **bold and underline**.

Scene	CV ADE / FDE	LSTM ADE / FDE	Arc-LSTM ADE / FDE	V-Jn-Arc-LSTM ADE / FDE	V-ND-Arc-LSTM ADE / FDE	V-LSTM-Arc-LSTM ADE / FDE
ETH	$0.73/1.04$	$0.87/1.22$	$0.73/1.01$	$0.94/1.37$	$\underline{\mathbf{0.66}}/\underline{\mathbf{0.88}}$	$0.69/0.93$
Hotel	$\underline{\mathbf{0.52}}/\underline{\mathbf{1.03}}$	$0.53/1.08$	$0.62/1.26$	$0.58/1.16$	$0.60/1.24$	$0.62/1.30$
Univ	$\underline{\mathbf{0.68}}/1.49$	$0.69/\underline{\mathbf{1.46}}$	$0.69/1.49$	$0.71/1.51$	$0.72/1.53$	$\underline{\mathbf{0.68}}/\underline{\mathbf{1.46}}$
Zara	$0.53/1.21$	$0.54/1.18$	$0.51/1.12$	$0.51/1.10$	$\underline{\mathbf{0.50}}/\underline{\mathbf{1.09}}$	$0.51/1.11$
Average	$0.62/1.19$	$0.66/1.24$	$0.64/1.22$	$0.68/1.29$	$\underline{\mathbf{0.62}}/\underline{\mathbf{1.18}}$	$\underline{\mathbf{0.62}}/1.20$
Weighted Average	$\underline{\mathbf{0.63}}/1.38$	$0.64/1.36$	$0.64/1.37$	$0.64/1.38$	$0.65/1.39$	$\underline{\mathbf{0.63}}/\underline{\mathbf{1.35}}$

Table 4.8: Comparison of baselines (first 2) with interaction-aware models (last 4). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 11$. Displacement errors are in *metres*. Lowest error (best model) is shown in **bold and underline**.

Scene	CV ADE / FDE	LSTM ADE / FDE	Arc-LSTM ADE / FDE	V-Jn-Arc-LSTM ADE / FDE	V-ND-Arc-LSTM ADE / FDE	V-LSTM-Arc-LSTM ADE / FDE
ETH	$0.42/0.76$	$0.40/0.72$	$0.40/0.69$	$0.36/0.64$	$0.37/0.66$	$\underline{\mathbf{0.35}}/\underline{\mathbf{0.61}}$
Hotel	$\underline{\mathbf{0.25}}/\underline{\mathbf{0.45}}$	$\underline{\mathbf{0.25}}/0.46$	$0.26/0.47$	$0.26/0.46$	$0.27/0.48$	$\underline{\mathbf{0.25}}/\underline{\mathbf{0.45}}$
Univ	$\underline{\mathbf{0.28}}/0.57$	$\underline{\mathbf{0.28}}/0.57$	$0.29/0.58$	$0.29/0.58$	$0.30/0.59$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.56}}$
Zara	$0.22/0.46$	$0.23/\underline{\mathbf{0.43}}$	$0.24/0.48$	$0.22/0.45$	$0.23/0.47$	$\underline{\mathbf{0.21}}/\underline{\mathbf{0.43}}$
Average	$0.29/0.56$	$0.29/0.55$	$0.30/0.55$	$0.29/0.53$	$0.29/0.55$	$\underline{\mathbf{0.27}}/\underline{\mathbf{0.51}}$
Weighted Average	$0.27/0.54$	$0.27/0.54$	$0.28/0.55$	$0.27/0.54$	$0.28/0.55$	$\underline{\mathbf{0.26}}/\underline{\mathbf{0.52}}$

Overall, the 4 interaction-aware variants do not provide a significant improvement over CV and the base LSTM. Some of the interaction-aware models have slightly worse results, both for $L = 21$ and $L = 11$. Overall, only V-LSTM-Arc-LSTM is consistently better than the base LSTM. The other variable arc shape configurations (Jn and ND) have similar or worse performance, showing that simple handcrafted solutions fail to generalize to the diverse types of interactions and their influence on a pedestrian's FOA.

Most interaction-aware models have worse performance in Univ scene, which is the most crowded. The same models have better results for ETH and Zara scenes, which have social interactions, although less diverse. This highlights a limitation of the interaction-aware models of section 3.5: since the models are not directly trained in any way to generate socially acceptable trajectories, the presence of neighbours might not actually be used to improve the predictions. Therefore, these models may not be fit to handle the diverse interactions present in Univ scene.

The analysis using plain ADE and FDE metrics gives no insight of the social accuracy of the predictions. Tables 4.9 and 4.10 contain the percentages of collisions between pedestrians - Col-P and Col-GT as defined in section 4.2.2 - for the same data of tables 4.7 and 4.8, respectively. There are overall more collisions in Univ (the most crowded scene), followed by Zara. Less crowded scenes, ETH and Hotel, have even less collisions, due to there being less proximity between pedestrians.

Table 4.9: Interaction-centric evaluation with pedestrian collision metric, in percentage (%). Comparison of baselines (first 2) with interaction-aware models (last 4). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Lowest collisions (best model) shown in **bold and underline**.

Scene	CV Col-P/GT	LSTM Col-P/GT	Arc-LSTM Col-P/GT	V-Jn-Arc-LSTM Col-P/GT	V-ND-Arc-LSTM Col-P/GT	V-LSTM-Arc-LSTM Col-P/GT
ETH	**0**/**0**	**0**/**0**	**0**/**0**	**0**/**0**	**0**/**0**	**0**/**0**
Hotel	**3.2**/**0**	9.7/**0**	**3.2**/**0**	**3.2**/**0**	9.7/3.2	5.8/**0**
Univ	13.9/13.2	12.4/11.8	**8.7**/**10.7**	10.7/13.4	10.3/12.4	9.8/12.0
Zara	6.2/9.9	7.4/9.1	6.2/6.6	**5.4**/8.2	7.4/8.2	**5.4**/**6.1**
Average	5.8/5.8	7.4/5.2	**4.5**/**4.3**	4.8/5.4	6.9/6.0	5.2/4.5
Weighted Average	11.1/11.6	10.8/10.5	**7.7**/**9.0**	8.8/11.3	9.4/10.8	8.3/9.7

Table 4.10: Interaction-centric evaluation with pedestrian collision metric, in percentage (%). Comparison of baselines (first 2) with interaction-aware models (last 4). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 11$. Lowest collisions (best model) shown in **bold and underline**.

Scene	CV Col-P/GT	LSTM Col-P/GT	Arc-LSTM Col-P/GT	V-Jn-Arc-LSTM Col-P/GT	V-ND-Arc-LSTM Col-P/GT	V-LSTM-Arc-LSTM Col-P/GT
ETH	1.2/3.5	1.2/3.5	1.2/3.5	**0**/2.3	1.2/1.2	0.5/**0.2**
Hotel	3.2/2.1	3.2/**0**	**2.1**/**0**	4.2/**0**	**2.1**/1.1	4.2/**0**
Univ	3.7/**2.5**	**2.7**/2.8	2.9/3.4	3.4/3.2	3.7/3.2	4.1/3.0
Zara	2.5/**1.5**	1.5/2.8	1.5/2.6	**1.1**/1.9	**1.1**/2.8	1.6/1.6
Average	2.6/2.4	2.1/2.3	**1.9**/2.4	2.1/1.9	2.0/2.1	2.6/**1.2**
Weighted Average	3.2/**2.3**	**2.3**/2.7	2.4/3.0	2.6/2.6	2.7/2.8	3.2/**2.3**

The interaction-aware models give a noticeable reduction in the collisions for $L = 21$ (up to $4\% - 5\%$ reduction for Arc-LSTM), but fail to reduce the number of collisions for $L = 11$, when compared to the base LSTM. This shows that while the social models may improve the social accuracy of the predictions, it still has limitations, in the same way it gives worse ADE and FDE than the base LSTM.

The variable arc shape models have an increased number of collisions when compared to Arc-LSTM. Once again, this highlights the difficulty in increasing social accuracy while also maintaining the same geometric errors. Even though the V-LSTM-Arc-LSTM model has lower ADE and FDE than Arc-LSTM, it was not chosen for integration with SMF in section 3.6 because of the higher rate of collisions, along the extra complexity of using a separate LSTM to learn the right arc dimensions to use.

4.5.2 Qualitative results

Similarly to section 4.4, it is important to get a different perspective on the impact of including social context for forecasting pedestrian trajectories. This will involve visualizing and comparing the Arc-LSTM model with the base LSTM that does not consider any neighbours to the pedestrian. Like section 4.4.2, only a few situations are shown, to analyze some strengths and limitations of Arc-LSTM.

An example of a situation where Arc-LSTM correctly learns a shift the primary pedestrian's direction is shown in fig. 4.8. At the fourth prediction instant, $t = 13$ (fig. 4.8(a)), LSTM and Arc-LSTM predictions are very similar to GT. There are some neighbours that are not moving (small black dots), one of which is in the FOV of the primary pedestrian. The Arc-LSTM generates a curve prediction, which can be seen in instant $t = 17$ (fig. 4.8(b)), in order to maintain a certain distance from that neighbour. The base LSTM predicts a more linear trajectory, resulting in a collision with the neighbour.

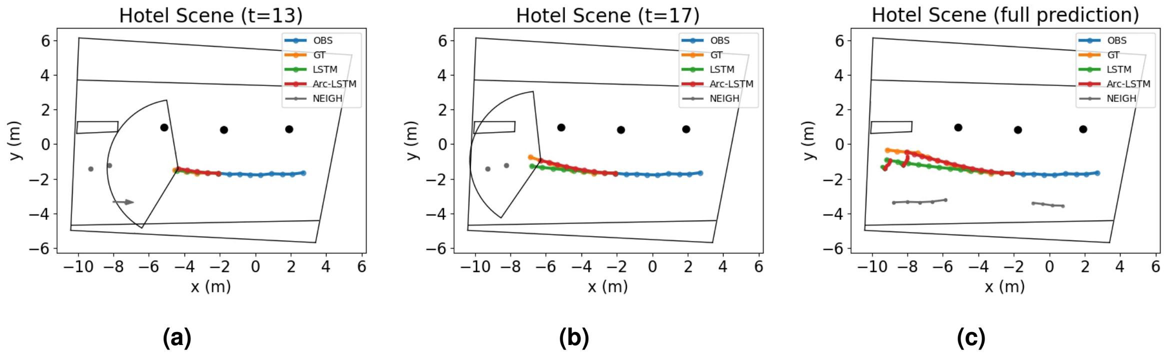
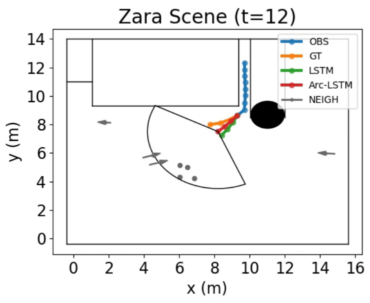


Figure 4.8: Prediction of a trajectory of length $L = 21$ with social interactions, in Hotel scene. There are predictions of 2 models: LSTM in green, Arc-LSTM in red. Past trajectory (OBS) and GT of primary pedestrian (GT) in blue and orange, respectively. The neighbour motion/trajectory (NEIGH) is shown in gray. The primary pedestrian's FOV is the arc shape in black. **(a)** Snapshot at instant $t = 13$. The real motion (not the whole trajectory) of neighbours is shown. **(b)** Snapshot at instant $t = 17$. **(c)** Complete prediction of the trajectory for primary pedestrian and some neighbours (full trajectories).

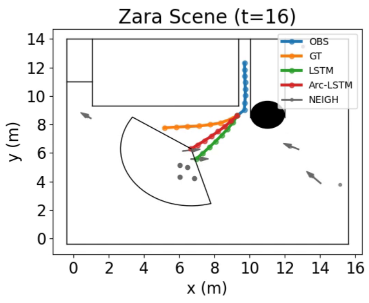
The above situation was simple, belonging to a scene with very low crowd density (Hotel). A more crowded and complicated situation is shown in fig. 4.9.

Most of the past trajectory of the past pedestrian is linear, with only the final instant showing signs of a curve. The GT trajectory curves left, in part to dodge neighbours, but also to in line with the pedestrians goal - to go to Zara store, in top-left corner of fig. 4.9(a). The Arc-LSTM predicts a change of direction closer to GT than the base LSTM one, but it is still fairly distant from it. At $t = 12$, there are several neighbours in the FOV of the primary pedestrian. Some are stationary, others are moving directly at the primary pedestrian. However, the Arc-LSTM doesn't predict a change of direction to avoid colliding with the moving neighbours, as seen in fig. 4.9(b). The Arc-LSTM is able to correctly avoid direct collisions with stationary neighbours, which is an improvement to the base LSTM, but can fail to predict collision avoidance with moving neighbours, since the relative velocities can vary from instant to instant.

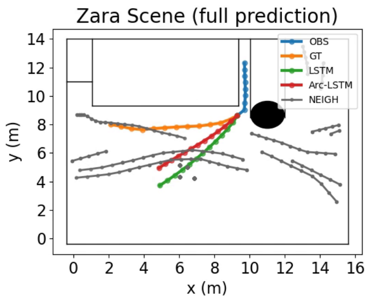
Overall, both the quantitative results of section 4.5.1 and the qualitative results of this section show that there is benefit in considering the existence of social interactions with LSTM models. However, these models are still prone to social inaccuracy. One of the main causes that can be attributed to this is



(a)



(b)



(c)

Figure 4.9: Prediction of a trajectory of length $L = 21$ with social interactions, in Zara scene. There are predictions of 2 models: LSTM in green, Arc-LSTM in red. Past trajectory (OBS) and GT of primary pedestrian (GT) in blue and orange, respectively. The neighbour motion/trajectory (NEIGH) is shown in gray. The primary pedestrian's FOV is the arc shape in black. **(a)** Snapshot at instant $t = 12$. The real motion (not the whole trajectory) of neighbours is shown. **(b)** Snapshot at instant $t = 16$. **(c)** Complete prediction of the trajectory for primary pedestrian, with real full trajectory (past and future) for neighbours.

the fact that the models are not being directly trained to improve the social aspects of pedestrian motion - or be penalized when such aspects are not being followed.

4.5.3 Benchmarking interaction-aware models using the original Trajnet++

To get a more robust view on the performance of the interaction-aware models of section 3.5, these were evaluated using the original Trajnet++ benchmark [26][4], which contains additional datasets with diverse social interactions [58–60]. Furthermore, these models were benchmarked against some state-of-the-art models, mentioned in section 4.3. The results of these experiments can be seen in table 4.11.

Table 4.11: Comparison of recent baseline methods (first 3) with the models proposed in section 3.5 (last 4). Use of the real Trajnet++ benchmark [26], with original length $L = 21$. Displacement errors are in *metres*. Social metrics (Col-P and Col-GT) are in percentage. Lowest values (best model) is shown in **bold and underline**.

Metrics	Base LSTM	D-LSTM [26]	S-LSTM [11]	Arc-LSTM	V-Jn-Arc-LSTM	V-ND-Arc-LSTM	V-LSTM-Arc-LSTM
ADE/FDE (m)	0.60/1.30	0.56/1.22	**0.53**/**1.14**	0.55/1.19	0.56/1.20	0.57/1.22	0.57/1.22
Col-P/GT (%)	13.6/14.8	**5.4**/13.0	6.7/13.5	6.7/13.2	7.2/12.8	10.7/**12.1**	11.0/12.5

The proposed Arc-LSTM, as well as the models that vary the pooling shape, are able to outperform a base LSTM network. They have competitive results with other interaction-aware models, such as D-LSTM and S-LSTM. However, the proposed models have higher displacement errors than S-LSTM, and higher Col-P than D-LSTM. It is worth mentioning that the model hyper-parameters were not optimized

[4]Submissions made to `https://www.aicrowd.com/challenges/trajnet-a-trajectory-forecasting-challenge/`

for Trajnet++ data. If they had been, the results could have improved slightly. Nonetheless, these results show that the use of a FOV pooling with LSTM networks can be a solid alternative to incorporating social interactions. Further comparison with the state-of-the-art will be conducted in the next section.

4.6 Scene and social evaluation with Arc-LSTM-SMF model

Having evaluated the performance of the scene-aware models in section 4.4, and the interaction-aware models in section 4.5, this section concerns the evaluation and discussion of the results obtained for the combination of those parts. This section will feature the experiments conducted with the Arc-LSTM-SMF model described in section 3.6. Recall that this is the model that satisfies the main goal of the thesis - to have the predicted trajectories be influenced by both scene and social aspects.

The Arc-LSTM-SMF will be compared with several state-of-the-art works - mentioned in section 4.3), trained in the same conditions and data. These models are S-LSTM, D-LSTM, and S-GAN[5]. Results will also be shown for the scene and social standalone parts used to build the Arc-LSTM-SMF model. These are the scene-specific LSTM-SMF model, and the socially-aware Arc-LSTM. The objective is to see if there is a clear advantage in combining the two parts.

4.6.1 Comparison with state-of-the-art: Quantitative results

First, the geometric performance will be compared between the proposed model, its standalone parts, and the baselines mentioned at the start of this section. The results can be seen for trajectory lengths $L = 21$ in table 4.12, and for $L = 11$ in table 4.13.

Table 4.12: Comparing geometric errors between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. ADE and FDE computed for primary pedestrians, in *metres*. Lowest error (best model) shown in **bold and underline**.

Scene	D-LSTM [26] ADE / FDE	S-LSTM [11] ADE / FDE	S-GAN [8] ADE / FDE	LSTM-SMF-II ADE / FDE	Arc-LSTM ADE / FDE	Arc-LSTM-SMF ADE / FDE
ETH	$1.14/1.69$	$1.18/1.74$	$0.75/0.91$	$\mathbf{0.54}/0.84$	$0.73/1.01$	$0.67/\mathbf{0.73}$
Hotel	$0.62/1.26$	$0.50/0.95$	$0.47/0.93$	$\mathbf{0.43}/\mathbf{0.72}$	$0.62/1.26$	$0.47/0.79$
Univ	$0.74/1.60$	$0.69/1.47$	$0.73/1.49$	$\mathbf{0.67}/\mathbf{1.43}$	$0.69/1.49$	$\mathbf{0.67}/1.44$
Zara	$0.50/1.09$	$\mathbf{0.49}/\mathbf{1.07}$	$0.55/1.14$	$0.54/1.12$	$0.51/1.12$	$0.54/1.13$
Average	$0.75/1.41$	$0.71/1.31$	$0.62/1.12$	$\mathbf{0.55}/1.03$	$0.64/1.37$	$0.59/\mathbf{1.02}$
Weighted Average	$0.66/1.44$	$\mathbf{0.62}/1.33$	$0.67/1.36$	$0.63/\mathbf{1.31}$	$0.64/1.37$	$\mathbf{0.62}/1.32$

[5]The S-GAN used in this section is deterministic, *i.e.*, trained without variety loss. It gave better results that the same model with variety loss. This is because the evaluation conducted so far concerns using a single prediction per pedestrian.

Table 4.13: Comparing geometric errors between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 11$. ADE and FDE computed for primary pedestrians, in *metres*. Lowest error (best model) shown in **<u>bold and underline</u>**.

Scene	D-LSTM [26] ADE / FDE	S-LSTM [11] ADE / FDE	S-GAN [8] ADE / FDE	LSTM-SMF-II ADE / FDE	Arc-LSTM ADE / FDE	Arc-LSTM-SMF ADE / FDE
ETH	$0.39/0.67$	$0.41/0.70$	$0.42/0.73$	$\underline{\mathbf{0.38}}/\underline{\mathbf{0.64}}$	$0.40/0.69$	$0.39/0.67$
Hotel	$0.27/0.48$	$0.29/0.52$	$0.27/0.49$	$\underline{\mathbf{0.24}}/\underline{\mathbf{0.41}}$	$0.26/0.47$	$0.28/0.47$
Univ	$0.30/0.60$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.56}}$	$0.33/0.65$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.56}}$	$0.29/0.58$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.56}}$
Zara	$0.23/0.46$	$0.23/0.46$	$0.25/0.49$	$\underline{\mathbf{0.21}}/\underline{\mathbf{0.42}}$	$0.24/0.48$	$0.23/0.46$
Average	$0.30/0.55$	$0.30/0.56$	$0.32/0.59$	$\underline{\mathbf{0.28}}/\underline{\mathbf{0.51}}$	$0.30/0.55$	$0.29/0.54$
Weighted Average	$0.28/0.56$	$0.23/0.53$	$0.31/0.60$	$\underline{\mathbf{0.26}}/\underline{\mathbf{0.52}}$	$0.28/0.55$	$0.27/0.53$

The Arc-LSTM-SMF model has competitive ADE and FDE values with state-of-the-art models. It is able to outperform D-LSTM and S-GAN for both trajectories of length $L = 21$ (table 4.7) and $L = 11$ (table 4.8). The S-LSTM model has similar performance, but still having larger errors for scenes like ETH and Hotel. The Arc-LSTM-SMF has slightly higher error than LSTM-SMF-II, meaning that the incorporation of arc pooling does not reduce geometric errors, which was already found in section 4.5.1. The only case where Arc-LSTM-SMF is clearly outperformed by the state-of-the-art is in the Zara scene for $L = 21$, where both D-LSTM and S-LSTM have smaller displacement errors. The fact that S-LSTM has smaller errors than D-LSTM is also consistent with the results reported in the Trajnet++ benchmark (table 4.11). Overall, it can be argued that the incorporation of LSTM with SMF and arc-shaped pooling gives a relevant performance boost, in terms of ADE and FDE, when compared with the state-of-the-art.

Regarding scene compliance, the same models were evaluated using the scene-specific CSE and OSB metrics introduced in section 4.2.1. The results are shown in tables 4.14 and 4.15 for trajectory lengths $L = 21$ and $L = 11$, respectively. Unlike ADE and FDE, the metrics were also computed for the neighbours, due to the lack of scene-relevant trajectories of primary pedestrians (recall that the primary pedestrians are chosen based on social interactions, without any consideration of the scene).

The Arc-LSTM-SMF has the best results in terms of obstacle collisions, as does LSTM-SMF-II , but has limited performance in regard to scene bounds. It is worth recalling that none of the state-of-the-art methods used have any consideration of the scene. In terms of collisions of primary pedestrians and neighbours with the environment (CSE metric), the Arc-LSTM-SMF has the least amount - except for ETH scene and $L = 11$. This shows that including SMF predictions improves the scene compliance of the model, even when including the arc pooling layer. The Arc-LSTM-SMF gets outperformed in OSB. S-GAN has around half of the number of trajectories going out of scene bounds than Arc-LSTM-SMF. This can be due to the adversarial training of S-GAN, that is not present in the models of chapter 3. The concept of scene bounds is also not being learnt by the LSTM-SMF-II model. However, as argued

Table 4.14: Comparing scene compliance between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. CSE and OSB computed for primary pedestrians and neighbours, in percentage (%). Lowest error (best model) shown in **bold and underline**.

Scene	D-LSTM [26] CSE / OSB	S-LSTM [11] CSE / OSB	S-GAN [8] CSE / OSB	LSTM-SMF-II CSE / OSB	Arc-LSTM CSE / OSB	Arc-LSTM-SMF CSE / OSB
ETH	**0**/**13.3**	6.7/**13.3**	6.7/**13.3**	**0**/**13.3**	6.7/**13.3**	**0**/**13.3**
Hotel	1.0/16.5	2.2/13.7	0.6/**10.2**	**0**/12.7	1.3/14.3	**0**/15.9
Univ	−/12.8	−/10.6	−/**5.3**	−/12.3	−/12.9	−/12.2
Zara	1.1/13.8	0.9/13.1	0.9/**11.0**	**0.4**/12.8	0.7/13.6	**0.4**/12.1
Average	0.7/14.1	2.5/12.7	2.1/**9.9**	**0.1**/12.8	2.2/13.5	**0.1**/13.4
Weighted Average	1.1/12.9	1.1/10.9	0.9/**6.0**	**0.4**/12.4	0.8/13.0	**0.4**/12.2

Table 4.15: Comparing scene compliance between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 11$. CSE and OSB computed for primary pedestrians and neighbours, in percentage (%). Lowest error (best model) shown in **bold and underline**.

Scene	D-LSTM [26] CSE / OSB	S-LSTM [11] CSE / OSB	S-GAN [8] CSE / OSB	LSTM-SMF-II CSE / OSB	Arc-LSTM CSE / OSB	Arc-LSTM-SMF CSE / OSB
ETH	2.1/7.3	2.9/7.3	**0.7**/**5.1**	1.9/7.3	2.8/7.2	1.9/7.3
Hotel	0.1/6.4	0.7/6.4	0.1/6.0	**0**/6.6	0.1/6.6	0.1/**5.9**
Univ	−/6.0	−/5.8	−/**3.2**	−/6.1	−/6.0	−/5.7
Zara	0.7/8.7	0.6/8.5	0.8/**6.7**	**0.1**/8.6	0.2/8.3	**0.1**/8.3
Average	1.0/7.1	1.4/7.0	**0.5**/**5.3**	0.7/6.2	1.0/7.0	0.7/6.8
Weighted Average	0.7/6.4	0.8/6.1	0.8/**3.7**	**0.2**/6.5	0.4/6.3	0.3/6.0

in section 4.2.1, the OSB metric is not as important as CSE. Overall, the Arc-LSTM-SMF has the smallest values of CSE, therefore it can be argued to be the most scene complaint model.

While the state-of-the-art baselines used in this section are not scene-aware, all of them employ techniques to consider social interactions between pedestrians. An interaction-centric evaluation will be especially relevant to assess the social accuracy of Arc-LSTM-SMF. Such evaluation will be done with the Col-P and Col-GT metrics from section 4.2.2. The results are presented in tables 4.16 and 4.17, for primary pedestrian trajectory lengths $L = 21$ and $L = 11$, respectively.

The Arc-LSTM-SMF model has the lowest amount of collisions for its predicted trajectories. The models that comes closest are D-LSTM, Arc-LSTM and LSTM-SMF-II. The social accuracy of the LSTM-SMF-II method may be surprising, since it does not consider social interactions. It seems that just by having a scene-specific model (with interactions specific to that scene) can reduce the collisions by the same level that including a FOV-shaped pooling would. Nonetheless, the Arc-LSTM-SMF is able to have less collisions than its standalone parts, especially for $L = 11$ (table 4.17). The S-LSTM and S-GAN

Table 4.16: Comparing social accuracy between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Col-P and Col-GT computed for primary pedestrians, in percentage (%). Lowest error (best model) shown in **bold and underline**.

Scene	D-LSTM [26] Col-P / GT	S-LSTM [11] Col-P / GT	S-GAN [8] Col-P / GT	LSTM-SMF-II Col-P / GT	Arc-LSTM Col-P / GT	Arc-LSTM-SMF Col-P / GT
ETH	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$
Hotel	$3.2/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$3.22/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$
Univ	$\underline{\mathbf{7.9}}/12.4$	$13.2/11.5$	$15.0/13.5$	$9.4/10.5$	$8.7/\underline{\mathbf{10.7}}$	$8.8/11.8$
Zara	$7.8/10.7$	$5.4/8.2$	$9.5/7.8$	$\underline{\mathbf{4.5}}/10.3$	$6.2/\underline{\mathbf{6.6}}$	$4.9/7.8$
Average	$4.7/5.8$	$4.6/4.9$	$6.1/5.3$	$3.5/5.2$	$4.5/\underline{\mathbf{4.3}}$	$\underline{\mathbf{3.4}}/4.9$
Weighted Average	$7.7/11.4$	$10.3/9.0$	$12.7/11.3$	$7.5/10.0$	$7.7/\underline{\mathbf{9.0}}$	$\underline{\mathbf{7.3}}/10.1$

Table 4.17: Comparing social accuracy between Arc-LSTM-SMF model (last column) and several state-of-the-art trajectory forecasting models (second to fourth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 11$. Col-P and Col-GT computed for primary pedestrians, in percentage (%). Lowest error (best model) shown in **bold and underline**.

Scene	D-LSTM [26] Col-P / GT	S-LSTM [11] Col-P / GT	S-GAN [8] Col-P / GT	LSTM-SMF-II Col-P / GT	Arc-LSTM Col-P / GT	Arc-LSTM-SMF Col-P / GT
ETH	$2.3/2.3$	$\underline{\mathbf{0}}/1.2$	$2.3/\underline{\mathbf{0}}$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$1.2/3.5$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$
Hotel	$3.2/1.1$	$2.1/0$	$\underline{\mathbf{0}}/3.2$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$	$2.1/0$	$\underline{\mathbf{0}}/\underline{\mathbf{0}}$
Univ	$3.5/\underline{\mathbf{2.5}}$	$4.5/\underline{\mathbf{2.5}}$	$5.8/4.6$	$3.4/3.1$	$\underline{\mathbf{2.9}}/3.4$	$3.1/2.7$
Zara	$2.3/1.7$	$2.3/2.1$	$2.3/2.5$	$1.9/1.9$	$1.5/2.6$	$\underline{\mathbf{0.6}}/\underline{\mathbf{1.5}}$
Average	$2.8/1.9$	$2.2/1.4$	$2.6/2.6$	$1.3/1.2$	$1.9/2.4$	$\underline{\mathbf{0.9}}/\underline{\mathbf{1.1}}$
Weighted Average	$3.0/\underline{\mathbf{2.1}}$	$3.5/2.2$	$4.3/3.7$	$2.6/2.4$	$2.4/3.0$	$\underline{\mathbf{2.0}}/\underline{\mathbf{2.1}}$

have higher percentages of Col-P, although S-LSTM has similar values for Col-GT. This was already hinted by the results with the original Trajnet++ (table 4.11), where D-LSTM had a reduced number of collisions when compared to S-LSTM. The S-GAN has a considerably higher number of collisions than a base LSTM model, for both $L = 21$ and $L = 11$ (see tables 4.9 and 4.10). This reinforces the difficulty in learning social interactions, especially since none of these models are explicitly trained to learn social concepts. Although the Arc-LSTM-SMF also experiences some of that difficulty, its arc pooling coupled with the fact that it is trained for each of the scenes helps to generate more socially accurate trajectories. This will be further corroborated with a qualitative evaluation.

4.6.2 Comparison with state-of-the-art: Qualitative results

Having quantitatively proven that the Arc-LSTM-SMF has competitive performance with state-of-the-art trajectory forecasting models, the next step is to provide further insight on the matter, using a qualitative evaluation. First, in fig. 4.10, there is a visual comparison of an Arc-LSTM-SMF prediction with the predictions of its standalone parts.

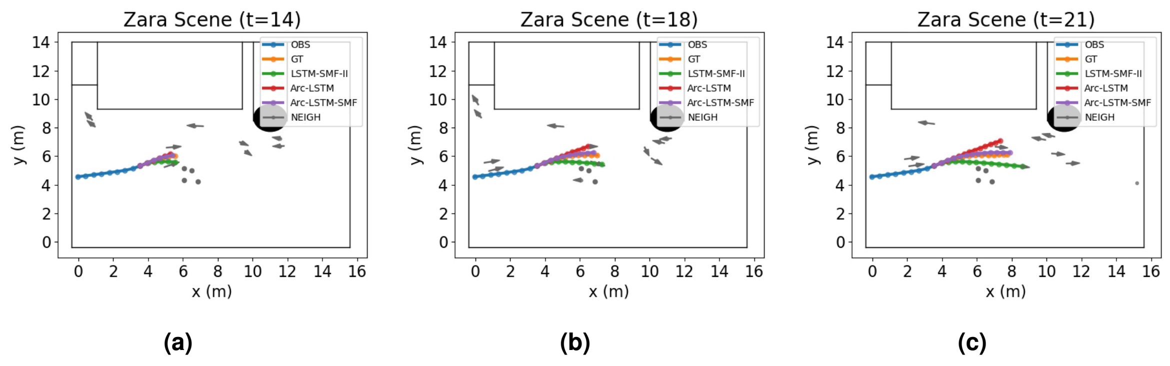


Figure 4.10: Example of a situation comparing Arc-LSTM-SMF (purple) with its standalone parts: LSTM-SMF-II (green) and Arc-LSTM (red). Past trajectory (OBS) and GT of primary pedestrian (GT) in blue and orange, respectively. The neighbour motion (NEIGH) is shown in gray. **(a)** Snapshot at instant $t = 14$. **(b)** Snapshot at instant $t = 18$. **(c)** Snapshot at final instant $t = 21$.

As seen on fig. 4.10(c), the Arc-LSTM-SMF is able to provide the best prediction out of the three models. It avoids the stationary pedestrians, which LSTM-SMF-II does not (fig. 4.10(b)), and curves to avoid colliding with a moving neighbour, something that Arc-LSTM fails to do (fig. 4.10(b)). Overall, the combination of SMF with arc pooling aids the Arc-LSTM-SMF in generating curved predictions.

The remainder of this section features visualization of several predictions from the Arc-LSTM-SMF model, comparing them with state-of-the-art baselines. Regarding scene compliance, some predictions of trajectories are shown in fig. 4.11, without consideration of neighbours.

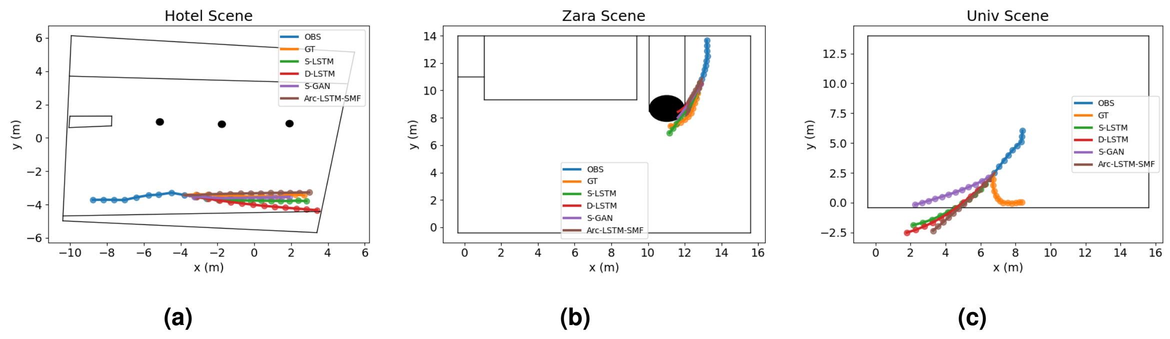


Figure 4.11: Examples of 3 situations to highlight the scene compliance of 4 models: S-LSTM (green), D-LSTM (red), S-GAN (purple), and Arc-LSTM-SMF (brown). Past trajectory (OBS) and GT of primary pedestrian (GT) in blue and orange, respectively. **(a)** Hotel scene. **(b)** Zara scene. **(c)** Univ scene.

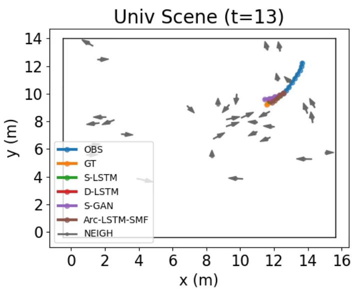
The first situation, in fig. 4.11(a), is fairly simple. The Arc-LSTM-SMF, having access to the scene-specific SMF, was able to predict the most accurate speed and direction of motion. While the predictions of the baselines were not necessarily bad, their direction or speed was less close to the reality. In the case of D-LSTM, it resulted in a near-collision with a building (see fig. 4.2(a)). In the situation from Zara scene (fig. 4.11(b)), the pedestrian is quite close to a physical obstacle. All baselines predict a trajectory that collides with that obstacle. The Arc-LSTM-SMF prediction is the only one that strafes away from the obstacle, even though its speed is smaller than the GT's, meaning geometrically, it is not the best prediction. The case in fig. 4.11(c) shows an already mentioned limitation of the proposed model: the inability to restrict predictions to the scene bounds. The prediction from Arc-LSTM-SMF, as well as from S-LSTM D-LSTM go far beyond the bounds of the scene. The S-GAN prediction, while not being close to the GT, is inside the scene bounds. A possible line of research could be to integrate the concepts proposed in chapter 3 with other kinds of ANN architectures, such as GANs. It could help with complying to scene bounds, while still maintaining a low number of collisions with obstacles.

The next step is to get a perception of the social accuracy of model predictions. This involves also looking at the trajectory - or motion - of neighbouring pedestrians. Four situations from the more crowded scenes of the test set [20] are shown in fig. 4.12.

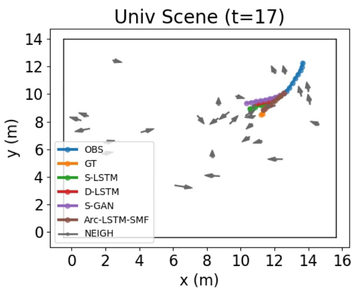
The first situation, from Univ scene (the most crowded scene), in figs. 4.12(a) to 4.12(c), highlights an advantage of the Arc-LSTM-SMF model: only considering the neighbours in front of the pedestrian. As seen in fig. 4.12(a), there is a considerable number of neighbours in front of the pedestrian. Using a grid-based pooling technique such as that of S-LSTM D-LSTM will end up including even more neighbours, many of which are not likely to influence the primary pedestrian's motion. As such, the baseline methods generate predictions that stray away from the GT, and even almost collide with neighbours (figs. 4.12(b) and 4.12(c)). The situation from figs. 4.12(d) to 4.12(f) is much less crowded, but still quite relevant. As seen in fig. 4.12(d), there are two neighbours that are nearly stationary (notice the arrows with little magnitude), with the past trajectory moving directly at them. All predictions from the baselines collide with these neighbours, while the Arc-LSTM-SMF curves to avoid this, matching the GT. The Arc-LSTM already had the ability to avoid stationary neighbours (figs. 4.8 and 4.9), and these results show the Arc-LSTM-SMF maintains this ability.

The third situation, in figs. 4.12(g) to 4.12(i), is more complicated. The past trajectory has a very small speed, especially when compared to the GT. While none of the methods have an accurate prediction of the pedestrian's speed, the direction of the Arc-LSTM-SMF prediction is the only one that stands closer to the GT direction. All other predictions end up going in a straight line, resulting in a collision with a neighbour at $t = 16$ (fig. 4.12(h)). Nonetheless, the model also has inaccurate predictions in some situations. For the case of figs. 4.12(j) to 4.12(l), the Arc-LSTM-SMF prediction is the most disparate from the GT. Having several neighbours moving towards the primary pedestrian (fig. 4.12(j)), the Arc-LSTM-SMF failed to predict the proper direction, resulting in a collision with a neighbour (fig. 4.12(k)).

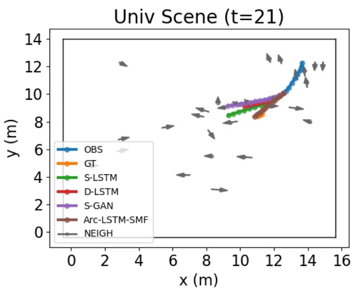
While the Arc-LSTM-SMF model has some limitations, mainly regarding social aspects, this qualitative evaluation has shown that the same limitations are present, possibly to an even greater extent, in the state-of-the-art methods. Overall, the quantitative and qualitative results have shown that there are clear



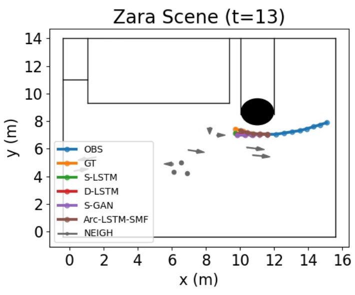
(a)



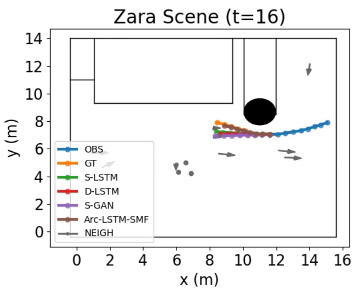
(b)



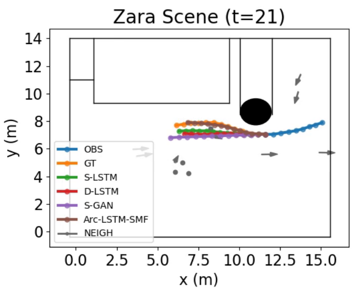
(c)



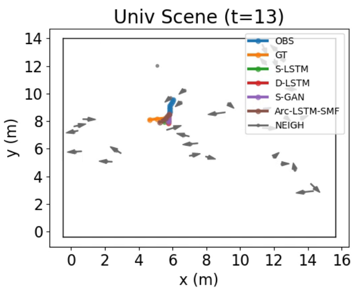
(d)



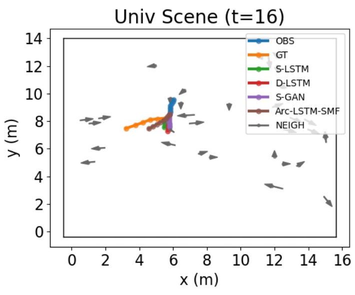
(e)



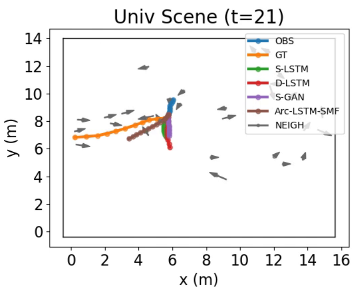
(f)



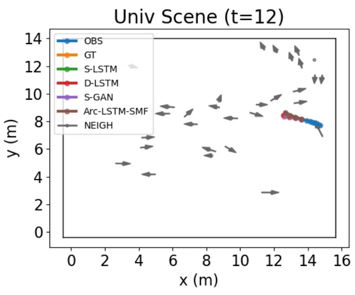
(g)



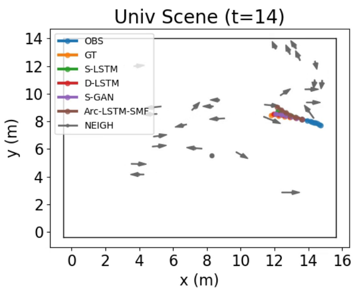
(h)



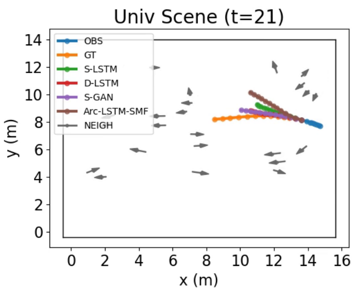
(i)



(j)



(k)



(l)

Figure 4.12: Examples of 4 social situations. There are predictions of 4 models: S-LSTM in green, D-LSTM in red, S-GAN in purple, and Arc-LSTM-SMF in brown. Past trajectory (OBS) and GT of primary pedestrian (GT) in blue and orange, respectively. The neighbour motion (NEIGH) is shown in gray. **(a), (b), (c)** First situation, $t \in \{13, 17, 21\}$. **(d), (e), (f)** Second situation, $t \in \{13, 16, 21\}$. **(g), (h), (i)** Third situation, $t \in \{13, 16, 21\}$. **(j), (k), (l)** Fourth situation, $t \in \{12, 14, 21\}$.

advantages with the Arc-LSTM-SMF model of pedestrian trajectory forecasting. The advantages include robust performance in unseen trajectories - lowest ADE and FDE -, lowest percentage of collisions with obstacles, and competitive social accuracy when compared to the state-of-the-art.

4.7 Multimodal evaluation using Arc-LSTM-SMF model

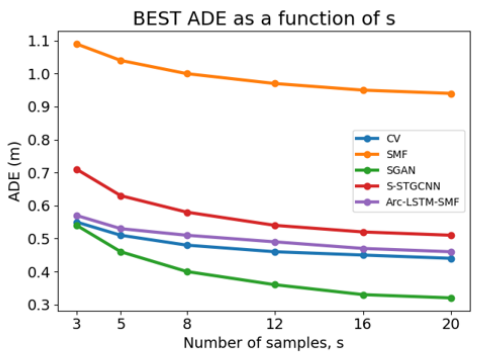
So far, all experiments conducted in the previous sections involved the use of a single predicted trajectory per pedestrian. In other words, only a unimodal evaluation was done using the proposed model, along with state-of-the-art baselines. Generating more than one prediction per pedestrian means the forecasting model needs to implicitly or explicitly account for uncertainty in its prediction(s). That is the case of the Arc-LSTM-SMF model, that besides outputting a 2D mean for pedestrian displacement, it also outputs a standard deviation and correlation factor. With these values, it is possible to sample several predictions by sampling from the bi-variate Gaussian distributions outputted from the model.

This section will feature quantitative results from a multimodal evaluation using the Arc-LSTM-SMF model and the following baselines:

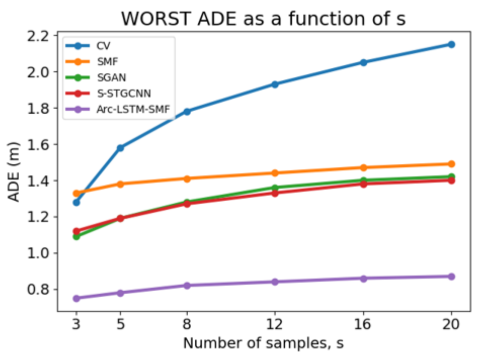
- CV - a multimodal constant velocity model from [69], where the speed is constant, but the direction is rotated by an angle sampled from a Gaussian distribution with zero mean and $\sigma = 25°$.
- SMF - the multimodal version of SMF [6] (see section 3.3), that incorporates white noise.
- S-GAN - the original version of Social GAN [8], trained with variety loss with $K = 20$ samples.
- S-STGCNN - the same multimodal model mentioned in section 4.3.

To conduct this multimodal evaluation, there is a need to use different metrics from the ones used thus far. The experiments of this section will use the metrics mentioned in section 2.5. First, the models will be compared by using the bestOf-s-ADE metric, *i.e.*, by selecting the sample - from a total of s samples - that results in the lowest ADE. Instead of just computing this metric for a total of $s = 20$ samples, a range of values for s will be used, to better understand the multimodal performance of the models. These results are shown in fig. 4.13(a), for the same test set, with trajectory length $L = 21$. Since bestOf-s-ADE give no information of the other $s - 1$ samples, results are also shown for worstOf-s-ADE, in fig. 4.13(b). The worstOf-s-ADE metric is computed by selecting the sample that yields the highest value of ADE.

The model with lowest bestOf-s-ADE is S-GAN, which shows that its use of variety loss can result in generating samples that are very close to the GT. Followed by S-GAN, is the multimodal CV and the proposed Arc-LSTM-SMF. There is a noticeable difference in how the bestOf-s-ADE evolves with s. For CV and Arc-LSTM-SMF, the difference between bestOf-3-ADE and bestOf-20-ADE is below $0.1\,\mathrm{m}$,



(a)



(b)

Figure 4.13: Multimodal evaluation with number of samples $s \in \{3, 5, 8, 12, 16, 20\}$. Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Lower is better. Multimodal models: CV in blue, SMF in orange, S-GAN in green, S-STGCNN in red, and Arc-LSTM-SMF in purple. The two plots are not at the same scale, to clearly distinguish each of the lines. **(a)** BestOf-s-ADE, *i.e.*, using the sample with the lowest ADE. **(b)** WorstOf-s-ADE, *i.e.*, using the sample with the highest ADE.

while for S-GAN that difference is around $0.2\,\mathrm{m}$, meaning there is clear benefit in increasing s for this method. The S-STGCNN has has the second highest value of bestOf-s-ADE. While this contrasts with its original results [33], it is worth mentioning the data configuration is considerably different. The SMF has the highest values for bestOf-s-ADE, which is consistent with its unimodal performance (table 4.3).

The worstOf-s-ADE metric gives a different insight on the performance of the multimodal models. Some of the models that have the lowest value of bestOf-s-ADE have the highest value of worstOf-s-ADE. Such is the case of the multimodal CV, which had a bestOf-s-ADE always below $0.6\,\mathrm{m}$, but worstOf-s-ADE goes from $1.3\,\mathrm{m}$ for $s = 3$ (more than double) to $2.2\,\mathrm{m}$ for $s = 20$ (more than quadruple). This shows the simple CV baseline, while being able to generate accurate samples, it can also generate samples that deviate considerably from the reality. The S-GAN is not the method with the lowest values of worstOf-s-ADE, having near-identical results to S-STGCNN. The Arc-LSTM-SMF model, even though it doesn't have the smallest bestOf-s-ADE, it has the lowest value of worstOf-s-ADE. This means that Gaussian distribution learnt by the model is less dispersed, resulting in the predictions deviating less from the GT. The same can said about the SMF model, with the worstOf-s-ADE being on average just 143% of bestOf-s-ADE (162% for Arc-LSTM-SMF).

Both the bestOf-s-ADE and worstOf-s-ADE cannot give a robust view on the performance of multimodal models, since they only look at one of the generated samples. To complement the previous experiments, a different multimodal metric, mentioned in section 2.5, is employed: a KDE-NLL. This metric has two steps. The first step (KDE) involves estimating the distribution of the s samples generated by the multimodal model, using a Kernel Density Estimate [70]. A larger number of samples is required to properly estimate the distribution. The value $s = 50$ was chosen, which was also used by [26]. In the second step, the estimated distribution is evaluated at the GT points, and then the symmetrical of the logarithm is computed from the likelihood (NLL). The lower this metric is, the better are the estimated distributions when compared with the GT. The results can be seen in table 4.18, for the same five models evaluated in this section, and with the same test set (trajectory length $L = 21$).

Table 4.18: Comparing the multimodal performance of Arc-LSTM-SMF model (last column) and several state-of-the-art baselines (second to fifth column). Use of Trajnet++ configuration described in section 4.1, with trajectory length $L = 21$. Use of KDE-NLL metric with a total of $s = 50$ predicted trajectory samples per pedestrian. Lowest value (best model) shown in **bold and underline**.

Model	CV	SMF [11]	S-GAN [8]	S-STGCNN [33]	Arc-LSTM-SMF
KDE-NLL	-3.35	**<u>-8.72</u>**	-3.60	-3.31	-5.53

Methods like the multimodal CV and S-GAN, which had the lowest best-Of-ADE in fig. 4.13(a), now have the highest KDE-NLL. This reinforces the fact that the predictions sampled from these models can quite likely deviate from the GT very easily. The S-STGCNN, while the difference between bestOf-s-ADE and bestOf-s-ADE is less pronounced, its distribution is still spread enough to yield a higher value of KDE-NLL. The SMF have the lowest KDE-NLL, followed by Arc-LSTM-SMF. While it may seem counter-intuitive that the method with the highest bestOf-s-ADE has the lowest KDE-NLL, it can be explained by two factors:

1. The probability distribution generated by the SMF is much less spread when compared with methods like S-GAN. This is what gives the model a lowest worstOf-s-ADE to bestOf-s-ADE ratio, and is also (partially) why the method has the lowest KDE-NLL.

2. All unimodal SMF predictions are limited to the bounds of the video image of each scene (recall the 0% value for OSB metric). As such, the multimodal predictions (with uncertainty) will not stray far from the limits of the scene. All dataset trajectories are inside scene bounds, and so the distribution of the SMF predictions is more likely to capture the GT. For the case of Univ scene, for instance, SMF has a KDE-NLL of -10.34, while Arc-LSTM-SMF has a value of -5.88, because it has a large number of trajectories outside scene bounds.

The Arc-LSTM-SMF fails to learn the concept of scene bounds. While not being as relevant as avoiding collisions with obstacles, it impacts the KDE-NLL. The distributions generated by the Arc-LSTM-SMF could be enhanced by merging them with the uncertainty from the SMF method. Another relevant aspect for multimodal evaluation would be to also compute scene and social metrics for multiple samples. Predicting a very spread distribution could increase the chance of generating trajectory samples that could collide with an obstacle, or another pedestrian. While such experiments were not conducted, its results could further yield interesting insights on the multimodality of trajectory forecasting.

4.8 Final remarks

The experiments in this chapter, in particular from section 4.6, show that the proposed solution for pedestrian trajectory forecast can successfully consider scene-specific elements and social interactions to improve the quality of the predictions. The results have shown that incorporating SMF in the model reduces the number of collisions with the environment. Furthermore, the use of relative neighbour velocities with an arc-shaped pooling can help to increase social accuracy, by reducing the number of collisions between pedestrians. Some remarks can be done that go beyond the aforementioned results.

An important discussion that has not yet occurred is regarding the applications of the proposed solution, as well as its standalone parts. All developed models process only sequences of 2D trajectories, not requiring direct access to video frames or semantic maps of each scene. This means that the model is able to learn pedestrian motion from data sources like GPS coordinates [55]. For the case of autonomous vehicles, for instance, there are a plethora of sensors at the disposal of a trajectory prediction method [56]. On the one hand, using extra information will increase the complexity of such method. But not using such information will, in one way or another, decrease the overall quality of the predictions, which is not desirable in a safety-critical system such as this. As such, while it is theoretically possible to integrate the proposed solution in an autonomous vehicle, its performance would most likely be poor when compared with a method that would integrate all available sensorial information [13, 56].

The consideration of the scene with SMF has proven to improve scene compliance, but it comes with some limitations. SMF are scene-specific, *i.e.*, they learn the sparsity of pedestrian motion for each scene, and therefore there should be a different model for each of the scenes. The Arc-LSTM-SMF was also trained on each of the available scenes. As discussed on section 3.4.3, it is not strictly mandatory, but helps the model to better incorporate the SMF predictions. Having a model specific to each available scene creates a real issue regarding scalability. There would be no issue deploying on a handful of scenes, *e.g.*, for a surveillance scenario with a fixed number of static cameras/sensors in predetermined places. However, problems would arise when the number of scenes can be immeasurable. In the example of an autonomous vehicle, it would not be appropriate to have a scene-specific model, since it is a mobile agent that can quickly go from one static environment to the next. Another issue with the scene aspect of the current model, that was already addressed in the SMF work [6], is its inability to handle change in the environment. In the data used, the static environment is the same for training and testing. However, that statement is not true in general. The environment can and most likely will suffer changes throughout time. Some examples of that are creation of new obstacles (*e.g.*, construction works), parked cars that were stationary at train time but are no longer there are test time. To combat this issue, the SMF, and the LSTM networks surrounding them would need to support some form of online learning, *i.e.*, observing new trajectories as they arise in time. This can allow the model to further consolidate the predictions, while also learning changes in the environment.

The social performance of the proposed solution has already been discussed in previous sections. While it has been shown that integrating the relative motion of neighbours can help reduce collisions between pedestrians, there are still many situations where the network fails to adopt social conventions. This is not suitable for applications with direct contact with humans. For instance, if a service robot would employ a trajectory prediction method that generated collisions between pedestrians, the chances are that the robot's own trajectory could collide with pedestrians, which is obviously undesirable. Another issue attached to this is model interpretability. This data-driven approach, on its own, can have issues when it comes to giving explanations to the decisions taken and the predictions given. Without a clear way to understand the reasoning behind failed predictions, such models should not be employed in safety-critical contexts [28].

It is worth stating that some of the limitations discussed in this section serve as possible areas of future research. Such possibilities for future work will be detailed in the following chapter.

5 Conclusions and future work

Contents

5.1 Conclusions

Pedestrian motion prediction is a challenging task, due to its many influencing factors. Two kinds of influencing clues have been highlighted in recent years: the environment of each scene - presence of obstacles and unwalkable areas -, and social interactions between pedestrians. The main goal of this thesis was to create a new data-driven model that would consider both these kinds of cues to improve trajectory forecasting, while only using as input the past trajectories of each pedestrian.

The thesis objective was fulfilled with the scene and interaction-aware Arc-LSTM-SMF model, which was built in stages. First, the consideration of scene constraints was done by combining LSTM networks with the SMF [6, 61] method, joining and weighing both predictions to improve the overall scene compliance. Second, the existence of social interactions FOV pooling layer that includes the relative velocities of each neighbour in front of the pedestrian.

The Arc-LSTM-SMF was shown to be better than its standalone parts (plain LSTM and SMF, LSTM with SMF, and LSTM with interactions). Not only were the collisions with the static environment reduced, but also there were less collisions between pedestrians. The proposed model was even able to outperform some state-of-the-art methods.

Nevertheless, the proposed solution has some limitations. For instance, it is still unable to learn several social aspects. On the other hand, being a scene-specific model, the type of applications can be more restricted. In the following section, new research directions will be proposed to combat some of the limitations.

5.2 Future work

Trajectory prediction is still a very open problem, and there are many possible lines of research that can be followed. This section contains some proposals to extend the work done in this thesis:

- **Improve social accuracy:** The failure of the proposed model to learn certain common social interactions may be partially attributed to the fact that this model is not directly trained to generate socially accurate predictions. An option could be to train the model not only using the loss function of (eq. (3.11)), but also a loss with aim to minimize the number of collisions between pedestrians, as defined in (eq. (4.3)).

- **Handle changes in the scene's environment:** This is a line of research already mentioned in [6]. The scene's environment is not fully static, and can suffer changes over time. As such, it is important to modify the SMF, as well as its incorporation with LSTM networks, to be able to adapt to change. This may involve, for instance, changing already trained models via online training, learning motion and subsequent changes in the environment by processing unseen trajectories throughout a certain time span.

- **Make the model more interpretable:** As mentioned on chapter 2, a recent line of work [26, 28] emerged to facilitate the understanding of trajectory predictions given by ANNs, which on their own can be hard to interpret. This is especially important in safety-critical applications, such as autonomous vehicles or social robots. Non-technical users, that do not possess expert knowledge of trajectory forecasting models, can understand the reasoning that led to such predictions. Some possibilities to achieve this can be creating a separate architecture that attempts to give explanations to the predictions [26], or to extend or modify the current model with a method that adds some form of interpretability [28].

- **Trajectory prediction of other agents:** It would be interesting to apply this model to predicting the trajectories of other types of agents, such as cyclists or vehicles. New datasets can be used, such as SDD [38] - where scene can play a more relevant role -, or nuScenes [56]. The actual model may require some changes, since the concept of social interactions does not apply, or at least not in the same fashion.

Bibliography

[1] W.-C. Ma, D.-A. Huang, N. Lee, and K. Kitani, “Forecasting interactive dynamics of pedestrians with fictitious play,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4636–4644.

[2] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, “Car-net: Clairvoyant attentive recurrent network,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 151–167.

[3] R. Mehran, A. Oyama, and M. Shah, “Abnormal crowd behavior detection using social force model,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2009, pp. 935–942.

[4] F. Poiesi and A. Cavallaro, “Predicting and recognizing human interactions in public spaces,” in *Journal of Real-Time Image Processing (JRTIP)*, vol. 10, no. 4, 2015, pp. 785–803.

[5] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, “Human motion trajectory prediction: a survey,” in *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 8, Jun 2020, p. 895–935.

[6] C. Barata, J. Nascimento, J. Lemos, and J. Marques, “Sparse motion fields for trajectory prediction,” in *Pattern Recognition*, vol. 110, 2021.

[7] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics,” in *Physical Review E*, vol. 51, no. 5, 1995, pp. 4282–4286.

[8] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2255–2264.

[9] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1349–1358.

[10] D. Ridel, N. Deo, D. Wolf, and M. Trivedi, “Scene compliant trajectory forecast with agent-centric spatio-temporal grids,” in *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, 2020, pp. 2816–2823.

[11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human trajectory prediction in crowded spaces,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[12] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, “Multi-agent tensor fusion for contextual trajectory prediction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 12 118–12 126.

[13] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control,” in *European Conference on Computer Vision (ECCV)*, 2020, pp. 683–700.

[14] J. Amirian, J. Hayet, and J. Pettré, “Social ways: Learning multi-modal distributions of pedestrian trajectories with gans,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2964–2972.

[15] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, “Context-aware trajectory prediction,” in *24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1941–1946.

[16] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, “Peeking into the future: Predicting future person activities and locations in videos,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5725–5734.

[17] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, F. Galasso, and M. Cristani, “Mx-LSTM: Mixing tracklets and vislets to jointly forecast trajectories and head poses,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6067–6076.

[18] C. Blaiotta, “Learning generative socially aware models of pedestrian motion,” in *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, 07 2019, pp. 3433–3440.

[19] J. Kooij, F. Flohr, E. Pool, and D. Gavrila, “Context-based path prediction for targets with switching dynamics,” in *International Journal of Computer Vision (IJCV)*, vol. 127, no. 3, 2019, pp. 239–262.

[20] A. Lerner, Y. Chrysanthou, and D. Lischinski, “Crowds by example,” in *Computer Graphics Forum (CVF)*, vol. 26, 09 2007, pp. 655–664.

[21] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 261–268.

[22] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, “Who are you with and where are you going?” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1345–1352.

[23] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” in *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780.

[24] F. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” in *Neural Computation*, vol. 12, 10 2000, pp. 2451–71.

[25] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, “SR-LSTM: State Refinement for LSTM towards pedestrian trajectory prediction,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 077–12 086.

[26] P. Kothari, S. Kreiss, and A. Alahi, “Human trajectory forecasting in crowds: A deep learning perspective,” in *IEEE Transactions on Intelligent Transportation Systems (ITS)*, 2021.

[27] Y. Liu, Q. Yan, and A. Alahi, “Social NCE: Contrastive learning of socially-aware motion representations,” in *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/pdf/2012.11717.pdf

[28] P. Kothari, B. Sifringer, and A. Alahi, “Interpretable social anchors for human trajectory forecasting in crowds,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 556–15 566.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 2014, pp. 2672–2680.

[30] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 11 2014. [Online]. Available: https://arxiv.org/abs/1411.1784.pdf

[31] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 29, 2016, pp. 2172–2180.

[32] L. Thiede and P. Brahma, “Analyzing the variety loss in the context of probabilistic trajectory prediction,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9953–9962.

[33] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, “Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for human trajectory prediction,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 412–14 420.

[34] Y. Li, Z. He, X. Ye, Z. He, and K. Han, "Spatial Temporal Graph Convolutional Networks for skeleton-based dynamic hand gesture recognition," in *EURASIP Journal on Image and Video Processing*, vol. 13, 2019, p. 78.

[35] B. Sifringer, V. Lurkin, and A. Alahi, "Enhancing discrete choice models with representation learning," in *Transportation Research Part B: Methodological*, vol. 140, 2020, pp. 236–261.

[36] G. Aoude, J. Joseph, N. Roy, and J. How, "Mobile agent trajectory prediction using bayesian non-parametric reachability trees," in *AIAA Infotech at Aerospace Conference and Exhibit*, 03 2011.

[37] L. Ballan, F. Castaldo, A. Alahi, F. Palmieri, and S. Savarese, "Knowledge transfer for scene-specific motion prediction," in *Lecture Notes in Computer Science (LNCS)*, vol. 9905, 10 2016, pp. 697–713.

[38] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 549–565.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 09 2014.

[40] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 4, 2017, pp. 640–651.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[42] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science (LNCS)*, vol. 9351, 10 2015, pp. 234–241.

[43] 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," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 137–146.

[44] P. Zechel, R. Streiter, K. Bogenberger, and U. Göhner, "Pedestrian occupancy prediction for autonomous vehicles," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 230–235.

[45] Y. Luo and P. Cai, "GAMMA: A general agent motion prediction model for autonomous driving," in *arXiv preprint*, 06 2019. [Online]. Available: https://arxiv.org/pdf/1906.01566.pdf

[46] H. Xue, D. Q. Huynh, and M. Reynolds, “SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction,” in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1186–1194.

[47] A. Syed and B. T. Morris, “SSeg-LSTM: Semantic scene segmentation for trajectory prediction,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2504–2509.

[48] T. Heiden, N. Nagaraja, C. Weiss, and E. Gavves, “SafeCritic: Collision-aware trajectory prediction,” in *arXiv preprint*, 10 2019. [Online]. Available: https://arxiv.org/pdf/1910.06673.pdf

[49] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, “Toward multimodal image-to-image translation,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 465–476.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations (ICLR)*, 2018.

[51] B. Ivanovic and M. Pavone, “The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2375–2384.

[52] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. K. Wong, and W.-c. WOO, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 28, 06 2015, pp. 802–810.

[53] Y. Li, “A deep spatiotemporal perspective for understanding crowd behavior,” in *IEEE Transactions on Multimedia*, vol. 20, no. 12, 2018, pp. 3289–3297.

[54] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2007, vol. 382.

[55] M. Karimzadeh, F. Gerber, Z. Zhao, and T. Braun, “Pedestrians trajectory prediction in urban environments,” in *International Conference on Networked Systems (NetSys)*, 2019, pp. 1–8.

[56] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuScenes: A multimodal dataset for autonomous driving,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.

[57] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, “TrajNet: Towards a benchmark for human trajectory prediction,” in *arXiv preprint*, 2018.

[58] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret, "Wildtrack: A multi-camera HD dataset for dense unscripted pedestrian detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5030–5039.

[59] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett, "3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5942–5948.

[60] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2211–2218.

[61] J. C. Nascimento, M. A. T. Figueiredo, and J. S. Marques, "Activity recognition using a mixture of vector fields," in *IEEE Transactions on Image Processing*, vol. 22, no. 5, 2013, pp. 1712–1725.

[62] A. Graves, "Generating sequences with recurrent neural networks," in *arXiv preprint*, 2013. [Online]. Available: https://arxiv.org/pdf/1308.0850.pdf

[63] P. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," in *Neural Networks*, vol. 1, no. 4, 1988, pp. 339–356.

[64] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory based Recurrent Neural Network architectures for large vocabulary speech recognition," in *arXiv preprint*, 2014. [Online]. Available: https://arxiv.org/pdf/1402.1128.pdf

[65] B. Yang, G. Yan, P. Wang, C. yao Chan, X. Liu, and Y. Chen, "TPPO: A novel trajectory predictor with pseudo oracle," in *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/pdf/2002.01852.pdf

[66] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *IEEE international Conference on Robotics and Automation (ICRA)*, 2018, pp. 4601–4607.

[67] Z. Zhou, "A brief introduction to weakly supervised learning," in *National Science Review*, vol. 5, 2018, pp. 44–53.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

[69] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020, pp. 1696–1703.

[70] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.