DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Pedestrian trajectory prediction with Convolutional Neural Networks

SIMONE ZAMBONI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Pedestrian trajectory prediction with Convolutional Neural Networks

SIMONE ZAMBONI

Master in Autonomous Systems
Date: June 17, 2020
Supervisor: Zekarias Tilahun Kefato
Examiner: Šarūnas Girdzijauskas
School of Electrical Engineering and Computer Science (EECS)
Host company: Scania CV AB
Swedish title: Förutsägelse för fotgängare bana med Convolutional Neural Networks

Abstract

Modelling the behaviour of pedestrians is essential in autonomous driving because consequences for misjudging the intentions of a pedestrian can be severe when dealing with vehicles. Therefore, for an autonomous vehicle to plan a safe and collision-free path, it is necessary not only to know the current position of nearby pedestrians but also their future trajectory.
In literature, methods to approach the problem of pedestrian trajectory prediction have evolved, transitioning from physics-based models to data-driven models based on recurrent neural networks.
This thesis proposes a new approach to pedestrian trajectory prediction, with the introduction of a convolutional model. This new model is able to outperform recurrent models, and it achieves state-of-the-art results on the ETH-UCY dataset and on the TrajNet dataset. Moreover, this thesis presents an effective system to represent pedestrian positions and powerful data augmentation techniques, such as the addition of noise and the use of random rotations, which can be applied to any model. Finally, a study on the effectiveness of various techniques to include social information is presented, which demonstrates that simpler approaches fail to capture complex social interaction.

Sammanfattning

Att modellera fotgängares beteende är en grundläggande byggsten inom förarlös körning eftersom att konsekvenserna av att misstolka en fotgängares intention kan leda till allvarliga konsekvenser när det finns bilar involverade. Därför, för ett förarlöst fordon ska kunna planera en kollisionsfri och effektiv trajektoria, behövs inte bara positionen på de omgivande fotgängarna utan även deras framtida rörelsemönster.
I litteraturen har metoder anpassade för att prediktera fotgängare evolverat, med en övergång från fysikbaserade modeller till data-baserade modeller som baseras på repetitiva och återkommande neurala nätverk.
Det här examensarbetet föreslår en ny approach för att prediktera fotgängares trajektorior, som introducerar en faltad neuralt nätverks modell. Den här nya modellen är kapabel att prestera bättre än repetitiva och återkommande neurala nätverks modeller, och åstadkommer toppresultat baserat på dataseteten ETH-UCY och TrajNet. Därutöver, presenterar det här examensarbetet ett effektivt system för att representera fotgängares positioner och kraftfulla data-augmenteringstekniker, som till exempel metoder för metoder för att lägga till brus och även att randomiserade rotationer, som kan tillämpas på alla modeller. Slutligen, så har en studie presenterats för att studera effekten av att inkludera social information, det vill säga de relativa tillstånden mellan fotgängare, vilket redovisar att mindre avancerade modeller inte klarar av att fånga komplex social interaktion.

Acknowledgements

I would like to thank my supervisors at SCANIA, Norén Christoffer and Laura Dal Col. Their guidance and assistance have been of primary importance in all the aspects of this work, but especially in the definition of the tasks and in the validation of the results with new data.
I would also like to thank my supervisor, Zekarias Tilahun Kefato. His technical expertise and insights were essential in pushing the developed models at their limit, and his writing skills crucial in the creation of the final report. I also thank my examiner Šarūnas Girdzijauskas, for giving me extensive feedback and guidance.
I thank everyone in the EARM team at SCANIA, including all the other master students in the team, for all the wonderful meetings and the time together at the office.
I also thank the KTH administration for supporting international students like me with useful guidelines and organizing career fair and other events that help students finding master thesis projects.
I finally thank everyone who supported me in this journey that lasted two years and crossed multiple countries, in particular, my family, Anna and the other EIT students.

Contents

Chapter 1

Introduction

Pedestrians are one of the most important actors in modern transportation, a field which is being quickly transformed by new technologies. Consequently, these new technologies have to take into consideration pedestrian safety in their design if they want to be successful. This is especially true in the case of autonomous driving, which has the potential to drastically improve pedestrian safety.
Currently, road safety for pedestrians is far from ideal: in 2017, around 25 thousands road fatalities were reported by the 28 EU Member States [1]. More than 20% of these fatalities were pedestrians, and if only urban roads are taken into consideration, this percentage grows to 40%.
Therefore, to improve road safety pedestrians security needs to be a top priority in the design and development of self-driving vehicles. To achieve this goal a key component is the detection and tracking system, that detects pedestrians and tracks them while they move. However, detection and tracking are not sufficient on their own, because pedestrians are moving entities that can quickly change their direction. Therefore, computing the future trajectory of pedestrians is a necessity in self-driving vehicles path planning. This problem can be also found in other areas, such as social robotics and crowd surveillance, and it is called pedestrian trajectory prediction.

1.1 Problem Formulation

The goal of pedestrian trajectory prediction is to predict the future positions of a pedestrian given its previous positions. Concretely, given a scene where pedestrians are present, their coordinates are observed for a certain amount of time, called T_{obs}, and the task is to predict the future coordinates of each

pedestrian from T_{obs} to T_{pred}.
The position of each pedestrian is characterized by its (x, y) coordinates with respect to a fixed point, therefore for pedestrian i the positions (x_t^i, y_t^i) for t $\in 0, ..., T_{obs-1}$ are observed and positions $(\hat{x}_t^i, \hat{y}_t^i)$ for t $\in T_{obs}, ..., T_{pred-1}$ are predicted. We denote all the past positions of a pedestrian i with X^i, the predicted future positions with $\hat{Y}^i$ and the real future positions of pedestrian i with Y^i. Moreover, a discretization of time is assumed, in which the difference between time t and time $t+1$ is the same as the difference between time $t+1$ and time $t+2$.
A graphical visualization of the pedestrian trajectory prediction task can be found in Figure 1.1. In essence, the problem to be solved can be formulated as:

How to predict the future positions of pedestrians from their past trajectory with the lowest possible error?

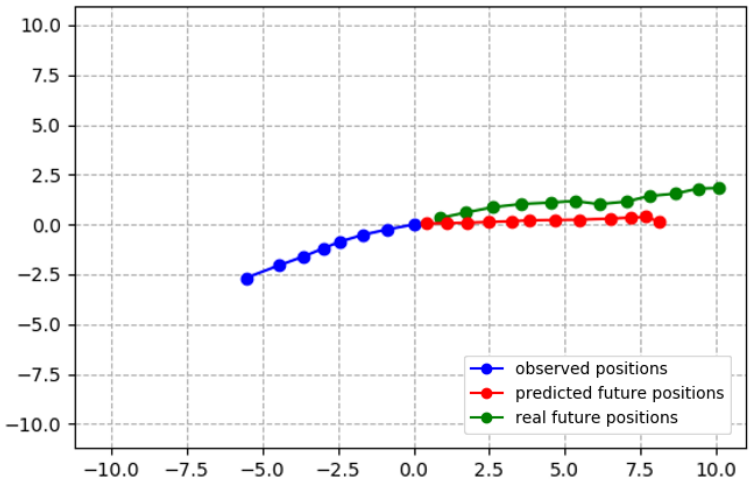


Figure 1.1: Visualization of a pedestrian trajectory prediction. The scale is in meters, the observed positions are the blue dots on the left, the predicted future positions are the red dots on the right and the real future positions are the green dots on the right.

1.2 Applications

Pedestrian trajectory prediction is a topic that is gaining increasing attention because its applications are becoming more and more relevant with time. Mod-

elling how people move in their environment and in the presence of other people can benefit mainly three areas:

- Crowd surveillance: systems able to predict pedestrian behaviour are also capable of detecting anomalies and unexpected behaviours.
- Social and mobile robots: robots that operate indoor and outdoor in environments with humans have to predict their movements to avoid accidentally hitting them.
- Autonomous driving: self-driving vehicles have to understand the intentions of pedestrians and predict their future trajectories to safely operate, especially in urban scenarios. This is a critical aspect in autonomous driving because consequences for misjudging the intentions of a pedestrian can be severe. Therefore, a system implementing pedestrian trajectory prediction will surely be one of the components in the autonomous driving software stack.

All of these applications are important and the findings of this thesis can be applied to all of them. However, special attention is put on the autonomous driving area since it is the most interesting one for the host company.
Because the aforementioned applications have become more important and demanding over time, methods to approach the problem of pedestrian trajectory prediction have evolved, transitioning from physics-based models to data-driven models that use deep learning.

1.3 Background

One of the first approaches in pedestrian behaviour modelling was the Social Forces Model [2]. It considers every pedestrian as a particle subject to forces from nearby people and obstacles, and the sum of these forces give the next pedestrian position. Physics-based models like this have been extensively developed in the past, with the introduction of other techniques such as BRVO [3]. However, in recent times the data-driven approach to pedestrian behaviour modelling has become increasingly popular, thanks to its promising results.
With the extensive development of deep learning models in the last few years, the first work to achieve state-of-the-art results using neural networks was published in 2016 and it is called Social LSTM [4]. The Social LSTM model outperforms physics-based approaches because it takes as input not only the past trajectory but also a representation of social information for each pedestrian,

and therefore it is able to learn complex social interactions.
Since 2016 a number of deep learning architectures have been proposed to improve the state-of-the-art. Common elements in these recent works are the use of Generative Adversarial Networks [5], the use of graph neural networks [6], the integration of attention [7] and the inclusion of spatial [8] and image information [9].

1.4 Purpose

The literature on pedestrian trajectory prediction is extensive, however, there are still some topics that can be investigated further. The first one of these topics is data augmentation. There are numerous datasets for trajectory prediction, as described in Section 3.1, nonetheless, the total quantity of data is still limited. On the other hand, it is widely understood that neural networks perform better with a vast amount of data. Thus, to address the issue of limited data, a solution could be to use data augmentation, but this approach is often not mentioned in publications. Consequently, it is currently unknown what data augmentation techniques can be applied to pedestrian trajectory prediction and which of them are most effective.
Another topic hardly explored in literature, [10] being the only exception, is the usage of convolutional models in pedestrian trajectory prediction. In the machine translation field it was proved, in works such as [11] and [12], that convolutional neural networks are a valid alternative to recurrent neural networks. However, in pedestrian trajectory prediction a detailed confrontation is still missing. Moreover, there is no publication presenting a convolutional model that also utilizes social information and how this model compares with recurrent networks that use the same approach to represent social information.
Based on these gaps present the literature, the following research question for this thesis has been formulated:

Do effective data augmentation techniques for pedestrian trajectory prediction exist? Is a convolutional model preferable to recurrent models in pedestrian trajectory prediction, even when social information is taken into consideration?

The purpose of this thesis is to answer this research question providing evidence with suitable experiments.

1.5 Methodology

The research question is answered by collecting and analyzing empirical evidence. The empirical evidence is obtained from experiments evaluating, on the same metrics and on the same data, various models and techniques. To correctly analyze the results from different experiments the metrics used must be adequate and must be the same on all the experiments. The metrics chosen are the Average Displacement Error (ADE) and the Final Displacement Error (FDE) explained in detail in Section 3.2. Another important aspect of the experiments is that they must be performed on the same data. The chosen dataset are the ETH [13], UCY[14] and TrajNet [15], datasets which are described in detail in section 3.1.
To evaluate different data augmentation techniques first some baselines networks needs to be implemented. These baselines are recurrent neural networks so that later they can also be compared to the convolutional model. To then assess the effectiveness of the various data augmentation techniques the results obtained utilizing the same baseline with different data augmentation techniques are compared.
Once it is clear which techniques it is better to use it is possible to confront the results obtained by the recurrent baselines with the ones obtained by the convolutional model. This comparison can prove if a convolutional model can perform better than recurrent models. Then both the baselines and the convolutional model can be modified to include social information to prove if results previously obtained also hold with the introduction of social information. In this way, it is also possible to evaluate the effectiveness of various methods to model social information on multiple architectures.

1.6 Contributions

The work presented in this thesis follows the methodology previously described and answers the research question with the following contributions:

- The identification of effective data augmentation techniques such as random rotations and the addition of Gaussian noise;
- The introduction of a novel model based on 2D convolutions capable of achieving state-of-the-art performances on the ETH-UCY and Trajnet datasets;

- A study on the effectiveness of various techniques to include social information, which demonstrates that complex models to represent social interaction are needed.

Researchers can benefit in multiple ways from the contributions of this thesis. The first way is to use the data-augmentation and pre-processing techniques presented to improve the results of their models. The second way is to use the results obtained by the LSTM-based and convolutional-based model with effective data-augmentation and pre-processing techniques as a baseline when assessing the performance of new models. The third way could be to use the convolutional model architecture as a starting point to create more complex models that utilize social information effectively.

1.7 Limitations

In a real application setting, data is collected from the surrounding environment and is then processed to obtain the coordinates of each pedestrian. Therefore, a system implementing pedestrian trajectory prediction receives the positions of each pedestrian from the detection and tracking systems. This means that the accuracy of the detection and tracking system heavily influences the quality of the results obtained from the trajectory prediction system.
Moreover, current approaches in pedestrian trajectory prediction do not take into account uncertainty on the positions of pedestrians. This, in turn, increases even more the reliance on detection and tracking systems. Consequently, it is possible to say that the prediction system is at best as good as the detection and tracking system.
Another limitation when pedestrian trajectory prediction is applied to a real-world scenario is the observation time. If a vehicle is going very fast very few observation can be made for each pedestrian, and this can have a significant effect on the accuracy of the predictions.

1.8 Ethics and sustainability

This work aims to improve the understanding of the pedestrian trajectory prediction topic to improve predictions, however, no system is perfect. If a failure in the prediction system will cause an accident, someone or something should take responsibility for what it has happened. Should the responsibility be placed on the driver, the company selling the vehicle, the software itself or the engineers who developed it? The fact that the prediction system might

be based on deep neural networks might complicate the situation even more, since understanding the reasons behind the output of deep learning models is notoriously difficult. The aim of this thesis is not to answer these questions but it is still important to bring them up so that a discussion can be created around them.

1.9 Outline

The document is structured in the following way: immediately after the current introductory chapter is the related work chapter, and then the metrics and data used are explained in the methodology chapter. The baselines models, the pre-processing techniques, the data augmentation, the convolutional model, the ways to include social information and the final results are presented in Section 4. Then there is the discussion chapter in which results are commented and analyzed more in detail and after that, the conclusion chapter summarizes the most important takeaways.

Chapter 2

Related Work

Methods to approach the problem of pedestrian trajectory prediction have evolved over time, transitioning from physics-based models to data-driven models that use deep learning. In this chapter, a brief overview of physics-based models is first given, to then analyze in more detail data-driven models that use deep neural networks.

2.1 Physics-based approaches

A possible approach to predict the future positions of pedestrians is to formalize rules and formulas that dictate the motion of people. These equations are hand-crafted and reflect in a numerical way our theories on how pedestrian walk and what they take into consideration when deciding where to move next. In the following subsections, the most commonly used techniques using this approach are described.

2.1.1 Constant Velocity Model

The simplest trajectory prediction technique that can be used is the constant velocity model [16]. This method uses the position of pedestrian i at time $t-1$, (x^i_{t-1}, y^i_{t-1}) , and at time t, (x^i_t, y^i_t), and it computes the instantaneous velocity v, with $v = (x^i_t - x^i_{t-1}, y^i_t - y^i_{t-1})$. It then assumes that the pedestrian will continue walking with the same velocity and direction indefinitely.
This method presents a lot of drawbacks: it is sensitive to noise, it does not take into account other pedestrians or the environment and most importantly it can't predict curving trajectories.
It is also possible to use the constant acceleration method instead of constant

velocity, reducing the impact of noise. However, the constant acceleration is still a simple model that cannot grasp complex behaviours such as turning to avoid obstacles and other people, that are common in human motion.

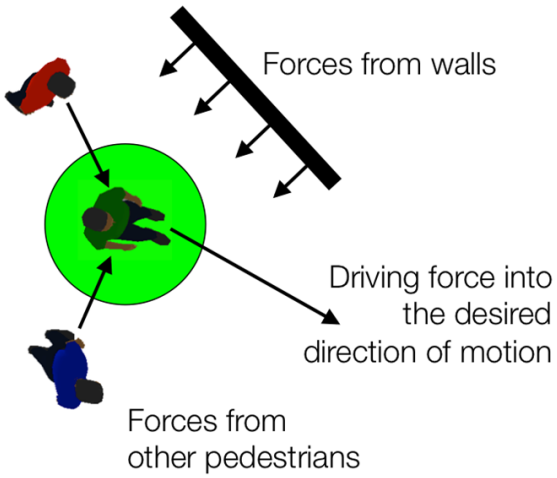


Figure 2.1: Visualization of the Social Forces model. Image taken from [17].

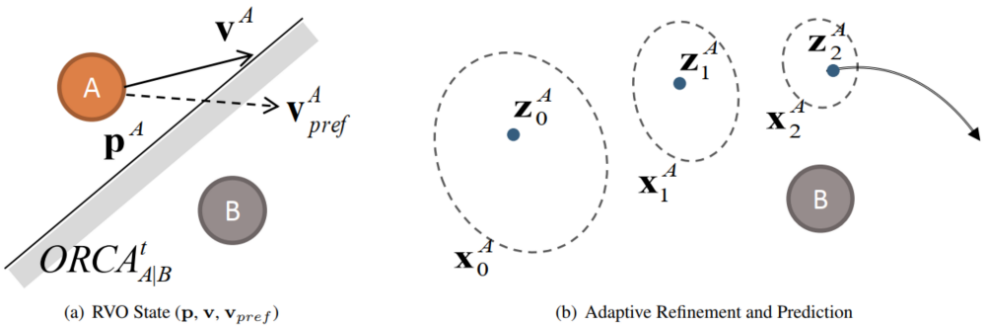
2.1.2 Social Forces

One of the first works in pedestrian trajectory prediction that remains very influential still to this day is the work done by Helbing and Molnar about Social Forces [2] in 1995. The proposed method considers each pedestrian as a particle and assumes that the motion of these particles can be described with three forces representing:

- The acceleration towards the preferred velocity for each pedestrian;
- The sum of all the repulsion forces: these forces simulate the desire to keep a certain distance from other people and obstacles;
- The sum of all the attractive forces: these forces simulate the desire of pedestrians to stay close to other people (friends and couples walk closely together).

A visual representation of these forces can be seen in Figure 2.1. The final acceleration of each pedestrian will be the sum of all the forces affecting that pedestrian.

The social forces model takes a physics-based approach to trajectory prediction, it has a clear formalization and it represents one of the most historically influential works. However, it needs precise information on the obstacles and positions of other pedestrians, therefore can be applied only in scenarios where this data is available.



(a) RVO State ($\mathbf{p}$, $\mathbf{v}$, $\mathbf{v}_{pref}$) (b) Adaptive Refinement and Prediction

Figure 2.2: Visualization of the BRVO method in action. (a) RVO simulation. (b) Agent state estimation. Image taken from [3]

2.1.3 BRVO

Proposed in 2015 for robotic navigation, BRVO [3] builds upon Reciprocal Velocity Obstacle (RVO) [18] and the Optimal Reciprocal Collision Avoidance (ORCA) [19].
As can be seen in Figure 2.2, each pedestrian is represented by its position, velocity and preferred velocity. If a pedestrian is not able to follow its preferred velocity because of collisions (computed by ORCA) the closest velocity to the preferred one is chosen. Some uncertainty is assumed in the position and velocity of each pedestrian and therefore an Ensemble Kalman Filter is used together with an Expectation-Maximization algorithm to have the best approximated RVO for every agent, together with the uncertainty in the model. In fact, the name of the model (BRVO) means Bayesian RVO because it computes also the uncertainty of the model.

2.2 Data-driven approaches

Until 2016 the most common approach for pedestrian trajectory prediction was the physics-based one, but it is also possible to predict future trajectories in a data-driven manner.

The use of a data-driven approach implies learning how people walk by training a machine learning model with real pedestrian trajectories. This approach has the advantage of being able to learn the complex patterns of walking and collision avoidance that are too difficult to formalize in mathematical terms. This is the reason why in the last years data-driven approaches based on deep learning have outperformed physics-based approaches like Social Forces. This proves that data-driven approaches can directly extrapolate from data the rules and subtle details of human walking behaviour that would be impossible to formalize.
Learning how people walk only from observed trajectories requires three main components: a machine learning model that has enough representation power, an effective optimization technique and enough real-world data. All these components where available before 2010 (datasets: UCY[14] in 2007 and ETH[13] in 2009, LSTM [20] in 1997, and backpropagation through time [21] in 1990), but the first data-driven approach using deep learning that achieved state-of-the-art results came in 2016 [4]. The model was called Social LSTM and it was proposed four years after the introduction of AlexNet [22], which sparked the interest of the scientific community in neural networks.
In the following sections, the key elements in sequence prediction using deep learning (LSTM and the Encoder-Decoder architecture) are first introduced, and then the various techniques that have been employed for pedestrian trajectory prediction are described.

2.2.1 LSTM Introduction

An LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN). An LSTM cell is composed by a cell state, a hidden state and three gates (forget, input and output gate). This type of RNN was first introduced in 1997 [20], and today is widely used in tasks that have as input a variable-length sequence [23].
An LSTM takes as input the next item in the sequence and the past hidden state, it then updates its internal cell state and outputs the next hidden state. The internal structure and main formulas of an LSTM can be found in Figure 2.3. We continue this section with a brief explanation of the key components in an LSTM.

Forget Gate. The forget gate decides what information should be kept or thrown away. It takes as input the next item in the sequence and the previous hidden state and it generates its output using a set of weights and a sigmoid

function (this means that each output value will be between 0 and 1).
Input Gate. The input gate decides what new information to store in the cell state. It has the same inputs as the forget gate and it uses two sets of weights to decide how important the new information is and what information should go into the next cell state.
Cell State update. The cell state is updated after the information is passed through the forget gate and input gate. This is done by summing the previous state multiplied by the forget gate and the outputs of the input gate multiplied together.
Output Gate. The output gate decides the cell output. It takes the same set of inputs as the forget gate, but also the new state. It uses a set of weights and a sigmoid to compute the next hidden state.

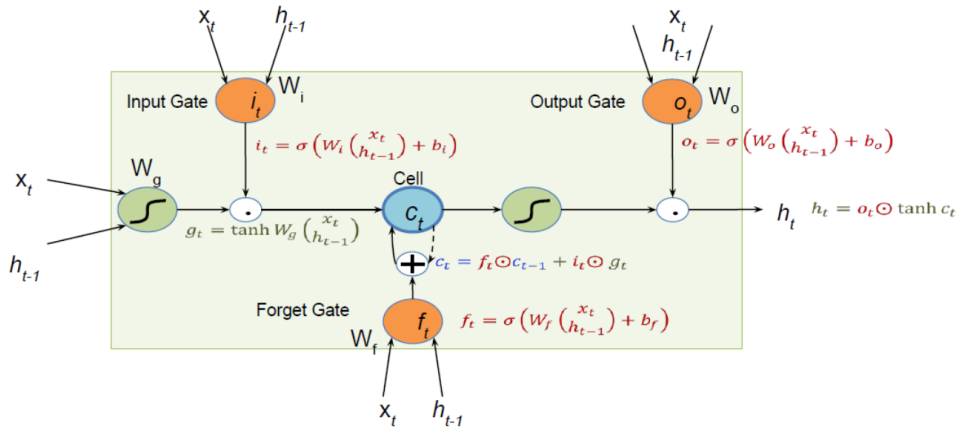


Figure 2.3: Visualization of an LSTM cell with the relevant formulas. The next item in the sequence is x_t, the last hidden state is h_{t-1} and the weights are denoted by W.

The vast majority of the data-driven methods for pedestrian trajectory prediction using deep learning employ one or more LSTMs. Consequently, having a clear understanding of how an LSTM cell works is essential in pedestrian trajectory prediction.

2.2.2 Encoder-Decoder RNN

The Encode-Decoder architecture [24] is a popular method that uses RNNs (especially LSTMs) in sequence to sequence tasks, such as pedestrian trajectory prediction.
In the Encoder-Decoder architecture, there are two RNN, one is the encoder

and another one the decoder. The encoder takes as input the whole variable-length sequence and it encodes it in its final fixed-length hidden state. This hidden state is then passed to the decoder that generates the desired number of hidden states and outputs. Therefore the Encoder-Decoder architecture transforms a variable-length input sequence into a fixed-length representation, and then it transforms a fixed-length representation into a variable-length target sequence. This can be visualized in Figure 2.4.

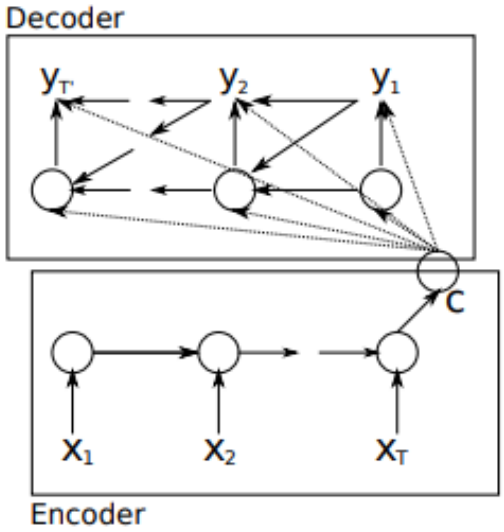


Figure 2.4: Encoder-Decoder architecture structure. All the information is compressed by the encoder in its final hidden state and is then passed to the decoder. Image taken from [24].

The encoder-decoder architecture has shown to be effective in the machine translation task [24], but it has the drawback that longer sequences can be difficult to compress it into one fixed-size vector. This issue is less relevant in trajectory prediction because usually T_{obs} is set to 8 and T_{pred} to 12 (as introduced by [4]). Therefore, the encoder-decoder architecture has been extensively used in the literature for pedestrian trajectory predictions, in works such as [25], [26], [27], [28], [5] and [9].

2.2.3 Social-LSTM

The Social LSTM model [4] was the first data-driven approach to achieve state-of-the-art performance on the pedestrian trajectory prediction task. It was proposed in 2016 by Alahi et al.

Analyzing their results is possible to conclude that a simple LSTM in the majority of cases does not outperform the social forces model. Therefore, the main contribution of Social LSTM is the introduction of social information as input to the LSTM. The social information is created placing a grid in the position of the pedestrian that is being taken into consideration, and filling each spot of the grid with the hidden state of the LSTM cell of each pedestrian in that cell of the grid. If two or more pedestrians are in the same place their hidden states are summed. This is shown visually in Figure 2.5.

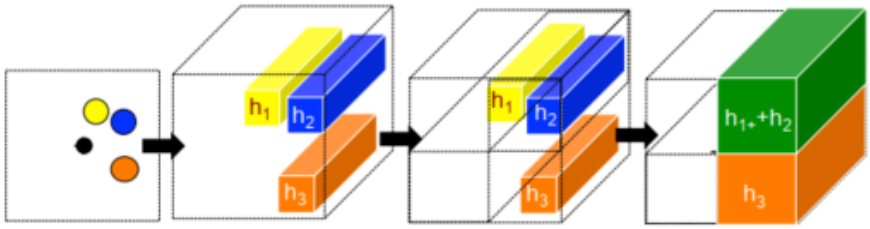


Figure 2.5: Construction of the social grid for the black pedestrian in the Social LSTM model. Image taken from [4].

The social LSTM method was of great importance in the field of pedestrian trajectory prediction because it demonstrated that deep learning techniques applied to pedestrian trajectory prediction can outperform physics-based method. Since 2016, numerous papers tackling the pedestrian trajectory prediction using deep neural networks have been published. These recent works introduce new architectures and techniques to improve the state-of-the-art in pedestrian trajectory prediction and are presented in the following sections. Therefore today, differently from before 2016, the data-driven approach using deep learning is the leading approach in pedestrian trajectory prediction.

2.2.4 The use of attention in pedestrian trajectory prediction

Attention was first introduced in the machine translation field[29] to create an alignment model to weight differently each word in a sentence when translating the next word. The same concept can be applied to trajectory prediction: not all the previous positions are equally important when predicting the next position.
The first work using attention in pedestrian trajectory prediction was introduced in 2017 by Fernando et al. [7]. Their method uses soft attention for

the pedestrian own past trajectory and hard-wired attention based on distance when considering the past positions of other pedestrians. The architecture proposed by Fernando et al. can be found in Figure 2.6.

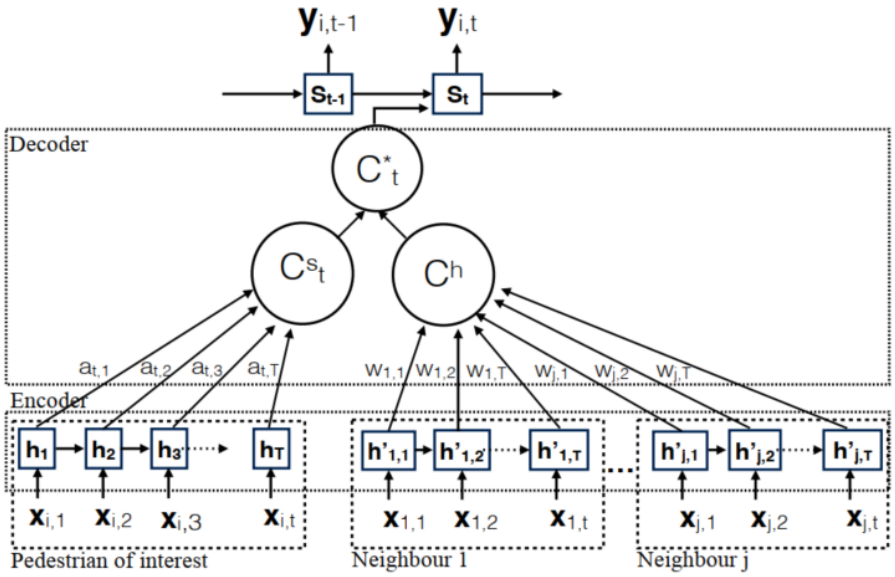


Figure 2.6: Architecture proposed by Fernando et al. [7], that concatenates two context vectors. One is created using the pedestrian own past cell states weighted using soft attention, and the other one is created with other pedestrians past cell states weighted by hard-wired attention based on distance. Image taken from [7].

From 2017 onward numerous publications have used attention in different ways to improve their results, like in [30], [9], [31], [32], [33] and [34].

2.2.5 GAN in pedestrian trajectory prediction

Generative Adversarial Networks (GAN) [35] are a way to generate new synthetic data similar to the training data. A generator network takes as input random noise and outputs a sample, while the discriminator network takes as input both real and generated samples and has to determine if the sample is fake or not. The objective of the generator is to fool the discriminator and the one of the discriminator is to correctly label the data it is fed with.
Generative Adversarial Networks are a way to address the multi-modal aspect of the trajectory prediction problem: there is no single correct future prediction given past positions. For example, the road could split in two and trajectories going in one or another direction are both possible.

The basic architecture proposed in 2014 by Goodfellow et al. [35] is not sufficient for trajectory prediction. It is the case because the objective in trajectory prediction is not to only generate trajectories similar to the real ones but to generate plausible future trajectories given the past positions. Therefore in trajectory prediction Conditional GAN (CGAN) [36] are used to condition the generation of future trajectories based on previous trajectories. The base architecture of a Conditional GAN can be found in Figure 2.7.

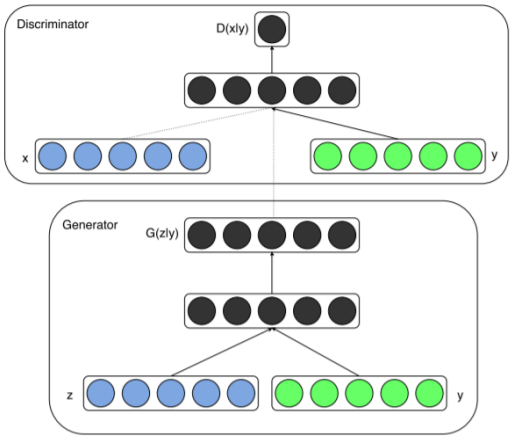


Figure 2.7: Structure of a Conditional GAN, y is the conditional term that is fed to both the generator and the discriminator. Noise is denoted with z and the samples are denoted with x. Image from [36].

A very influential work in the use of GAN applied to pedestrian trajectory prediction is Social GAN[5]. The Social GAN architecture uses an LSTM encoder, pooling for social interaction and an LSTM decoder in the generator, and an LSTM encoder for the discriminator. The full architecture is shown in Figure 2.8.
An important contribution in the Social GAN architecture is the addition of a variety loss on top of the standard discriminator loss. For every input trajectory, multiple future trajectories are generated and the variety loss computes how far these synthetic trajectories are from the real one. However, only the best generated trajectory, the one that produces the lowest error, is used when backpropagating the error. In this way, the generator is incentivized to create multiple possible trajectories. However, this also means that the generator will also create trajectories very far away from the ground truth.
The same approach is used in evaluating the model: for each input, the network outputs multiple trajectories, and only the one closer to the real one is

used to compute the error. This is important an aspect to keep into consideration when comparing results from non-generative models and generative models because results are computed in a very different way.

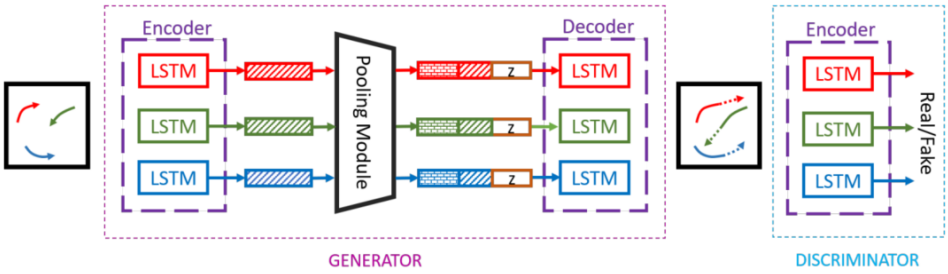


Figure 2.8: Architecture of the Social GAN network. Image taken from [5].

Attention can also be used in conjunction with GANs, in the encoding of the past trajectory that represents the conditional part [32], or in the social information construction [9].
Another interesting work that applies GAN to pedestrian trajectory prediction is [33], in which an Info-GAN [37], a type of GAN in which a latent code is fed to the generator and the discriminator has to reconstruct it, has been used in conjunction with a CGAN.

2.2.6 Including social information

The introduction of the Social LSTM [4] model proved that including social information represented as a grid as input to the network can improve the results of trajectory prediction. Since then, other methods have been proposed to address the problem of how to better represent social information.
Other types of layout that have been used to present social information are: a circular grid with the radius exponentially increasing [27], a circle divided in regions of different shapes [38] or an ellipse (o semicircle like [39]) to better represent how humans perceive their environments [40] [41]. Other modifications of the social information include the use of an angular pedestrian grid that indicates the distance from the closest pedestrian in that direction [8], or the hidden state of nearby pedestrians in every direction scaled based on distance [42]. An illustration of these approaches can be found in Figure 2.9.

It is also possible to implicitly model social information creating a displacement volume with the displacement vectors of each pedestrian and then

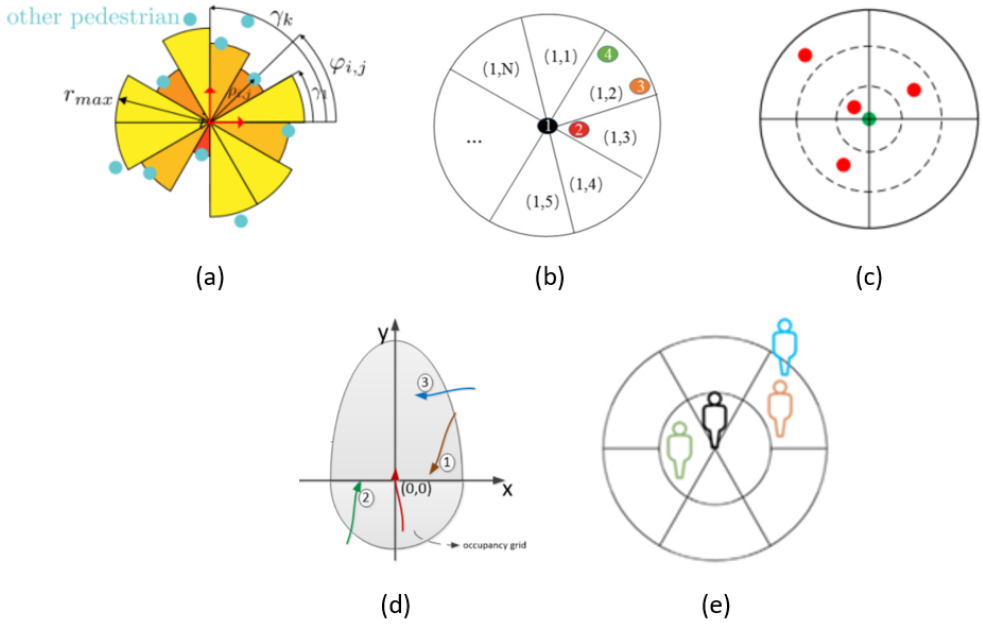


Figure 2.9: (a) Angular pedestrian grid that indicates the closest pedestrian in each direction, from [8]. (b) Circle of neighbours divided by direction, taken from [42]. (c) Circular shape neighbourhood, with three circles divided into 4 cells each, taken from [27]. (d) Ellipse-like neighbourhood, notice that the ellipse is bigger in front (where the pedestrian can see) and smaller behind (where the pedestrian cannot see), taken from [40]. (e) Circular neighbourhood divided into different regions, from [38].

applying convolutions to the entire displacement volume, like it is done in [43]. Another interesting approach is the one of [40], in which the position and velocity of each nearby pedestrian are first embedded and then fed to the neural network. In this way, the model knows the exact position of nearby pedestrians and not an approximation like when using occupancy grids. The architecture in detail can be viewed in Figure 2.10.

Not all the nearby pedestrians have the same importance when predicting the path of a person, therefore attention has been extensively used in the embedding of social information to improve result, in works like [7], [9], [31], [33] and [34]. Another way to differentiate between influential pedestrians or not is to cluster them based on their direction and consider only pedestrians in the same direction as the current pedestrian, as done in [44].
In the examples previously discussed each pedestrian has different social information. However, it is possible to instead use the same social information for

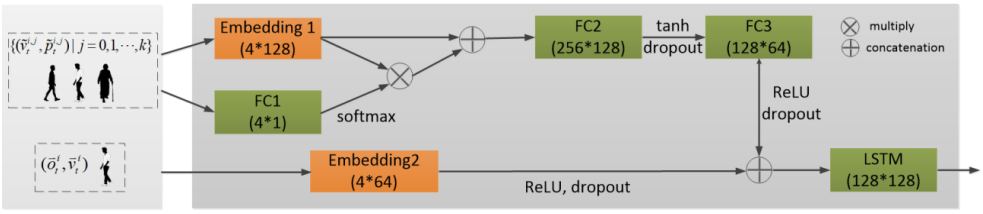


Figure 2.10: Embedding of nearby pedestrians (on top) and current pedestrian(on the bottom) in [40]. The nearby pedestrians are represented with relative position and velocity with respect to the current pedestrian. The current pedestrian is represented with its velocity and position. Image taken from [40].

every pedestrian: this is the approach of StarNet [45]. The StarNet architecture uses a unique network to embed the social information of all the scene and then every pedestrian (with its own LSTM) uses the same social information to predict the future trajectory. This method avoids to compute the social information for each pedestrian and therefore it has a smaller computation time. The architecture can be viewed in Figure 2.11.

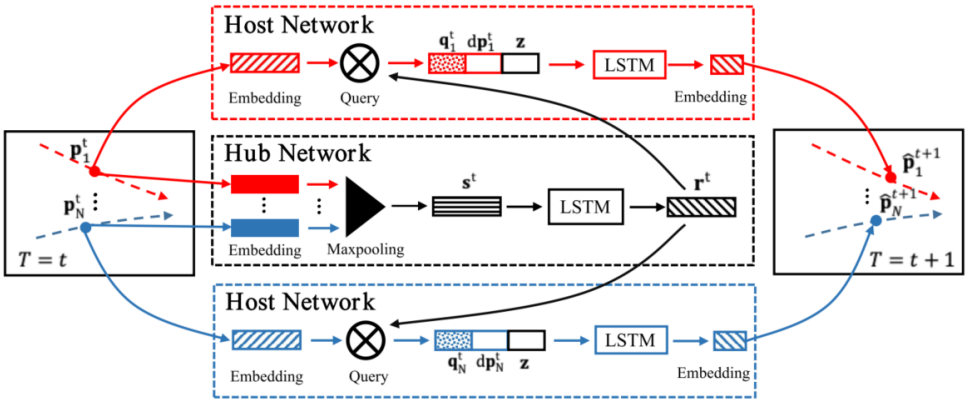


Figure 2.11: StarNet architecture. The Hub Network computes the social information of the entire scene and its output is used by every pedestrian, represented by the Host Network. Image taken from [45].

2.2.7 Including spatial information

On top of past positions and social information, it is also possible to include information about the surrounding of each pedestrian as input to a network. In this way, the model can learn behaviours like avoiding obstacles and therefore

generate better predictions.
One way to include social information is the one introduced by Bartoli et al. in [46]. First, points of interest in the environment (like obstacles, entrances or artworks) are annotated. Then, a grid similar to the one used in Social LSTM [4] is used to indicate the presence of points of interest in every cell. This spatial grid, the same social grid as Social LSTM and the past positions are then fed to an LSTM to predict the future trajectory.
A drawback in using points of interest is that they are not able to capture the dimensions of obstacles like cars and buildings. A method that does not have this downside is the use of an occupancy grid to represent spatial information, as done in [8]. The grid is centred on the pedestrian and indicates which cells are occupied by obstacles. Then a pre-trained encoder compresses the grid into a smaller representation that can be given as input to an LSTM. The complete architecture that uses this method can be found in Figure 2.12.

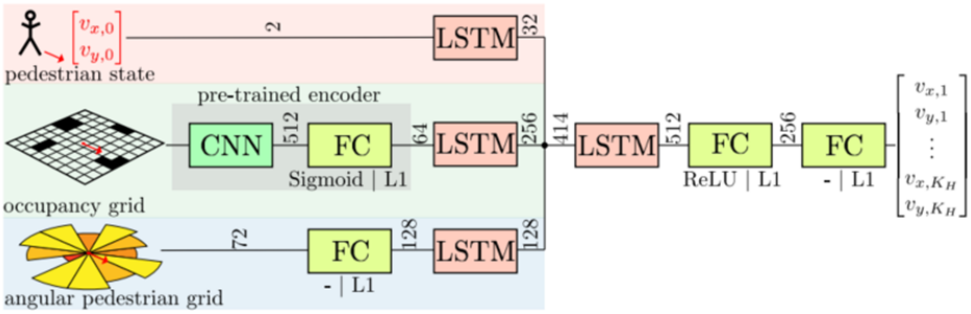


Figure 2.12: Architecture of [8]. Notice how every type of input has its own dedicated LSTM. The output of the three LSTM is then merged and fed to the main LSTM that predicts the future displacements all at once.

A different approach to include spatial information is to use a semantic segmentation of the scene. This semantic map is fed to convolutions and then is used as input to the pedestrian LSTM. Works like [47] and [48] use this approach. Classes like grass, building, sidewalk and obstacle are used for the semantic segmentation. A visualization of this method can be found in Figure 2.13.

Knowledge about the topology of the environment is crucial in the use of points of interest, occupancy grids or to perform semantic segmentation. In the datasets the authors of these works used there is no explicit representation of such information. Therefore the points of interest, the occupancy grids and the class of each pixel are annotated by hand on every scene. This is of

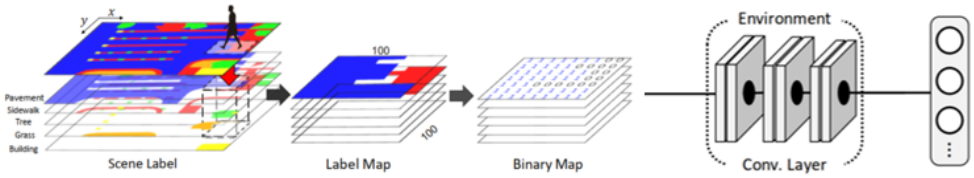


Figure 2.13: Segmentation of the scene transformed in to a binary map to be fed to a CNN (Convolutional Neural Network) to encode scene features. These feature can then be used as input to an LSTM. Illustration from [47].

course not a scalable approach and represents a limitation in the use of spatial information in real applications.

2.2.8 Including image information

The majority of the datasets used for pedestrian trajectory prediction are annotated videos taken from a bird's eye view, like in the case of [13], [14], [49] and [50]. For more information on these datasets see Section 3.1. Including information coming directly from the video image could permit to better understand the context around a person and improve prediction accuracy.
One of the most influential works using image information is SoPhie [9]. It uses a pre-trained VGG-19 [51] as a feature extractor on all the image (therefore the features extracted are the same for every pedestrian) combined with a module for physical attention. The physical attention takes as input the hidden state of the past trajectory encoder for each pedestrian and the image feature vector to generate an attention context vector. It also uses social information weighted on each pedestrian by the social attention module. All these vectors are then used as the condition for a CGAN generating future paths for each pedestrian. Thus these generated paths are conditioned on the past trajectory, social and physical information. The complete architecture can be found in Figure 2.14.

Another approach on the use of image information is the one used in SS-LSTM [27] and in [52]: only a portion of the whole image centred on the current pedestrian is used. Then convolutions are applied to encode the image (different for each pedestrian) and then the encoding is fed to an LSTM. Different LSTM encoders are also used for the social information(in the form of a circular occupancy map) and for the trajectory observed. The state of the three encoders is used as input for a single decoder that outputs the future trajectory. The SS-LSTM architecture is shown in Figure 2.15.

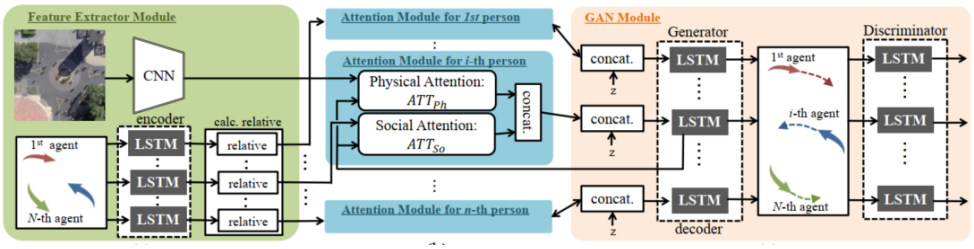


Figure 2.14: SoPhie architecture. Image taken from [9].

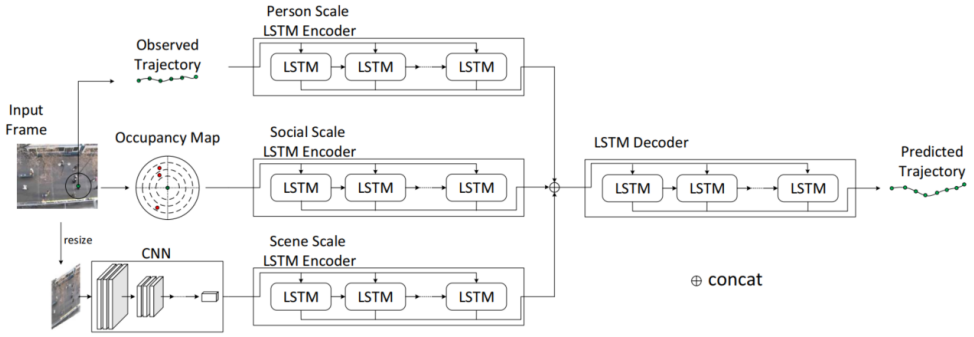


Figure 2.15: Architecture overview of SS-LSTM [27]. Seg-LSTM [52] has a similar architecture but uses a pre-trained Seg-Net as the feature extractor instead of three convolutions as done in SS-LSTM. Image taken from [27].

After the introduction of SoPhie, several works, for example, [53], [54] and [55], have used image information for trajectory prediction, usually together with social information. The use of image information is easily applicable in current datasets and in video surveillance scenario. However, in robotic and autonomous driving applications, it is difficult to assume that a camera with a bird's eye view on the scene is always available. Therefore, image information can realistically only be used in surveillance applications or very similar cases.

2.2.9 ConvLSTM in pedestrian trajectory prediction

LSTMs are capable of modelling relationships between time, but in pedestrian trajectory prediction it is also important to model spatial relationship, like in images. This is the reason behind the use convolutional LSTM (ConvLSTM) [56] for pedestrian trajectory prediction. A ConvLSTM has the same structure as a normal LSTM, but convolutions are used instead of matrix multiplications. This approach has shown promising results in the work of Yuke Li [25] that uses a 3d grid of displacements (like in [43]) and in the work of

Chen et al. [57] that also mixes social, spatial and image information. More information on the method proposed by Chen et al. can be found in Figure 2.16.

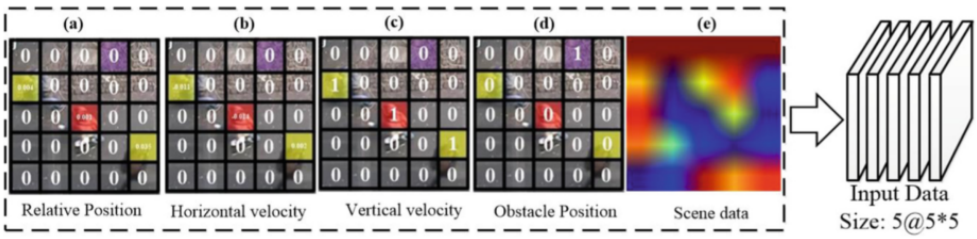


Figure 2.16: Input to the ConvLSTM in [57]. Five grids which are different for every pedestrian are used. These grids correspond to: the relative position of other pedestrians in each cell, horizontal and vertical velocities of other pedestrians in each cell, occupancy grid where cells with obstacles that are signalled with 1, and the result of feeding the scene around the pedestrian in a CNN. Image taken from [57].

2.2.10 Convolutions in pedestrian trajectory prediction

RNNs are not the only method applicable to tasks involving sequences: it is also possible to use Convolutional Neural Networks (CNN), as shown in [58]. A method that applies convolutions to pedestrian trajectory prediction was introduced by Nikhil and Morris in 2018 [10]. Their approach does not take into consideration social or spatial aspect for prediction, but it is still able to obtain comparable results with the recurrent architectures that also use other types of information. Unfortunately, the paper was published without peer review and there is not enough information in it to reproduce it, but it still confirms that convolutions can be used in trajectory prediction.

2.2.11 CVAE in trajectory prediction

Conditional Variational Auto-Encoders (CAVE) are an alternative architecture to CGAN for generating multiple paths for each agent. The encoder network encodes the future path using as condition the past trajectory, while the decoder network reconstructs the future path. At test time the encoded future path is substituted by a sample from a normal distribution and it is concatenated with the encoded past trajectory. This newly constructed vector is then

being fed to the decoder that generates the future trajectory.
This method has been successfully employed in the prediction of pedestrians, in works like [59], [53], [55] and [60] (shown in Figure 2.17), and in the predictions of vehicles trajectory, in works like [61].

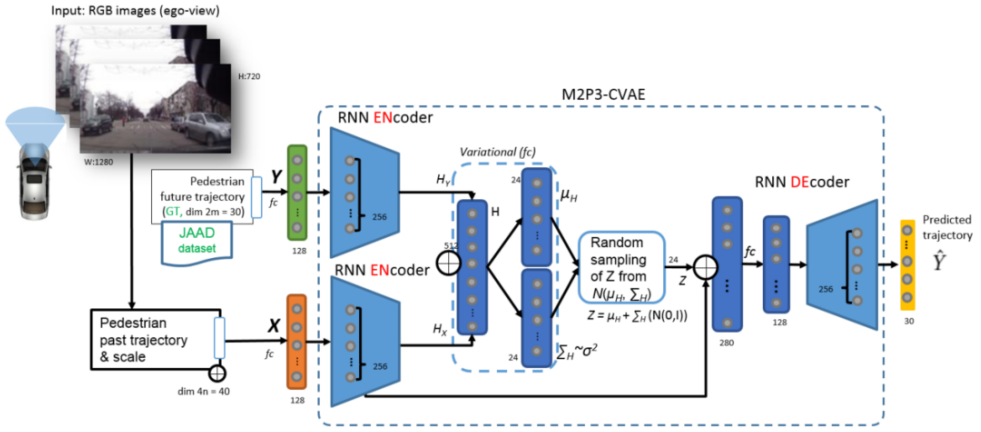


Figure 2.17: Architecture overview of M2P3 [60] during training. A variational auto-encoder conditioned on the past trajectory is used to predict the future trajectory. Image taken from [60].

2.2.12 Graph Neural Networks in pedestrian trajectory prediction

The first work using Graph Neural Networks(GNN) in pedestrian trajectory prediction was presented in 2017 by Vemula et al. in [6]. Their main contribution is the use of spatial-temporal graphs: between different pedestrians there are spatial connections that model how they interact with each other, while a single pedestrian a has temporal connections with itself that represent the trajectory. The main components of this architecture are the EdgeRNN, the nodeRNN and the attention module for deciding the importance of other pedestrians.
Following this first work, GNN for pedestrian trajectory have been used in other publications like [62], [63], [64], [65], [66], [54], [67] and [68].

Chapter 3

Datasets and Evaluation Methods

In this section, two aspects of primary importance in data-driven approaches are presented: the datasets and the evaluation methods.
The two datasets used in this thesis are the ETH-UCY dataset and the TrajNet dataset and are introduced in Section 3.1. Afterwards, in Section 3.2, the two quantitative metrics called Average Displacement Error (ADE) and Final Displacement Error (FDE) that are used to compare different model are presented.

3.1 Datasets

The use of data-driven methods requires the availability of quality data in sufficient quantity. More specifically, in pedestrian trajectory prediction the data available can be in two different formats: in image coordinates or real-world coordinates. Image coordinates means that each pedestrian is represented with the pixels it occupies in the camera image, while real-world coordinates means that each pedestrian is represented by its position in meters with origin in an arbitrary point of the world.
The choice of the coordinates format depends on the application: image coordinates are used in video surveillance applications, while real-world coordinates are used in robotics and autonomous driving applications. The focus of this thesis is on the autonomous driving application and therefore only datasets using real-world coordinates can be used.
The datasets used in this thesis are the ETH, UCY and TrajNet datasets because they are widely used in literature, publicly available and use real-world

coordinates.

3.1.1 ETH and UCY datasets

The most commonly used datasets in literature are the ETH and UCY datasets. The ETH (name taken from the ETH Zurich University) dataset [13] contains two scenes (named ETH and Hotel) taken from a bird's eye view. In total it comprises of 750 different pedestrians trajectories. One frame is annotated with pedestrian positions every 0.4 seconds.
The UCY (name taken from the University of Cyprus) dataset [14] contains three scenes (Zara1, Zara2 and Univ), taken from a bird's eye view. In total it comprises of more than 900 different pedestrians trajectories. One frame is annotated with pedestrian positions every 0.4 seconds. A frame from the Zara1 scene can be found in Figure 3.1.
The two datasets are usually used together: combined they contain five scenes (ETH, Hotel, Univ, Zara1 and Zara2), with more than 1600 pedestrian trajectories. The training and testing are done with the leave-one-out approach: the model is trained on four scenes and tested on the fifth, and this procedure is repeated five times, one for each scene.
Since these two datasets are mainly used in combinations with each other from now onward the two datasets together will be referred to as the ETH-UCY dataset.
The data in real-world coordinates for the ETH-UCY dataset used in this thesis has been downloaded from the Social GAN GitHub repository [69].

3.1.2 TrajNet dataset

The TrajNet dataset [15] is a curated collection of datasets, comprising in total of more than 8000 pedestrian trajectories. It merges the ETH, UCY, Stanford Drone Dataset and PETS2009 [70] datasets. The PETS2009 dataset contributes only with 19 pedestrians so it is not considered very relevant. The Stanford Drone Dataset [49] contributes to the majority of the pedestrian tracks and is a collection of videos taken with a drone at Stanford. It contains annotations of pedestrians, cars and bicycles. An example frame can be seen in Figure 3.2. It is available on the official site[71] with one frame annotated every 0.4 seconds, but coordinates are in the image reference frame. The authors of the TrajNet dataset transformed the Stanford Drone Dataset (SDD) in real-world coordinates so that all of the TrajNet dataset is in real-world coordinates. Their contribution doesn't stop here, in fact, they split the data from all the datasets

Figure 3.1: A frame of the UCY [14] Zara1 scene.

in training and test set, made the splits available on the official dataset site [72], and most importantly of all they did not release the ground truth of the test set. Instead, users upload their result to the official site that computes the metrics and (optionally) publishes the results on the online leaderboard. This has been done to "standardize" pedestrian trajectory prediction: methods have been evaluated on different subsets of available datasets, with contrasting coordinates systems and with different evaluation scripts. These inconsistencies have made challenging to objectively compare different techniques, therefore the TrajNet benchmark dataset has been proposed.
This standardization of metrics coupled with the online validation makes results on the TrajNet dataset reliable since they are computed by a third-party and not directly by researchers. Results obtained on this dataset are also very relevant, since it is the biggest publicly available collection of pedestrian trajectories.

3.1.3 Other datasets

In this subsection are presented additional datasets that are not used in this thesis but are still relevant in pedestrian trajectory prediction and are worth mentioning.
Datasets not used either because they are not available or because they are in image coordinates are: the Grand Central dataset [50], the CUHK Crow

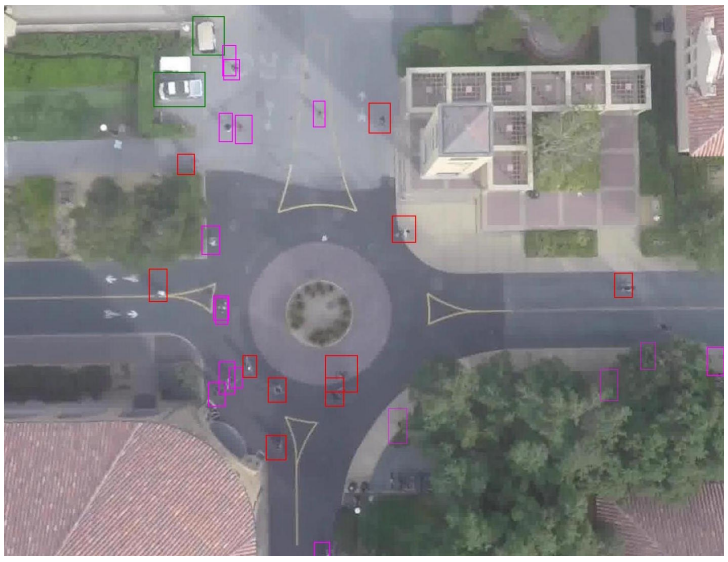


Figure 3.2: A frame of the Stanford Drone Dataset [49] death_circle scene.

dataset [73], the Edinburg Informatics Forum dataset [74] and the MuseumVisitors dataset [75].
Datasets not used because they are in image coordinates and are also not taken from a fixed camera but from a vehicle-mounted camera are: the KITTI [76] dataset, TrapHic [77] dataset and JAAD [78] datasets.
All of these datasets are interesting and studies have used them in the trajectory prediction task. However, they are not considered in this thesis because either not available, not in real-world coordinates or not used enough in literature to have meaningful baselines to confront with.

3.2 Metrics

In data-driven approaches, there is the need to standardize test settings and commonly used metrics to have quantitative results. These quantitative results permit scientists to understand how different models compare with each other, and what are their strength and their weaknesses.
The test setting and the metrics explained in this section are the same as the one used in Social LSTM [4] and in the majority of the publications dealing with pedestrian trajectory prediction with deep learning.

3.2.1 Test setting

The test setting tackles the problem of how trajectories should be observed and predicted. The most commonly used practice in literature is to observe 8 positions and predict the next 12 positions. Since the datasets used have one frame annotated every 0.4 seconds, this means that the model will observe for 3.2 seconds and predict the next 4.8 seconds. This is by far the most widely used setting and therefore, for the sake of comparing obtained results with literature, the results reported in this thesis are also obtained by observing 8 positions and predicting the next 12.

3.2.2 Average Displacement Error (ADE)

The first (and most important) metric that is used in pedestrian trajectory prediction is the Average Displacement Error (ADE), which was introduced in [13]. The ADE is the error over all the predicted points and the ground truth points from T_{obs+1} to T_{pred} averaged over all pedestrians. This error is calculated using the Euclidean distance between the predicted position and the real pedestrian position for each time steps. Formally the ADE formula is the following:

$$ADE = \frac{\sum_{i=1}^{n} \sum_{t=T_{obs}}^{T_{pred-1}} \left\| \hat{Y}_t^i - Y_t^i \right\|}{n(T_{pred} - T_{obs})} \tag{3.1}$$

The number of pedestrians is n, the predicted coordinates for pedestrian i at time t are $\hat{Y}_t^i$, the real future positions are Y_t^i and $||$ is the Euclidean distance. Since this thesis uses only datasets with real-world coordinate, the ADE is a measure in meters. That would not be the case if the dataset used is in image coordinates.
In essence, the ADE is the average distance (in meters) from every position of the prediction to every corresponding position of the real trajectory. This can be visualized in Figure 3.3.

3.2.3 Final Displacement Error (FDE)

The second metric that is used in pedestrian trajectory prediction is the Final Displacement Error (FDE), which was also introduced in [13]. The Final Displacement Error is the error between the final predicted position at T_{pred} and the real position at T_{pred}. This error is also calculated using the Euclidean distance. Formally the FDE formula is the following:

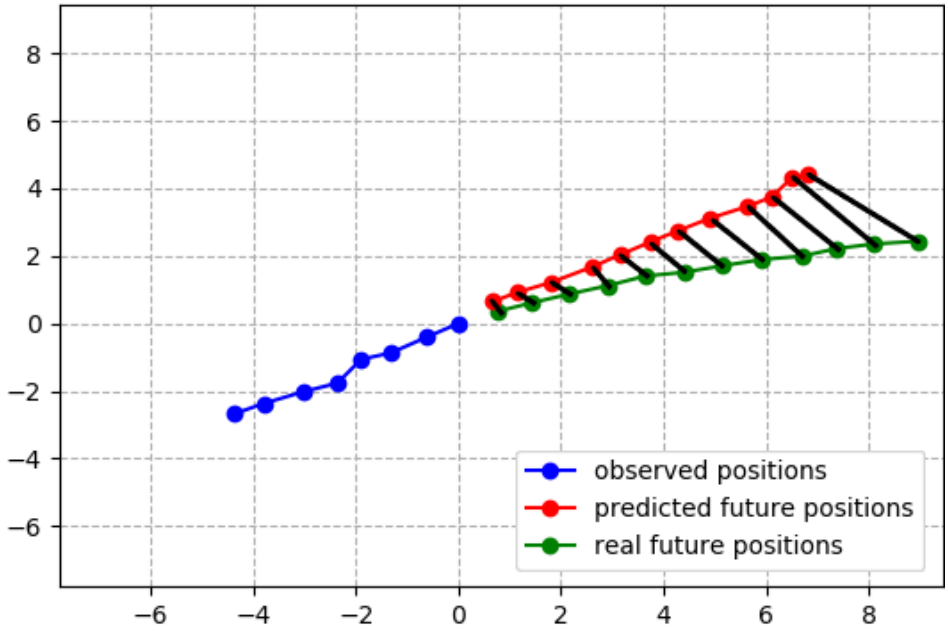


Figure 3.3: The black lines are the distance between every prediction and the corresponding real position. The Average Displacement Error is the length of the black line on average. The observed positions are 8 while the predicted positions and real future positions are 12, thus in this image is represented the standard test setting previously described.

$$FDE = \frac{\sum_{i=1}^{n} \left\| \hat{Y}^i_{T_{pred-1}} - Y^i_{T_{pred-1}} \right\|}{n} \tag{3.2}$$

The number of pedestrians is n, the predicted coordinates for pedestrian i at time t are $\hat{Y}^i_t$, the real future positions are Y^i_t and $||$ is the Euclidean distance. Since this thesis uses only datasets with real-world coordinate, the FDE is also a measure in meters. That would not be the case if the dataset used is in image coordinates.
In essence, the FDE is how big is the distance error (in meters) between the last predicted position and the last real position, as can be seen in Figure 3.4.

The Final Displacement Error is strongly correlated with the Average Displacement error: a model with a low ADE will also have a low FDE. This is confirmed by the results show in Section 4. Therefore, it could also be possible to consider only the Average Displacement Error when evaluating a model.

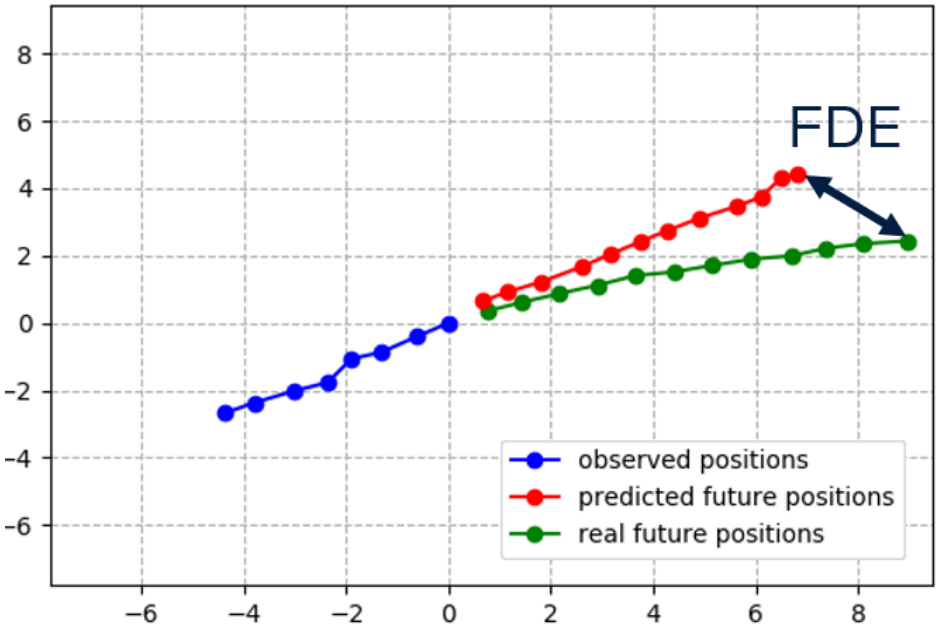


Figure 3.4: The black arrow is the distance between the last prediction and the last future position. The length of the arrow corresponds to the Final Displacement Error. The observed positions are 8 while the predicted positions and real future positions are 12, thus in this image is represented the standard test setting previously described.

Chapter 4

Models and results

Before developing any network it is important to define exactly which data the model will take as input: this aspect is discussed in Section 4.1. Regarding the network itself, recurrent models have been extensively used in literature for pedestrian trajectory prediction, and that is why an LSTM model and an Encoder-Decoder model have been implemented as baselines in Section 4.2. To improve the results obtained by the baselines, it is possible to change the loss, as explained in Section 4.3, and to apply effective data augmentation techniques, such as noise addition and random rotations, as explained in Section 4.4.
Recurrent models are not the only models that can be applied to time sequences: in Section 4.5 a convolutional model for trajectory prediction is presented, and it is able to outperform recurrent models. Furthermore, experiments on adding social information to the proposed models are presented in Section 4.6.
A comparison of all the implemented models with architectures from literature is then done in Section 4.7. Moreover, the presented models are also tested on newly acquired data in Section 4.8.

4.1 Input Coordinates

As stated in Section 1.1, the input and target data of models in pedestrian trajectory prediction are coordinates, however, the origin point of these coordinates was not specified. Therefore, is appropriate to raise the question of which coordinate system to use when feeding positions to the networks, as a form of data pre-processing. To answer this question, four coordinates pre-processing techniques have been identified:

1. **Absolute coordinates**. With absolute coordinates, we refer to the most naive approach possible: taking directly the coordinates from the datasets as they are, without any manipulation. This is not a sensible approach since each scene has the origin point in a different position. Thus, the network may not be able to produce meaningful results at test time when presented with a new scene that uses a different origin with respect to the training set.

2. **Coordinates with the origin in the first observation point** ($t = 0$). It does not matter from which scene the coordinates are from: to each one of the points in the sequence of 20 positions which represent the input and target coordinates is subtracted the first position in the sequence. This effectively puts the origin of all the sequence in the first observed position. In this way the coordinates are scene-independent and therefore this approach could solve the drawbacks of using absolute coordinates.

3. **Coordinates with the origin in the last observation point** ($t = T_{obs-1}$). Similar to the previous coordinates type, but with the difference that in this case the subtracted position is the one at $t = T_{obs-1}$, which is the last position the network will observe.

4. **Relative coordinates (velocities)**. In this case instead of coordinates with fixed reference systems, the network is fed with relative displacements. The first observed position will be (0,0), and all the subsequent input coordinates will be the relative displacements from the previous input. If the relative displacements are scaled accordingly to the annotations per seconds they become velocities. In that case, the input of the network is the velocity at each time-step, and the task is to predict the velocity at the next time step.

An example of the same trajectory represented in different coordinate types can be found in Figure 4.1.

4.2 Baselines

As baselines two recurrent models have been implemented: an LSTM and an Encoder-Decoder.

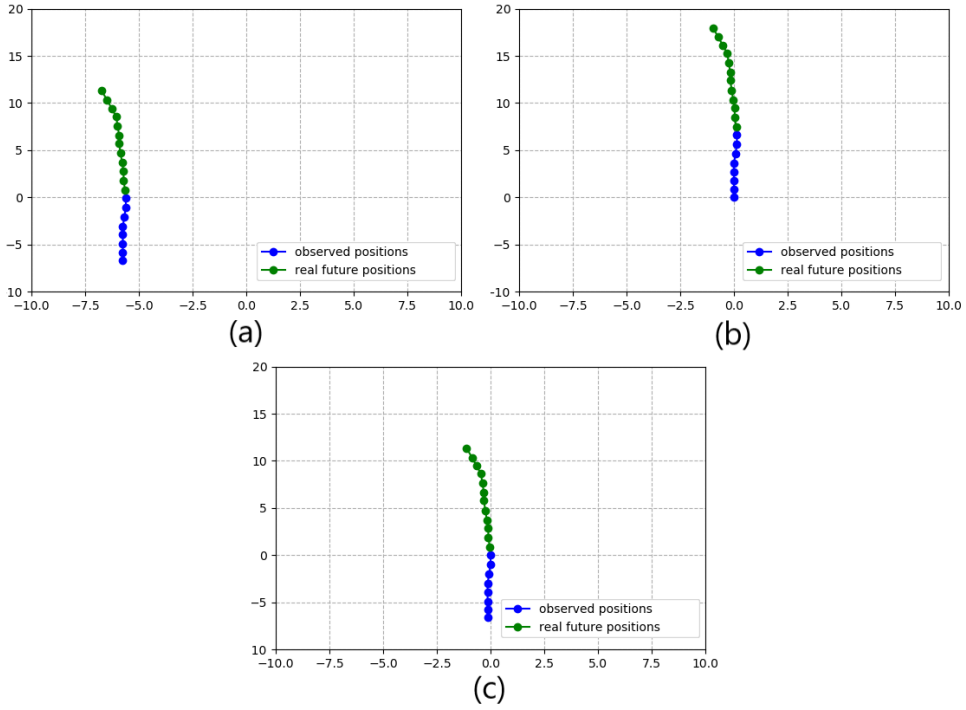


Figure 4.1: The same trajectory of a pedestrian going upward in three different coordinate systems. The observed positions are in blue and the future positions are in green. (a) Absolute coordinates, the coordinates are directly taken from the dataset. (b) Origin at time $t = 0$, the first observation point is the origin, independently of the original coordinates origin. (c) Origin at time $t = T_{obs-1}$, the last observation point is the origin, independently of the original coordinates origin.

4.2.1 LSTM baseline

The architecture for the LSTM baseline can be viewed in Figure 4.2. At the start the input coordinates are embedded in a feature vector by a fully connected layer. The features are then fed to the LSTM cell, and then the output of the cell is fed to other fully connected layers that output the future coordinates. During training teacher forcing, a technique first introduced in [79] and today widely used mainly in the machine translation field, is used to improve results.
Architecture's parameters are the following: the embedding dimension is 64, the LSTM cell size is 128, and the output of the LSTM passes through two fully connected layers, one with output dimension 64 and the other with output dimension 2. The model with two layers at the output (with ReLu non

linearity between the two) showed better results than the one with only one layer. The teaching forcing probability is set to 0.3.
It is to note that in this thesis the term "embed" is used the same way as it is used by Alahi et al. in Social-LSTM[4]: it is a technique in which a vector (usually containing the input coordinates or social information) is transformed into a fixed-length vector of features by a single fully connected layer.

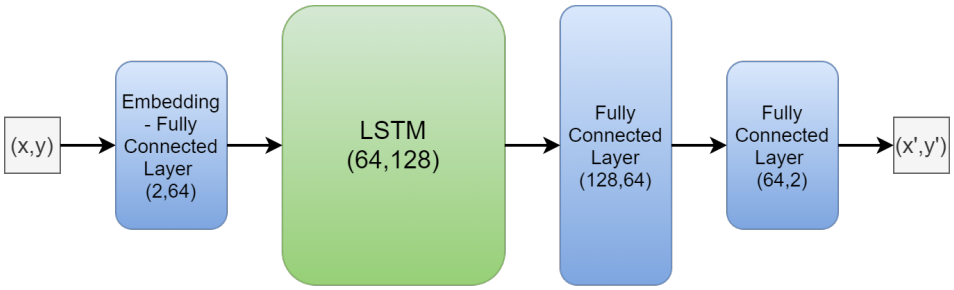


Figure 4.2: Architecture of the LSTM baseline. Layer's input and output dimensions are denoted with (input dimension, output dimension). For the LSTM cell input dimension and cell size are denoted with (input dimension, cell size).

Results obtained training the LSTM baseline with different coordinate types can be found in Table 4.1, Table 4.2 and Table 4.3, together with the effects of different losses, as explained in Section 4.3, and different scene frame rates, as explained in Section 4.3.1. All the networks in the three tables have been trained using the following parameters: batch size equal to 32, number of training epochs equal to 60 and learning rate initialized to 0.005 and further halved every 17 epochs. The parameters are the same for every network to have a fair comparison. Table 4.3 is the most relevant one since it uses the correct loss and the correct frame rate. From the results, it is evident that the best coordinate type is the one with origin in the last observation point. An explanation on the reason why is it the case is given in 5.1, and it is about the order of coordinates and the fact that the last observation point is the most important one, since it is the most recent one. Except for Table 4.1, Table 4.2 and Table 4.3, all the results reported in this chapter are obtained using coordinates with origin in the last observation point, because they produce better performances.

4.2.2 Encoder-Decoder baseline

The architecture of the Encoder-Decoder [24] baseline is very similar to the one presented in Section 2.2.2, but with the proper adjustments to be applied in pedestrian trajectory prediction. The proposed architecture uses an LSTM cell for both the encoder and the decoder, as shown in Figure 4.3.
When time $t \in [0, ..., T_{obs-1}]$ the input coordinates are first embedded by a fully connected layer and then fed to the LSTM encoder. At time $t = T_{obs-1}$ the cell state of the LSTM encoder is copied into the LSTM decoder. When $t \in [T_{obs}, ..., T_{pred-1}]$ the coordinates are first embedded by the decoder fully connected layer, and then fed to the LSTM decoder. The decoder output passes through two fully connected layers with the same architecture as in the LSTM baseline, and the final output is the next position, which is then fed at the next time-step to the decoder.
Thus, the functioning of the Encoder-Decoder baseline is very similar to the LSTM baseline. Teacher forcing is also used in the Encoder-Decoder model. The architecture's parameters are the following: the embedding dimension is 64, the LSTM cells size is 128, the output of the decoder passes through two fully connected layers, one with output dimension 64 and the other with output dimension 2, and the teacher forcing probability is set to 0.3.

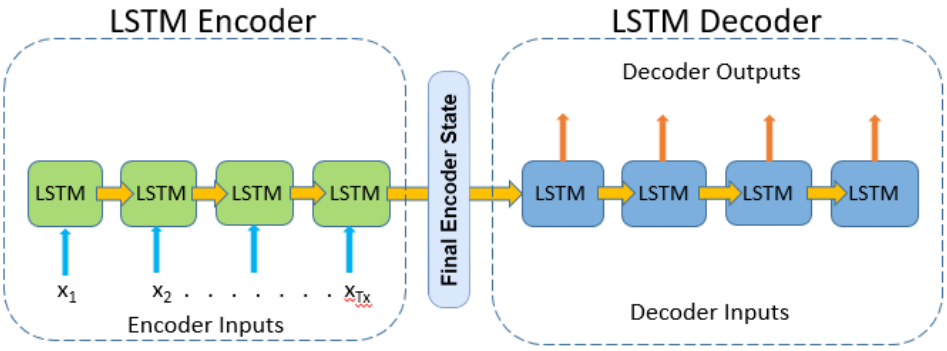


Figure 4.3: Architecture of the LSTM encoder-decoder baseline.

4.3 Loss

When training a network it is very important to choose the most appropriate loss. The simplest approach to the loss choice is to use the Mean Square

Error(MSE): for each couple of predicted and real coordinates, the squared difference is averaged on all time steps. However, when assessing the performance of the models the Average Displacement Error is used, not the Mean Square Error. Therefore using the same metric, the Average Displacement Error, both during training and during testing, makes the training more consistent with the test setting.
If only one pedestrian is considered, if the predicted coordinates at time t are $(\hat{x}_t, \hat{y}_t) \in \hat{Y}$ and the real coordinates at time t are $(x_t, y_t) \in Y$, the formulas of the two losses can be compared:

$$MSE = \frac{\sum_{t=T_{obs}}^{T_{pred}-1}(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2}{(T_{pred} - T_{obs})} \tag{4.1}$$

$$ADE = \frac{\sum_{t=T_{obs}}^{T_{pred}-1}\sqrt{(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2}}{(T_{pred} - T_{obs})} \tag{4.2}$$

The two formulas are very similar, but the difference in performance is substantial. The best network in Table 4.2, which uses the MSE as loss, reaches an average error of 0.575, while the best one in and Table 4.3, which uses the ADE as loss, reaches an average error of 0.535. These results show the importance of having a loss coherent with the testing criteria.
Therefore, except for Table 4.1 and 4.2, all the results reported in this chapter are obtained using the ADE as loss.

	Absolute Coordinates	Origin at t=0	Origin at t=T_{obs-1}	Relative Coordiantes
ETH	2.451 / 3.732	1.151 / 2.043	**1.021 / 1.950**	1.061 / 2.134
Hotel	8.483 / 8.754	0.642 / 1.217	0.661 / 1.383	**0.631 / 1.311**
Univ	1.531 / 2.490	0.903 / 1.571	**0.676 / 1.386**	0.652 / 1.381
Zara1	0.561 / 1.085	0.452 / 0.953	**0.422 / 0.903**	0.462 / 1.000
Zara2	0.666 / 1.287	0.351 / 0.772	**0.350 / 0.751**	0.364 / 0.793
Average	2.744 / 3.475	0.704 / 1.315	**0.623 / 1.273**	0.633 / 1.323

Table 4.1: Results obtained training the LSTM baseline on the ETH-UCY datasets. The loss used was the Mean Square Error and the ETH scene has a frame rate issue. Results are in the format ADE / FDE and the best results are in bold.

	Absolute Coordinates	Origin at t=0	Origin at t=T_{obs-1}	Relative Coordinates
ETH	1.802 / 2.933	0.848 / 1.594	**0.756 / 1.452**	0.805 / 1.637
Hotel	12.277 / 13.881	0.625 / 1.230	0.672 / 1.329	**0.582 / 1.218**
Univ	1.635 / 2.504	0.758 / 1.432	**0.666 / 1.358**	0.688 / 1.473
Zara1	0.640 / 1.265	0.471 /1.004	**0.434 / 0.936**	0.453 / 0.982
Zara2	0.722 / 1.223	0.361 / 0.788	**0.344 / 0.753**	0.361 / 0.792
Average	3.415 / 4.347	0.612 / 1.210	**0.575 / 1.165**	0.578 / 1.220

Table 4.2: Results obtained training the LSTM baseline on the ETH-UCY datasets. The loss used was the Mean Square Error and the ETH scene has the frame rate issue corrected. Results are in the format ADE / FDE and the best results are in bold.

	Absolute Coordinates	Origin at t=0	Origin at t=T_{obs-1}	Relative Coordinates
ETH	7.499 / 7.961	0.779 / 1.509	**0.734 / 1.432**	0.747 / 1.450
Hotel	6.769 / 8.733	0.546 / 1.101	**0.501 / 1.053**	0.589 / 1.186
Univ	1.278 / 2.061	0.729 / 1.452	**0.687 / 1.430**	0.688 / 1.447
Zara1	0.498 / 1.042	0.425 / 0.925	**0.424 / 0.920**	0.445 / 0.951
Zara2	0.464 / 0.962	**0.318 / 0.702**	0.330 / 0.719	0.324 / 0.708
Average	3.302 / 4.152	0.559 / 1.138	**0.535 / 1.111**	0.558 / 1.149

Table 4.3: Results obtained training the LSTM baseline on the ETH-UCY datasets. The loss used was the Average Displacement Error and the ETH scene is the correct one. Results are in the format ADE / FDE and the best results are in bold.

4.3.1 ETH framerate issue

As stated in Section 3.1.1 the data for the ETH-UCY dataset was downloaded from [69]. However, in Table 4.1 results on the ETH scene are considerably worse when compared to other scenes. As suggested by SR-LSTM [34], there might be a frame rate issue in that video, which can be corrected considering annotated one every 6 frames and not one every 10 as in the other scenes.
As it is possible to see in Table 4.2, after correcting the frame rate, results on the ETH scene are more in line with the other scenes. Therefore, except in table 4.1, all the results reported on the ETH-UCY dataset use the frame rate correction on the ETH scene.

4.4 Data augmentation

In the ETH-UCY dataset there are more than 1600 pedestrians trajectories. These trajectories have a variable length, but the models need samples of exactly length 20, 8 for observation and 12 for prediction. Therefore, to create the training set, every possible 20-length subset of the pedestrian tracks is taken. This means that, for example, from a pedestrian with 30 known positions, it is possible to have 11 samples. With this technique, the ETH-UCY dataset contains in total around 34000 samples that can be fed to a network. The TrajNet dataset contains more pedestrians than the ETH-UCY dataset, but it only consists of 11000 samples. This is because the authors directly provide the samples and not the raw trajectories.
The data points available are sufficient to train a deep neural network and obtain significant results, as Table 4.3 can show. However, the same network with more data will probably perform better since the total number of samples is quite limited. More data from these datasets is unfortunately not available, but it is possible to implement data augmentation techniques to get the most out of the accessible data. To do so the following techniques are proposed:

1. Apply a **random rotation** to each sample. This will make the network learn patterns in a rotation-invariant matter. Thus, it should not matter if in the test set there is a pedestrian going left to right and in the training set there are only people going right to left: the network should perform well in every case.

2. **Mirror** the trajectory on the x-axis or y-axis with a probability. No rotation applies a mirroring, therefore mirroring could enhance the effects or random rotations. This technique is implemented in the following way: there is a 25% probability of mirroring a sample on the x-axis, a 25% probability to mirroring it on the y-axis and a 50% probability of not applying any mirroring at all. An example of both random rotation and mirroring applied to a trajectory can be found in Figure 4.4.

3. Apply **Gaussian noise** with mean 0 to every point. This should make the network more robust to small perturbations and imprecisions in the training data. In this way, the model should concentrate more on the high-level patterns of pedestrians.

These techniques have been applied to the LSTM baseline trained on the ETH-UCY dataset, and the results can be viewed in Table 4.4. Rotating and adding noise produce a massive error reduction on the ETH and Hotel scenes,

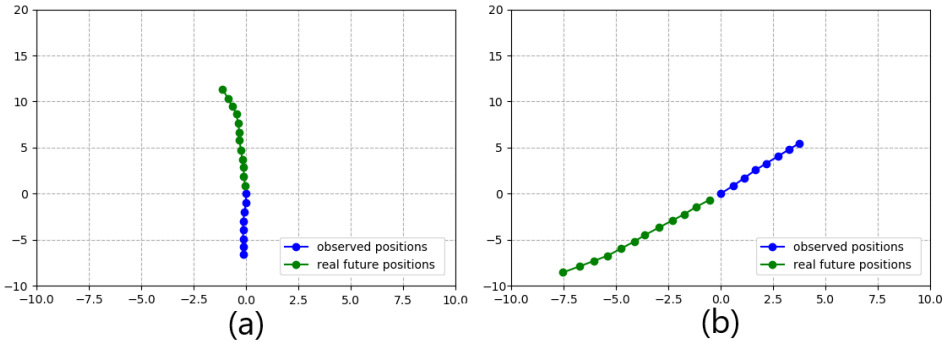


Figure 4.4: The effects of rotation and mirroring on a trajectory. (a) A pedestrian trajectory with the origin in the last observation point (b) The same pedestrian trajectory rotated and mirrored.

however, the combined application of rotations and noise slightly degrades the performances on the Univ, Zara1 and Zara2 scenes. Mirroring, instead, does not seems to add any benefit. Nevertheless, applying all the previously mentioned data augmentation techniques improves the average network performances, reaching an average ADE of 0.446.

	Gaussian noise	Random rotations	Random rotations and mirroring	Random rotations, mirroring and Gaussian noise
ETH	0.621 / 1.249	0.689 / 1..331	0.689 / 1.314	**0.581 / 1.168**
Hotel	0.421 / 0.865	0.305 / 0.576	0.314 / 0.599	**0.259 / 0.503**
Univ	0.698 / 1.447	**0.549 / 1.199**	0.550 / 1.199	0.578 / 1.241
Zara1	**0.428 / 0.917**	0.439 / 0.971	0.432 / 0.961	0.463 / 1.022
Zara2	0.334 / 0.712	0.330 / 0.728	**0.329 / 0.719**	0.346 / 0.748
Average	0.500 / 1.038	0.462 / 0.961	0.463 / 0.959	**0.446 / 0.936**

Table 4.4: Results obtained training the LSTM baseline (with coordinates with the origin in the last observation point) on the ETH-UCY dataset with the use of different data augmentation techniques. The loss used was the Average Displacement Error and the ETH scene is the one with the correct frame rate. Results are in the format ADE / FDE and the best results are in bold.

Findings about coordinates and data augmentation do not only apply to the LSTM baseline, but also to the Encoder-Decoder baseline, which reaches similar, but slightly worse, results, as can be seen in Table 4.5.
Results regarding the data augmentation techniques are discussed in more details in Section 5.2. Since the use of Gaussian noise and random rotations

massively decreases the error, all the results reported in the following sections (Section 4.5, 4.6, 4.7 and 4.8) are obtained training the networks with random rotations and Gaussian noise.

	Encoder-Decoder with random rotations, mirroring and Gaussian noise	LSTM with random rotations, mirroring and Gaussian noise
ETH	0.585 / 1.170	**0.581 / 1.168**
Hotel	**0.246 / 0.491**	0.259 / 0.503
Univ	0.589 / 1.245	**0.578 / 1.241**
Zara1	0.467 / 1.023	**0.463 / 1.022**
Zara2	0.360 / 0.771	**0.346 / 0.748**
Average	0.449 / 0.938	**0.446 / 0.936**

Table 4.5: Comparisons of the two baselines with data augmentation. Results are in the format ADE / FDE and the best results are in bold.

4.5 Convolutional Model

As shown by works such as [11] and [12], convolutional neural networks can be used in problems involving sequences, such as machine translation or image captioning, achieving competitive results in comparison with recurrent neural networks. It has also been shown by Nikhil and Morris in [10], that indeed convolutional neural networks can be applied to pedestrian trajectory prediction. However, their model is not explained in enough detail to be reproduced. One of the aims of this thesis is to understand how a convolutional model for pedestrian trajectory prediction can be implemented and how it performs in comparison to recurrent models. After extensive implementation and testing, the convolutional model in Figure 4.5 represent a valid alternative to recurrent models in the pedestrian trajectory prediction field.

At the start, all the input positions are embedded in 64-length vectors, similarly to what is done in the recurrent baselines, but with the difference that in this case the embedding is done for all the input coordinates at the same time. After this first step, the input trajectory is represented by features that are arranged in a 64x8 matrix, in which 64 is the embedding dimension and 8 is the number of input positions. This matrix can also be represented as 64 one-dimensional channels with 8 features each, one for each time step. Thus, it is possible to apply 1D convolutions to this matrix.

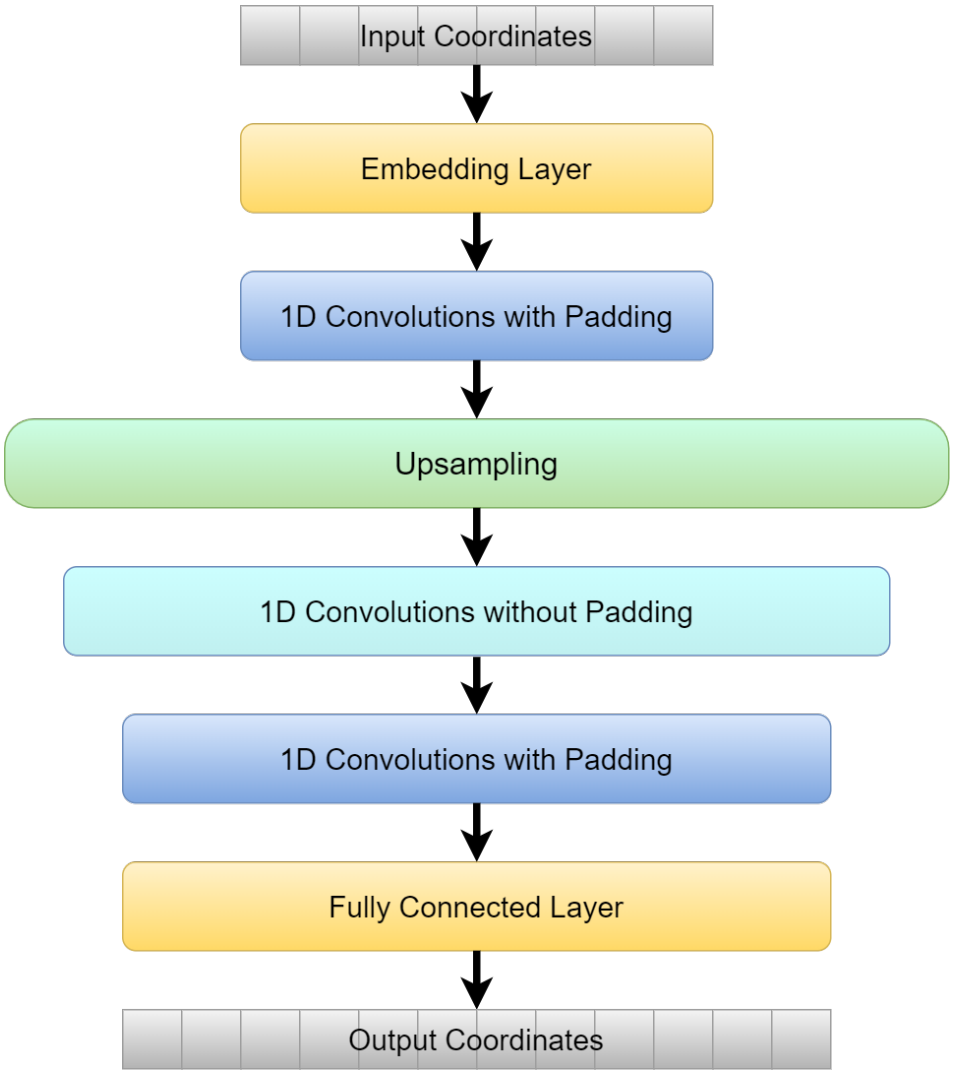


Figure 4.5: Proposed architecture for the convolutional model.

After the embedding, the first group of 1D convolutions with padding is applied. The padding depends on the kernel size of the convolutions and it is employed to keep the number of features in output the same as the number of features in input. This means that as many convolutional layers as wanted can be stacked at this step. After the first convolutions group, an upsampling layer is applied to double the number of features from 8 to 16. Afterwards, convolutional layers without padding are applied to reduce the number of features from 16 to 12, the number of positions the model needs to output. Lastly, the second group of convolutions with padding is applied and then a fully connected layer transforms each feature vector in an output position.

This convolutional model is scalable, in a sense that there is no limit at the number of layers in the initial and final convolutions groups. It is also one-shot, in a sense that in one pass all the output coordinates are generated, differently from recurrent models where one pass gives only the next position. An in-depth analysis of the differences between the convolutional model and the recurrent model can be found in Section 5.3.
The insights obtained in the previous sections regarding coordinate systems and data augmentation still apply to the convolutional model. The best coordinate system to use with the convolutional model is the one with origin in the last observation point. Moreover, the use of Gaussian noise and random rotations improves performances especially in the ETH and Hotel scene, with a small increase of the error in the Univ, Zara1 and Zara2 scenes. Therefore, all the convolutional model results reported in this and in the following sections are obtained using coordinates with the origin in the last observation point, Gaussian noise and random rotations.

Some variations of the basic convolutional model of Figure 4.5 have also been explored:

1. Apply **positional embeddings** on the inputs. As proposed by [11], to give the information of order in the input data, the positional information of each input is used. This means that the input of the first convolutional group is the embedding of the input coordinates summed with the embeddings of the numerical position that each coordinate has in the sequence. Thus, the first pedestrian position will be embedded and summed to the embedded representation of the number '0' in a one-hot-vector. This is then repeated for all the input pedestrian positions.

2. Use **transpose convolutions** instead of the upsampling layer followed by convolutions without padding, to transition from 8 features to 12 features.

3. Use **residual connections**. First introduced in [80], residual connections help information and gradient flow especially in very deep architectures. In this architecture variation, all convolutional layers are transformed in residual convolutional layers.

4. Use **2D convolutions** instead on 1D convolutions. To do that the 64x8 matrix that is created with all the embeddings needs to be considered as an image with only one channel. 2D convolutions can then be applied to this artificial image. However, the 2D convolutions increase the number

of channels, and therefore the final convolutional layer needs to decrease the channels number to one so that the final fully connected layer can be applied.

The results obtained training various versions of the convolutional model can be found in Table 4.6.

	ETH	Hotel	Univ	Zara1	Zara2	Average
Base model, kernel size 3	0.605 / 1.190	0.264 / 0.509	0.588 / 1.241	0.521 / 1.095	0.351 / 0.755	0.466 / 0.958
Base model, kernel size 7	0.560 / 1.149	0.246 / 0.427	0.590 / 1.249	0.478 / 1.046	**0.346 / 0.737**	0.444 / 0.931
Kernel size 7, positional embedding	0.568 / 1.125	0.248 / 0.467	0.594 / 1.257	0.459 / 0.990	0.369 / 0.789	0.447 / 0.926
Kernel size 7, transpose convolutions	0.606 / 1.197	0.267 / 0.517	0.595 / 1.254	**0.451 / 0.989**	0.356 / 0.762	0.455 / 0.944
Kernel size 7, residual connections	0.560 / 1.121	0.245 / 0.470	0.589 / 1.251	0.516 / 1.073	0.349 / 0.741	0.452 / 0.931
2D convolutions, kernel size 5	**0.559 / 1.114**	**0.240 / 0.464**	**0.581 / 1.225**	0.456 / 0.993	0.347 / 0.751	**0.436 / 0.909**

Table 4.6: Comparisons of the different convolutional models. Results are in the format ADE / FDE and the best results are in bold.

The model using 2D convolutions (shown in detail in Figure 4.6) is the one that performs better with an average ADE of 0.436, outperforming the 0.446 ADE of the LSTM baseline. Using positional embedding, transpose convolutions and residual connections did not improve performances, while increasing the kernel size improved them. An increase kernel size means that the network is able to see more information with one layer and thus can create better predictions. This is why the model with kernel size 7 outperforms the model with kernel size 3 and why the 2D convolutional model (in which each kernel sees more information than the kernels in the 1D convolutional model) outperforms the 1D convolutional model. The presented results are discussed more in detail in Section 5.4.
Thus, it is possible to conclude that it is feasible to develop a convolutional model for trajectory pedestrian prediction that, with hyperparameter tuning and the use of 2D convolutions, is able to outperform recurrent models.

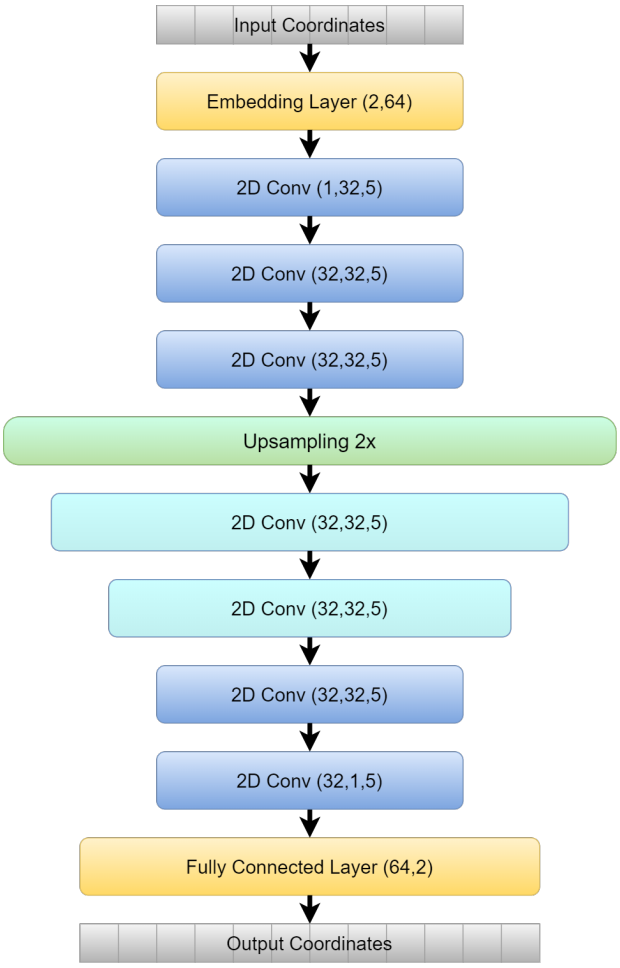


Figure 4.6: Convolutional model that uses 2D convolutions. For the fully connected layers in the parenthesis there are input and output dimensions. For convolutional layers in the parenthesis there are input channels, output channels and kernel size. All the convolutions have padding 2 so that the output dimension is the same as the input dimension except for the two convolutions after the upsampling layer that have padding of 1. Each layer has a corresponding batch normalization layer.

4.6 Adding social information

Until now, the models presented do not use any other information than the past trajectory. However, great emphasis has been put in including social information in the literature, thus it is also interesting to show how the models presented until now perform when social information is included.

4.6.1 Occupancy information

One of the main ways in which social information is represented is thought occupancy information: the space around the pedestrian is divided into multiple areas, and for each area, it is noted if it contains a pedestrian or not. The vector containing pedestrians occupancy in each area is then fed to the network. After analyzing what has been done in literature the three following occupancy techniques for representing social information have been implemented:

1. A square occupancy grid, like the one used in O-LSTM [4].
2. A circular occupancy matrix, like the one used in SS-LSTM [27].
3. A "radar", introduced in [8]. The angular space around a pedestrian is divided in a number of directions (for example 36, one for every 10 degrees) and then the closer pedestrian in that direction, within a certain range, is computed. If there are no pedestrians in that direction the maximum range is used as the value.

Examples of these techniques can be visualized in Figure 4.7.

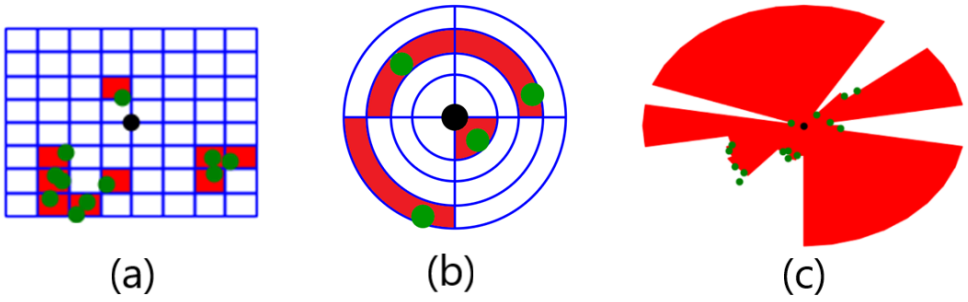


Figure 4.7: Various methods for representing the occupancies of nearby pedestrians in the space. (a) Square occupancy grid. (b) Circular occupancy matrix. (c) Occupancy based on direction and distance (like a "radar"). The current pedestrian is in black and the other pedestrians are in green. For (a) and (b) the occupied space is in red, while for (c) the free space is in red.

The occupancy grid is represented with a matrix $l * l$ where l is the number of cells on each side. The circular occupancy is represented with a matrix $c * 4$ where c is the number of circles, and finally, the "radar" can be represented by a vector of length $int(360/d)$, where d is the number of degrees an element of the vector represents. Social information which is not already in vector form

is flattened to be used as an input to the models.
Social information can be integrated both in the convolutional model and in the Encoder-Decoder model. The convolutional model requires minimal modifications: the social information is embedded by another fully connected layer, then is summed with the embedded positions and finally fed to the convolutional network that has the same architecture as the one in Figure 4.6. The social convolutional architecture can be seen in Figure 4.8.

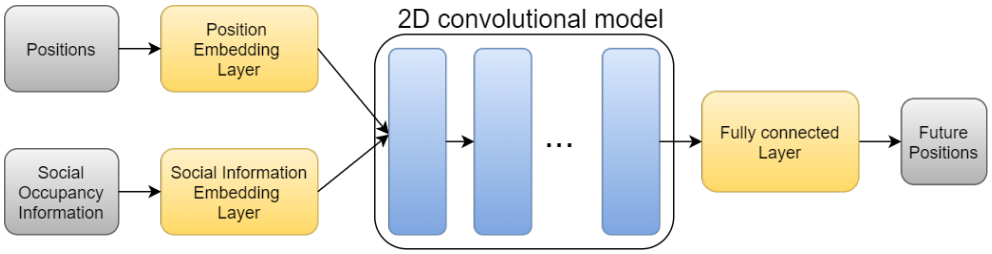


Figure 4.8: Convolutional model modified to accept social information as input. The convolutional model in blue is the same as the one in Figure 4.6. Embedding layers in yellow are fully connected layers.

The LSTM architecture has not been adapted to include social information because it would need to have social information as input not only when $t \in [0, ..., T_{obs-1}]$, but also when $t \in [T_{obs}, ..., T_{pred-1}]$. This would mean either giving to the network the real occupancies also during $t \in [T_{obs}, ..., T_{pred-1}]$, which would be impossible to do in a real-time scenario since the future is not available, or computing simultaneously the future position of all the pedestrians in the scene at each time step. This last technique proved to be very complex to implement and very computationally intensive, and was abandoned.
As for adapting the Encoder-Decoder model to use social information, the decoder part remains the same, however, there are now two encoders, one for the positions and one for the social information. Their states are then concatenated to become the decoder hidden state. This architecture can be seen in Figure 4.9. The double encoder is a widely use technique in literature, present in works like [27], [52] and [8].
Results obtained using the different types of social occupancy information with both networks can be found in Table 4.7.
The results obtained using occupancy social information are not better than the ones obtained without it. For example, the best occupancy convolutional model, the one with radar information, reaches 0.438 of average ADE, while the model without social information reaches 0.436 of average ADE. This might be caused by the fact that occupancy methods for modelling social infor-

mation are too simple and do not capture the complexity of social scenarios. This idea is discussed more in-depth in Section 5.6.1.

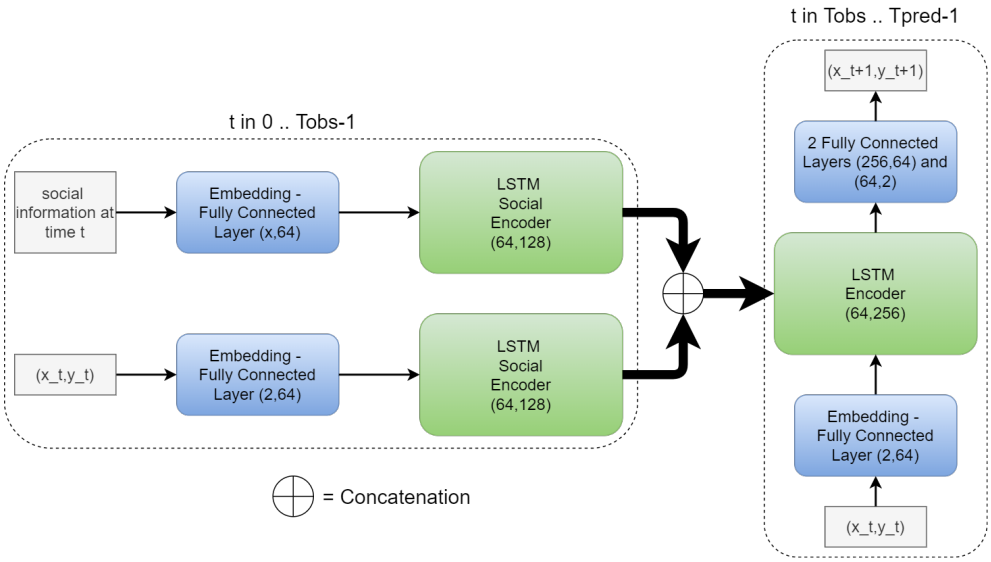


Figure 4.9: Encoder-Decoder model modified to use social information. Fully connected layers have in parenthesis their input and output dimensions, while LSTM cells have in parenthesis their input dimension and cell dimension. The dimension of the social vector is denoted with x.

	ETH	Hotel	Univ	Zara1	Zara2	Average
EncDec, Grid	0.569 / 1.139	0.224 / 0.424	0.616 / 1.288	0.478 / 1.028	0.358 / 0.765	0.449 / 0.929
EncDec, Circle	0.607 / 1.206	0.246 / 0.473	0.633 / 1.323	0.500 / 1.070	0.375 / 0.787	0.472 / 0.972
EncDec, Radar	0.585 / 1.172	0.232 / 0.448	0.597 / 1.255	0.483 / 1.054	0.352 / 0.754	0.450 / 0.937
Conv2D, Grid	0.558 / 1.118	0.233 / 0.445	0.604 / 1.269	0.464 / 1.005	0.342 / 0.740	0.440 / 0.915
Conv2D, Circle	0.561 / 1.122	0.235 / 0.447	0.590 / 1.240	0.461 / 0.991	0.348 / 0.746	0.439 / 0.910
Conv2D, Radar	0.567 / 1.109	0.235 / 0.449	0.589 / 1.231	0.464 / 0.997	0.337 / 0.719	0.438 / 0.901

Table 4.7: Results obtained adding social occupancy as input to the 2D convolutional model and to the Encoder-Decoder model. Grid results are obtained using 10 cells per side ($l = 10$) and each cell has a side of 0.5m in the real world. Circle results are obtained using 12 circles ($c = 12$) 0.5m apart from each other. Radar results are obtained using 8 degrees per element ($d = 8$). These parameters are the ones that produced the best results.

4.6.2 Social vector information

Including social occupancy information proved to be ineffective, but in literature a lot of other methods to include social information exist. One of such methods is proposed in [40] and consists of using directly the positions of nearby pedestrians, not an approximation of it as it happens when using occupancy information.
Changing the shape of the social information requires only some minor modifications to the convolutional architecture of Figure 4.8 and to the Encoder-Decoder architecture of Figure 4.9. What changes from the last section is the way social information is created and how it is embedded: after the embedding step the architecture is the same.
To include social information in a similar way to [40], for each considered pedestrian the positions and velocities of all nearby pedestrians from time 0 to time T_{obs-1} are gathered. Then all of these positions are put in the reference system of the last observation point of the considered pedestrian, to have a unique origin both for the trajectory and for the social information. Then, for each time-step, nearby pedestrians positions are sorted based on their distance from the considered pedestrian position at that time. Only the closest N pedestrian are then used for the social information. If there are less than N nearby pedestrians placeholder values are used.
At the end of this pre-processing, the social information is represented in a matrix of dimension $T_{obs} * N * 4$, in which N is the number of pedestrians and 4 means that each pedestrian is represented with both its position and its velocity. At each time step, the matrix of dimension $N * 4$ is fed to the network. Three methods for effectively embedding social information at each time step have been tested:

1. Method 1, the same as [40] (visualized in Figure 2.10): each nearby pedestrian position and velocity is embedded, with also an attention factor. This new vector then passes trough two fully connected layers. At the end to have the same social vector dimension as the coordinate embedding dimension, a maxPool layer is applied to all the vectors derived from the nearby pedestrians positions and velocities.

2. Method 2: each nearby pedestrian position and velocity is embedded by a fully connected layer. Then, to have the same social vector dimension as the coordinate embedding dimension, a maxPool layer is applied to all the vectors derived from the nearby pedestrians positions and velocities.

3. Method 3: each nearby pedestrian position and velocity is embedded

by a fully connected layer. After that, a fully connected layer takes as input all the embeddings and outputs the final social vector. No maxPool operation is used in this method.

Results obtained using these three methods are reported in Table 4.8.

	ETH	Hotel	Univ	Zara1	Zara2	Average
EncDec, Method 1	0.580 / 1.166	0.230 / 0.448	0.593 / 1.250	0.486 / 1.056	0.355 / 0.748	0.449 / 0.934
EncDec, Method 2	0.642 / 1.256	0.228 / 0.434	0.613 / 1.270	0.477 / 1.010	0.358 / 0.761	0.463 / 0.946
EncDec, Method 3	0.624 / 1.22	0.231 / 0.436	0.624 / 1.30	0.462 / 1.003	0.369 / 0.779	0.462 / 0.948
Conv2D, Method 1	0.593 / 1.183	0.247 / 0.468	0.599 / 1.262	0.443 / 0.954	0.338 / 0.734	0.444 / 0.920
Conv2D, Method 2	0.589 / 1.162	0.241 / 0.457	0.574 / 1.222	0.443 / 0.956	0.340 / 0.736	0.438 / 0.907
Conv2D, Method 3	0.593 / 1.186	0.235 / 0.442	0.613 / 1.279	0.445 / 0.970	0.339 / 0.730	0.445 / 0.921

Table 4.8: Results obtained using social social information under the form of nearby pedestrians position and velocities, with the 2D convolutional and Encoder-Decoder models. Results are obtained considering 7 nearby pedestrians.

Methods proposed in this subsection for including social information produced similar results to the ones obtained with occupancy methods. Thus, it is possible to conclude that the methods presented in this subsection and in the previous one to include social information proved to be ineffective. This might be due to their simplicity: more complex method have been proposed in literature and have been show to work. This aspect is discussed in more detail in 5.6.1.

4.7 Comparison with literature

In this section the ADE and FDE of models presented in literature and of models developed in this thesis are compared. This comparison is done first on the ETH-UCY dataset and then on the TrajNet dataset.

4.7.1 ETH-UCY dataset results comparison

The ETH-UCY dataset is the most used dataset in literature for pedestrian trajectory prediction. The following models from literature have been chosen to do a comparison with the results obtained in this thesis:

- Linear Velocity, a linear regressor that estimates linear parameters by minimizing the least square error, taken from [5].
- A simple LSTM, trained by [5];
- Social-LSTM [4], trained by [5];
- Convolutional Neural Networks for trajectory prediction [10], convolutional model for pedestrian trajectory prediction, by Nijhil and Morris;
- Social-GAN [5], a generative model that uses social information;
- Sophie [9], a generative model that uses both social and image information;
- Stochastic Trajectory Prediction with Social Graph Network (Stochastic GNN) [67], generative model that uses social information and graph neural networks;
- MCNET [53], generative model based on a conditional variational auto-encoder that uses both social and image information;
- Conditional Generative Neural System [55], generative model based on a conditional variational auto-encoder that uses both social and image information;
- Social-BiGAT [54], generative model that uses both social and image information;
- SR-LSTM [34], model based on the state refinement of the LSTM cells of all the pedestrians in the scene to take into for account social interaction;
- Social spatio-temporal graph convolutional neural network (STCGCNN) [68], generative model that uses social information and graph convolutional neural networks;
- STGAT[64], generative model that uses social information and graph convolutional neural networks;

The result comparison can be found in Table 4.9.

	ETH	Hotel	Univ	Zara1	Zara2	Average
Linear Velocity (from [5]	1.33 / 2.94	0.39 / 0.72	0.82 / 1.59	0.62 / 1.21	0.77 / 1.48	0.79 / 1.59
Social LSTM [4] (from [5])	1.09 / 2.35	0.79 / 1.76	0.67 / 1.40	0.47 / 1.00	0.56 / 1.17	0.72 / 1.54
LSTM (from [5])	1.09 / 2.41	0.86 / 1.91	0.61 / 1.31	0.41 / 0.88	0.52 / 1.11	0.70 / 1.52
CNN [10]	1.04 / 2.07	0.59 / 1.17	0.57 / 1.21	0.43 / 0.90	0.34 / 0.75	0.59 / 1.22
Social GAN [5]	0.81 / 1.52	0.72 / 1.61	0.60 / 1.29	0.34 / 0.69	0.42 / 0.84	0.58 / 1.18
Sophie [9]	0.70 / 1.43	0.76 / 1.67	0.54 / 1.24	**0.30 / 0.63**	0.38 / 0.78	0.54 / 1.15
Stochastic GNN [67]	0.75 / 1.63	0.63 / 1.01	0.48 / 1.08	**0.30 / 0.65**	0.26 / 0.57	0.49 / 1.01
MCNET [53]	0.75 / 1.61	0.37 / 0.68	0.58 / 1.18	0.33 / 0.65	**0.23 / 0.49**	0.49 / 0.98
CGNS [55]	0.62 / 1.40	0.70 / 0.93	0.48 / 1.22	0.32 / 0.59	0.35 / 0.71	0.49 / 0.97
Social-BiGAT [54]	0.69 / 1.29	0.48 / 1.01	0.55 / 1.32	**0.30 / 0.62**	0.36 / 0.75	0.48 / 1.00
SR-LSTM [34]	0.63 / 1.25	0.37 / 0.74	0.51 / 1.10	0.41 / 0.90	0.32 / 0.70	0.45 / 0.94
Social STCGCNN [68]	0.64 / 1.11	0.49 / 0.85	**0.44 / 0.79**	0.34 / 0.53	0.30 / 0.48	0.44 / 0.75
STGAT [64]	0.65 / 1.12	0.35 / 1.12	0.52 / 1.10	0.34 / 0.69	0.29 / 0.60	**0.43 / 0.83**
My LSTM	0.581 / 1.168	0.259 / 0.503	0.578 / 1.214	0.463 / 1.022	0.346 / 0.748	0.446 / 0.936
My Enc-Dec	0.585 / 1.170	0.246 / 0.491	0.589 / 1.245	0.467 / 1.023	0.360 / 0.771	0.449 / 0.938
My Conv1D	0.560 / 1.190	0.246 / 0.427	0.590 / 1.249	0.478 / 1.046	0.346 / 0.737	0.444 / 0.926
My Conv2D	**0.559 / 1.114**	**0.240 / 0.464**	0.581 / 1.225	0.456 / 0.993	0.347 / 0.751	0.436 / 0.909

Table 4.9: Comparison with literature results on the ETH-UCY dataset. The results of models that have a reference have been taken directly from the publication. In bold the results with the best ADE.

In looking at Table 4.9 it is important to note that generative models are evaluated differently with respect to non-generative models. The evaluation method for generative models was introduced in Social-GAN [5] and consist in generating multiple possible future trajectories for any given input trajectory. Then when evaluating the ADE and the FDE of the model only the closest trajectory to the reality is considered. This is an important difference that makes results obtained using generative models difficult to reproduce and difficult to apply in a real-world scenario. This is the case because if the real future trajectory is not known the average of the multiple outputs needs to be used, worsening the performance of the model. The number of generated trajectories for every input trajectory is usually 20 in literature.
The networks developed in this thesis have a particularly low error on the ETH and Hotel scenes. This might be due to the fact that they are the scenes with less pedestrian density, and therefore social information is less important. Moreover, in the Hotel scene, the linear velocity model performs very well, with only 0.39 of ADE, indicating that a lot of the trajectories are straight. Models in literature struggle to obtain similar results to the linear velocity, however, thanks to the data augmentation techniques employed, the baselines and the proposed convolutional model are able to achieve a very low error.

This suggests that data augmentation is very helpful especially in cases of relatively straight trajectories. A more detail discussion on why non-social models (like the proposed convolutional model) are still able to obtain a very low error comparable to social models can be found in Section 5.6.
The importance of the coordinate types and the data augmentation techniques can be seen in the comparison of the results between the LSTM model trained by [5] and the LSTM baseline of this thesis. The first obtains an average ADE of 0.72, while the second obtains an average ADE of 0.446. The actual network is almost the same, but with a thoughtful pre-processing of the coordinates, effective data augmentation techniques and the right loss the difference in the results is massive.
It is possible to affirm that the 2D convolutional model achieves state-of-the-art performances on the ETH dataset, in the ETH and Hotel scenes, but this is not true for the UCY dataset. Nonetheless, the overall average error on the entire ETH-UCY dataset is still very low, with only the STGAT model and Social spatio-temporal graph convolutional neural network model reaching similar results. However, it is important to remember that these are generative models, which are tested differently. A more realistic comparison could be drawn with the SR-LSTM model, which is not generative, uses coordinates with the origin in the last observation point and random rotations. When compared to the SR-LSTM model, the convolutional model and the baselines perform better on the ETH dataset and a worse on the UCY dataset, but the difference in the results is less pronounced with respect to other models.

4.7.2 TrajNet dataset results comparison

The TrajNet dataset is less used in literature that the ETH-UCY dataset. This is very curious since it contains more pedestrians, more scenes, and it has a bigger training and test set. Moreover, it is in the same format of the ETH-UCY dataset, thus, from an implementation point of view, it does not require too much effort to adapt code written for the ETH-UCY dataset to work on the TrajNet dataset. The test is done online in a challenge-style format, so that various models can be compared in an unbiased way with reliable metrics. This approach is very popular in the deep learning community: it is at the base of platforms such as Kaggle [81] and it has also been used in literature, for example in challenges such as the Microsoft COCO[82].
For the reasons just stated, I would encourage researcher in the pedestrian trajectory prediction field to present their results also on the TrajNet dataset, to have a plain field in which it is possible to compare models on the same data

and with the same metrics. If the lack of ground truth in the test set is a problem in the development it is possible to use the ETH-UCY dataset to develop the models and find the optimal parameters, and then once the results are satisfactory train and then test the model on the TrajNet dataset. This is the approach used in this thesis.
Results obtained by the baselines, the convolutional models and the social models developed in this thesis, in comparison with the ones that can be found in literature and from the official challenge site [72] can be found in Table 4.10.

The models from literature present in Table 4.10 are:

- Social LSTM [4], results are taken from the TrajNet official site and from [83];
- Social GAN [5], results are taken directly from the TrajNet official site;
- Location-Velocity Attention [30], model that uses location and velocity in two different LSTM with an attention layer between the two cells, does not use social information and the results are taken directly from the paper;
- Social Forces model [2], with results taken from the official TrajNet site and from [26];
- SR-LSTM [34], model based on the state refinement of the LSTM cells of all the pedestrians in the scene to take into for account social interaction, the results are taken directly from the official TrajNet site;
- RED V2[26], a model consisting of a recurrent encoder with a dense multi-layer perceptron stacked on top the predictor, it is the best model reported in the literature and the second-best model on the official TrajNet site;
- Ind-TF, it is the best model on the official TrajNet site, but no information is available on this model since it does not have any publication or article connected to it.

The Social-LSTM and the Social-GAN models achieve a very high error. It is worth noting that that the results of the Social-GAN model are obtained using the mean of the 20 generated trajectories, since the ground truth was not available at test time.
Surprisingly, the Social Forces model performs extremely well on the TrajNet

	ADE	FDE
Social LSTM [4]	0.675	2.098
Social GAN [5]	0.561	2.107
Location-Velocity Attention [30]	0.438	1.449
Social Forces [2] (from [26])	0.371	1.266
SR-LSTM [34]	0.370	1.261
My Encoder-Decoder with Occupancy Grid	0.369	1.231
My Conv1D model	0.365	1.220
My Encoder-Decoder with Occupancy Radar	0.364	1.218
My Encoder-Decoder	0.362	1.220
My Conv2d Model with Occupancy Grid	0.360	1.215
RED V2 [26]	0.359	1.207
My Encoder-Decoder with Social Method 1	0.358	1.199
My Conv2D with Social Method 2	0.356	1.215
My LSTM	0.356	1.212
First in the competition (Ind-TF)	0.356	1.197
My Conv2D model	**0.352**	**1.192**

Table 4.10: Comparison of different models on the TrajNet dataset. In bold the model with the smallest ADE.

dataset. Meanwhile, the SR-LSTM model achieves very good results both in the ETH-UCY dataset and in the TrajNet dataset. Furthermore, it is worth noticing that the best model present in literature, RED V2, does not use social information, but it is still able to outperform all other methods that use it.
To remark the importance of the coordinate type, the loss used and the data augmentation techniques, the baseline LSTM network is able to achieve 0.356 ADE, outperforming all models from literature. However, the new best model is the 2D convolutional model of Figure 4.6.
It also is interesting to note that results obtained by the convolutional model with social information are worse than the same model with only trajectory information, but this is not true for the Encoder-Decoder model. The Encoder-Decoder model with social occupancy grid and with method 1 both outper-

form, even though marginally, the Encoder-Decoder baseline. Results obtained using the LSTM baseline and the 2D convolutional model will be published on the official TrajNet site after the publication of the thesis, so that the results have a source linked to them.

4.8 Additional data

Additional real-world data has been collected to validate the findings and investigate how the developed models could be applied in a real-word autonomous driving application.
The data has been collected using a camera and an automated tracking software, with the aid of 4 volunteers that acted as pedestrians. Their positions were retrieved every 0.4 seconds, and in total 15 scenes were recorded:

- In 5 of them pedestrians go straight at different velocities;
- In 6 of them pedestrians cross a crosswalk;
- In 2 of them pedestrians curve sharply;
- In the last two scenes pedestrians go straight, curve and cross a crosswalk.

In total 60 pedestrian trajectories have been collected, and from these, around 3200 samples can be used to train or test a network.
Since the new data has a very low concentration of pedestrians in the scenes, trajectory-only models are the most appropriate to use. The best performing models on the TrajNet dataset and on the ETH-UCY dataset have been tested on the newly acquired data, and results are reported on Table 4.11. For the models trained on the ETH-UCY dataset, the ones trained on all the scenes except the ETH scene are chosen, since that scene is the one with fewer pedestrians.

Model	Trained on	ADE	FDE
My LSTM	ETH-UCY dataset (except ETH scene)	1.299	2.463
My Conv2D	ETH-UCY dataset (except ETH scene)	1.255	2.367
My LSTM	TrajNet dataset	1.211	2.300
My Conv2D	TrajNet dataset	**1.188**	**2.259**

Table 4.11: Comparison of different models tested on the newly acquired data.

From the results of Table 4.11 it is possible to confirm what previously found: the LSTM model is inferior to the Conv2D model, which performs better independently of the training set.
A second observation that can be done is on the training dataset: it is clear that the TrajNet dataset constitutes a better training set since both models trained on it outperform their counterparts trained on the ETH-UCY dataset. This is because in the TrajNet dataset there are more pedestrians and more heterogeneous trajectories. This means that a model trained on it will have greater generalization capabilities, and therefore will achieve a lower error when tested on new data, when compared to the same model trained on a smaller and less heterogeneous dataset, like the ETH-UCY dataset.
A third observation is that the test error is higher than the one obtained on the other datasets. The 2D convolutional model achieves 0.352 ADE on the TrajNet dataset and 0.436 ADE on the ETH-UCY dataset, but on the newly acquired data the ADE is 1.188. This observation is true also for the LSTM model. There are two main reasons behind it.
The first is that the collected scenes preset a lot of the failure scenarios explained in Section 5.5. In the collected data there are a lot of pedestrians curving and stopping because of environmental factors (like a road turning or a crosswalk), and in that cases, a model that does not take spatial information into account struggles to create accurate predictions in the moments before the turning point. For more detailed explanation on this topic see Section 5.5 and Figure 5.2.
The second reason behind is exposed in Section 1.7: a prediction model is at best as good as the detection and tracking system. The data was taken directly from the tracking system of the camera, without any trajectory smoothing or noise cancellation. Instead, in datasets such as the ETH-UCY and TrajNet, manual review of each pedestrian trajectory, correction of wrong annotations and smoothing of each trajectory was performed, to make the data more accurate and "clean". Thus, models trained on clean data are susceptible to every imprecision of the tracking systems, leading to a high error. Therefore it is possible to conclude that, in a concrete application setting, real-time data preprocessing from the detection and tracking system could be as impactful on the quality of the predictions as the network itself.
An example of inaccurate predictions caused by imprecisions in the tracking system can be seen in Figure 4.10.

As a final note on the newly collected data, Table 4.12 shows the results obtained by models trained on the new data and then tested on the TrajNet dataset and on the ETH-UCY dataset. The error is, of course, bigger than

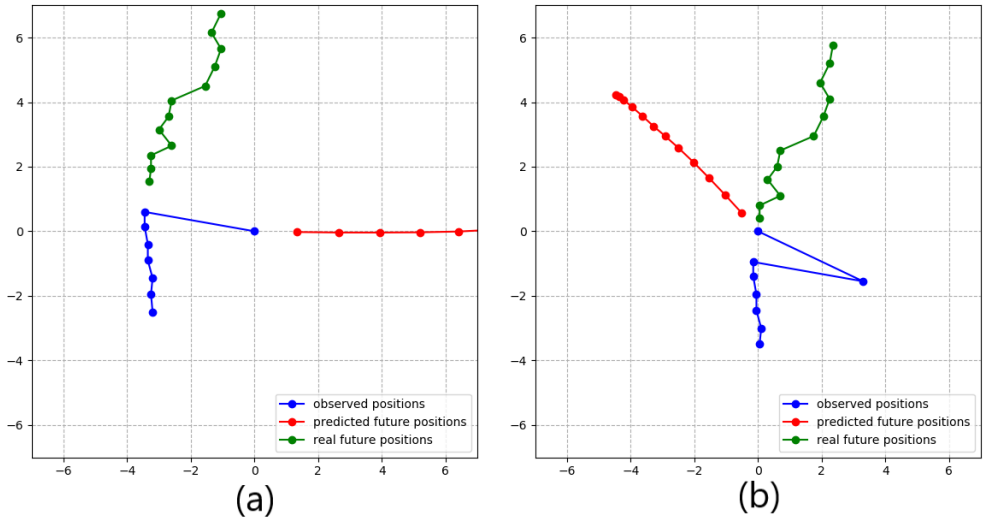


Figure 4.10: Inaccurate predictions caused by tracking system imprecisions. (a) The last observation point is more than three meters on the right with respect to its (presumably) correct location, therefore the model wrongly predicts that the pedestrian will turn very fast on the right. (b) One frame after, the detection system correctly detects the pedestrian position, but the past trajectory retains the imprecision and therefore the model wrongly predicts that the person will turn left.

the one obtained using the correct training set for the datasets, but it is still impressive that models trained on only 3200 samples are able to reach such a small error. This could be attributed to the effectiveness of data augmentation techniques.

Model	Tested on	ADE	FDE
My LSTM	ETH-UCY dataset	0.488	0.993
My Conv 2D	ETH-UCY dataset	0.477	0.967
My LSTM	TrajNet dataset	0.398	1.321
My Conv2D	TrajNet dataset	0.389	1.295

Table 4.12: Comparison of different models trained on the newly acquired data and tested on other datasets.

Chapter 5

Discussion

The quantitative results presented in the last chapter can be analyzed to gain insights on the reasons why some approaches work better than others. In Section 5.1 the impact of different coordinates types is analyzed and in Section 5.2 the effects of data augmentation are discussed.
The newly proposed convolutional model presents different characteristics when compared to recurrent models, as analyzed in Section 5.3, and it also presents new challenges regarding its parameter tuning, as discussed in Section 5.4.
Qualitative data can also be used to understand when models fail to provide accurate predictions, and this aspect is addressed in Section 5.5. Finally, in Section 5.6, the addition of social information is discussed, to understand when it might be useful and when it might be detrimental.

5.1 Effects of different coordinates types

In Table 4.3 there are the results obtained with the baseline LSTM using different coordinates types, the correct ETH scene and the ADE loss. From the reported numbers it is clear that absolute coordinates are the worst performing one, especially in the ETH and Hotel scenes. It is also not too difficult to understand why: if a network is trained using coordinates with one reference system and in a certain range (for example, all training coordinates are between 10 and 20) when tested on coordinates using a different reference system and with a different range (for example, all test coordinates are all between -20 and -10) the network will probably fail to produce meaningful results on the test set because it is too different from the training set. This drawback is circumvented by using coordinates with a fixed reference frame, or relative coordinates, and that is why these types of coordinates perform better.

Reexamining results of Table 4.3, it is also immediate to understand that the coordinates type that performs best is the one with the origin in the last observation point. What it is not immediate is to understand why it is the case.
The reason is the order. If the reference system is put in the last observation point, when the network sees a position far away from the origin, like (-5,-5), it understands that is one of the oldest positions and it is less important. Meanwhile, if the network sees a position like (-1,-1), it knows is a recent position and therefore is more important. Therefore putting the origin in the last observation point gives an implicit order to the coordinates that the network can learn.
However, what just explained is also true if the origin is in the first observation point, with the difference that this time smaller coordinates are older and bigger coordinates are more recent. Therefore the order explanation justify why it is good to put the origin in the first or last observed point, and not why is better to put it in the last point.
The explanation of why putting the origin at t = T_{obs-1} is better is straightforward: the last observation point is the most important one because is the most recent. Therefore, if the origin is placed in that point all the trajectory is seen through the lens of the most important point. However, if the origin is put in the first observation point, all the trajectory is seen through the lens of the least important point, making more difficult for the network to understand the trajectory.
Results obtained in the study of coordinates types are in line with the ones presented in SR-LSTM [34], in which the authors also claim that the use of coordinates with the origin in the last observation point gives better results when compared to relative coordinates. Moreover, the fact that the network can easily understand the order of positions if the origin is placed in the last observation point is supported by results of Table 4.6, in which the convolutional model with positional embedding performs worse than the one without it. There is no need for positional embedding since thanks to a well-thought coordinate manipulation the networks already easily understand the order of the coordinates, thus adding positional embedding confuses the network.
Regarding relative coordinates, they are basically the same as coordinates that have the origin in the first observation point. If every relative displacement is summed to all of the previous ones the result is the same as having coordinates in the first observation point. This is the reason why in Table 4.3 these two types of coordinates produce very similar results.
In essence, putting the origin in the last observation point gives an order to the coordinates, and all the trajectory is seen through the point of view of the

most important point, the most recent one, and therefore the network can more easily understand the important features of the observed trajectory.

5.2 Effects of data augmentation

Data augmentation was one of the key techniques that contributed to the results obtained in this thesis, as can be seen in Table 4.4. As previously mentioned, the ETH-UCY dataset contains only 34000 samples that can be fed to the network, and in the case of TrajNet, this number falls to 11000. Therefore it is not surprising that data augmentation techniques are so effective: the data points available are very limited. Thus, artificially creating new ones that share the same characteristics as real data points can drastically improve the generalization capabilities of the trained model, and in turn, improve performances on the test set.
The two most important data augmentation techniques used are the addition of Gaussian noise to every point and the application random rotation of the trajectory.
The benefits of applying random rotations are clear: if in the training set there are no trajectories that go from left to right but that type of trajectory is present in the test set, it is difficult for any model to accurately predict trajectories that are too different from the ones in the training set. However, applying random rotations artificially enlarges the training set to account for trajectories in every direction. Benefits of random rotations can be understood confronting Table 4.3 with Table 4.4, in which the LSTM baseline goes from 0.535 to 0.463 of average ADE when trained with random rotations. The biggest improvement is in the Hotel scene, which is also the one that has more people going straight (the linear model has a very low ADE, 0.39, as can be seen in Table 4.9). Therefore it is possible to affirm that random rotations improve performances especially in the case of more linear trajectories. In SR-LSTM [34] the authors also noted that the use of random rotations could improve results, but in their case, the error was reduced from 0.55 to 0.53, while in the case reported in this thesis the average ADE reduction is bigger, from 0.535 to 0.463 of average ADE.
The addition of Gaussian noise with mean 0 to every point also improves performances, but in a smaller way if compared to random rotations. The standard deviation of the noise is set to 0.05 in the reported experiments because it was found that smaller values do not have any impact, and bigger values decrease performances. The addition of Gaussian noise makes the network predictions more robust to small insignificant changes and improve the perfor-

mances mainly on the ETH and Hotel scenes.
The introduction of mirroring on top of random rotations does not improve the results, probably because what the network could learn with mirroring it is already learned with random rotations. Mirroring a trajectory makes too small of a difference to be impactful when the trajectory is already rotated.
When random rotation and noise are combined there are big improvements in the ETH and Hotel scenes, while performances on the Univ, Zara1 and Zara2 slightly decreases. This shows that the effectiveness of data augmentation techniques depends on the data and that the decision of which one to use has to be treated as a hyperparameter. Nevertheless, the use of data augmentation techniques such as Gaussian noise and random rotations improves the performance significantly, from 0.535 to 0.446 of average ADE in the LSTM baseline.

5.3 Convolutional model and recurrent models comparison

One of the main contributions of this thesis is, together with the study on coordinates types and data augmentation, the introduction of a convolutional model for pedestrian trajectory prediction. In the development of the two different types of models presented, recurrent and convolutional, three main difference emerged.
The first is computation time. As can be seen in Table 5.1, the convolutional model is more than three times faster than the Encoder-Decoder baseline and more than four times faster than the LSTM baseline at test time. These results are also valid during training time. Reduced computational time means that models can be developed faster and that on systems with constrained resources a convolutional model is preferable to a recurrent model, even if the convolutional model might be less accurate. However, as Table 4.9 and Table 4.10 show, there is no reason to prefer the two developed recurrent models to the convolutional model since the latter produces better results and it is also more computationally efficient.

The second difference between the recurrent models and the convolutional model is the number of hyperparameters. The LSTM and Encoder-Decoder models have a very small number of hyperparameters (embedding size, hidden state length and the fully connected layers after the LSTM cell). Meanwhile, the convolutional model has a bigger number of hyperparameters (embedding size, number of layers, number of channels for each layer and kernel size for

	batch size = 1	batch size = 32
Convolutional2D model (155k parameters)	**0.0033s per element**	**0.00017s per element**
LSTM baseline (106k parameters)	0.0207s per element	0.00064s per element
Encoder-Dedocer baseline (208k parameters)	0.0118s per element	0.00043s per element

Table 5.1: Comparison of the computational test time of different models on an Nvidia Quadro 1000.

each layer). Therefore, the convolutional model requires more hyperparameter tuning than the recurrent models.
The third difference between the two types of models is flexibility. The recurrent model can be trained to observe, for example, 6 positions and predict the next 16 without any change in the architecture. This is not true in the case of the convolutional model, in which some adjustments need to be done, probably revolving around the upsampling layer and the convolutional layers without padding. This, in turn, means that hyperparameter tuning has to be performed again to extract the most out of the architecture. The hyperparameter tuning task is alleviated by the fact that training is faster, but less flexibility is still a relevant downside of convolutional models when compared to recurrent models.
This thesis proves that a convolutional model for pedestrian trajectory prediction can outperform recurrent models and that it can do it using less computational time. However, more effort needs to be put into the convolutional model architecture to get the most out of it.

5.4 Convolutional model hyperparameters

The results of the convolutional model, in Table 4.6, can give some insights on how various hyperparameters choices affect the model's performances.
The first critical hyperparameter is the kernel size. From the experiments, a pattern emerges: a bigger the kernel size leads to better results. This is because a bigger kernel size is able to see more features of the trajectory at once and thus the model can generate more refined predictions. The idea behind why a bigger kernel size is better is the following: the more information a kernel can see the better can interpret complex behaviours in the trajectory. This idea still applies when the 1D convolution model is confronted with the 2D convo-

lution model. In the first, the kernel looks at the same feature (computed by the embedding layer) on multiple time steps (a kernel size 3 sees 3 time steps, a kernel size 5 sees 5 time steps and so on). The 2D convolutional model, instead, looks at multiple features in multiple time steps and therefore is able to see more information with one kernel and in turn to generate better predictions. However, the rule of seeing more has diminishing returns: experiments with the 2D convolutional model using kernel size 7 generated slightly worst results compared to the same 2D model with kernel size 5. From the study done, it is possible to conclude that a 2D kernel with kernel size 5 processes the optimal amount of information to best predict the future trajectory.
Another important parameter in the convolutional model is the numbers of layers. Pedestrian trajectory prediction does not have a sizable and complex input such as image classification. Therefore, from experiments, it has been observed that increasing the number of layers does not improve results, even if residual connections are applied. In fact, it has been observed that there is no benefit in applying residual connections when layers are less than 15, and at that point, results are very poor, both with and without residual connections.
Regarding the other convolutional model variations, using residual connections is ineffective, as explained in 5.1, and the upsampling layer proved to generate better results when compared to the use of transpose convolutions layers.

5.5 Failure cases

Until now metrics such as the ADE and the FDE have been used to analyze the results of various models, however, these metrics alone can be limiting in understanding the quality of the predictions. A better grasp on this aspect can be reached looking at the distribution of the average displacement error over all the predictions, in Figure 5.1. There, it is possible to see that the prediction error distribution resembles a Gaussian curve with a long tail. The situations depicted in the long tail, which are situations that result in very high errors, are either sharp turns, pedestrians stopping or still pedestrians that start walking again.

In the first case, with sharp turns, the typical scenario is the following: a person is going straight and then does a quick 90-degree turn because the road was either turning or forking. An example of such behaviour can be seen in Figure 5.2. In scenarios like that it is reasonable to assume that only models including spatial information can predict the turn, but if the road is forking even that kind of models could have some difficulties. What models that do

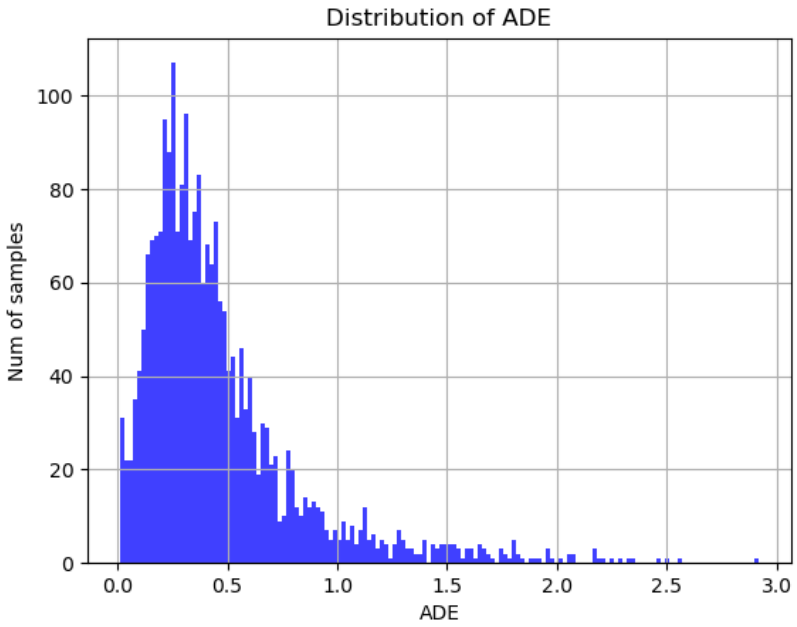


Figure 5.1: Distribution of the ADE of the 2D convolutional network from Figure 4.6 on the Zara1 scene. The average is 0.456, with a variance of 0.137 and a standard deviation of 0.370. The biggest value is 2.92.

not include social information can learn is to adapt quickly to sharp changes in trajectory, as shown in Figure 5.2.

In the second and third scenario, the ones of pedestrians either stopping or walking again after stopping, it can be difficult to understand the reasons for this kind of behaviour. People could stop to look at some shops windows, to greet some friends, or to simply wait for someone else. Instead, when people start walking after being still it is usually after the previously described behaviours: people resume walking after seeing shop windows of after a chat with some friends. Predicting if a person will look at that particular shop window or not, or exactly predicting when two persons that have stopped for a chat will resume walking, it is very difficult, especially if the network only uses coordinate information. Some of these behaviours can be seen in Figure 5.3. In these two cases what a model can learn is the same as what it can learn in the case of turns: to quickly adapt to changes in the trajectory.

It is also important to note that these behaviours that cause the model to fail are not social behaviours. A person turning because the street is turning does not involve any social interaction, and the same is true for people stopping to

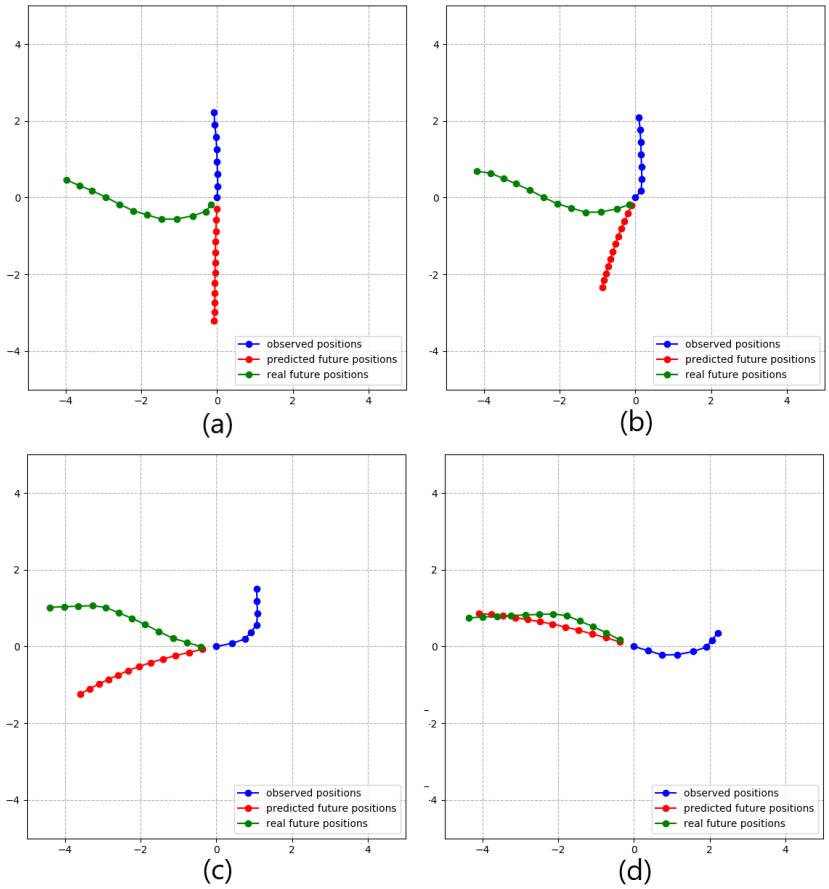


Figure 5.2: (a) The worst prediction of the 2D convolutional network on the Zara1 scene: a person is going down and then changes direction very sharply because the road has a turn. The prediction is wrong because in the observed positions there are no clues of a turn. (b) The prediction on the same person one frame (0.4 seconds) after: with only one position pointing in a different direction the network is able to understand that that person is turning. (c) The trajectory after other three frames: the prediction aligns even more with reality. (d) After other three frames the prediction is very similar to the ground truth.

see shop windows or to start walking again after that. When groups of friends stop to talk or resume walking after chatting, they do it as a group. Therefore, members of the group stop or resume walking at the same time. Thus, they can be considered as a single pedestrian which has the behaviour of stopping or to resume walking, behaviours really hard to predict. Consequently, models relying on only trajectory information and models relying on both trajectory and social information should have the same failure cases.

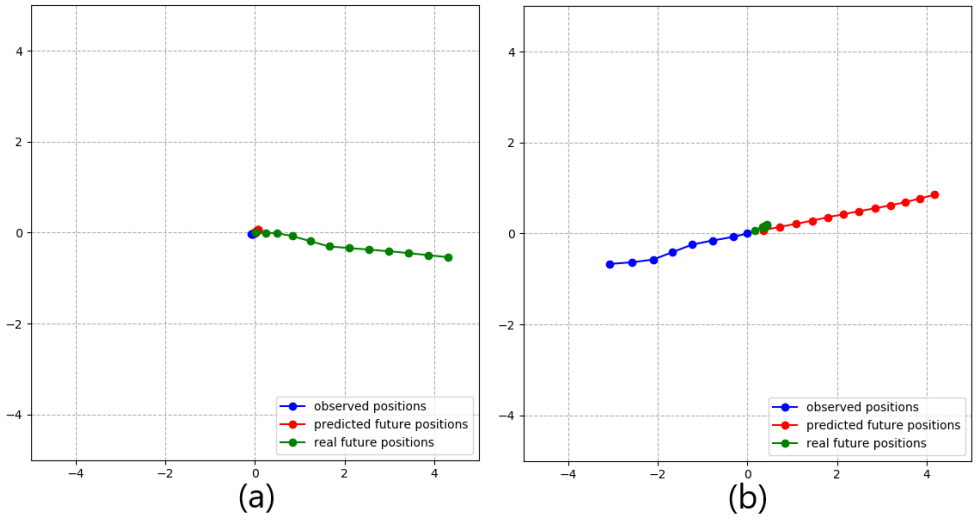


Figure 5.3: Other predictions of the 2D convolutional network. (a) A pedestrian that was stationary (all observed positions overlap in the origin) starts moving again (b) A pedestrian that was walking stops. In both cases the observed positions give no clue on the future behaviour.

5.6 Effects of social information

Since the first publications about pedestrian trajectory prediction with deep learning, great emphasis has been put on the addition of social information. The addition of social information was, in fact, one of the main contributions of the first paper in trajectory pedestrian prediction with deep learning, Social-LSTM [4]. Therefore, it is curious that the proposed 2D convolutional model, which does not use social information, is capable of outperforming the majority of the methods that use social information on the ETH-UCY dataset. However, this is not exactly the case. In fact, looking at results in Table 4.9, it is possible to note that the proposed models that do not use social information are outperformed in the scenes Univ, Zara1 and Zara2 from the UCY dataset, which are the scenes with more pedestrian density. At the same time, the proposed models are outperforming the models from literature that use social information on the ETH and Hotel scenes, from the ETH dataset. This could mean that deep learning architectures present a trade-off: in cases in which social information is useful models that use social information perform better than the ones without it, while in cases in which social information is not useful they perform worse since useless information confuses the network. Therefore, when developing pedestrian trajectory models it is useful to under-

stand when social information is actually useful. As discussed in 5.5, a lot of the failure cases, such as people unexpectedly turning, stopping, or resuming to walk, are not correlated to the social aspect, therefore social information does not help the network generate better predictions in that cases. The main scenarios in which social information is useful is when a person avoids another person turning or that is going towards him/her.
Analyzing the Zara2 scene as an example, over its 7 minutes of length, around 30 of such cases have been identified, involving around half of the 189 total pedestrians in the scene. Similar numbers can be obtained analyzing the Zara1 scene. Therefore, a model that does not use social information would obtain on half of the pedestrians the same, or better, results than the ones obtained by a model that does use social information. This proportion is even bigger, since the trajectories of pedestrians affected by social interaction are not affected in their entirety, but only in part of their overall trajectory. This might explain why in the results of Table 4.9 the best social models from literature achieve around 30% less error in the Zara1, Zara2 and Univ scenes with respect to the proposed models that do not use social information. That 30% might be the error that it is possible to reduce only considering social interaction. The remaining error that none of the models can eliminate might be due to spatial interactions and unpredictable behaviours, such as stopping or resuming walking.
Analyzing the results and the data it is possible to conclude that social models perform better in scenes where social interaction is present, but they actually perform worse in scenes where social interaction is not present. In that scenes, models that use only trajectory information should be used since they can obtain better results because they are optimized to use only trajectory information and not use useless, and possibly misleading, social information.
Results on the TrajNet dataset are consistent with the ones obtained on the ETH-UCY dataset. The TrajNet dataset is a collection of datasets, but the majority of the scenes and the pedestrians are actually from the Stanford Drone Dataset (SDD). In that dataset, social interaction is rarely present since the recordings where taken in uncrowded places. There are multiple pedestrians in each frame of the scenes, but they are far apart from each other and do not interact. This means that models that do not use social information, such as RED v2, the LSTM baseline and the Convolutional model, perform very well, while social models such as Social LSTM, Social GAN and SR-LSTM, obtain inferior results.
In essence, models with social information should be used in scenes such as Univ, Zara1 and Zara2 from the UCY dataset because in these scenes social

interaction is present and therefore social information permits the network to make better predictions. In other scenes such as ETH and Hotel from the ETH dataset and the whole TrajNet dataset as a whole, social interaction is very rare and hence social models are confused by useless information. Therefore, in those scenes models that only use trajectory should be used since they are optimized for those cases. This explains why the proposed convolutional model performs really well on the ETH and TrajNet dataset but it is outperformed by other models from literature in the UCY dataset.

5.6.1 Inclusion of social information failure

In Section 4.6 models that use social information still performed poorly on the UCY dataset, contrary to the ones in the literature. This might be because the proposed techniques to model social information are too simple. In Table 4.9 architectures that perform well on the UCY dataset use more complex systems to model social interaction, such as state refinement or spatio-temporal graph neural networks.
The proposed models that use social information behave almost like the models without it, because they struggle to find connections between the social information in input and the target trajectory. This can be seen in Figure 5.4, in which the gradient flow of the 2D convolutional model with occupancy grid information is presented. Therefore, it is possible to conclude that social information is useful in the UCY dataset because models from the literature are able to use it to achieve better results than the proposed methods that use only trajectory information. However, the techniques proposed in Section 4.6 to represent social information, even though they are based on literature, are not effective in representing social interaction.

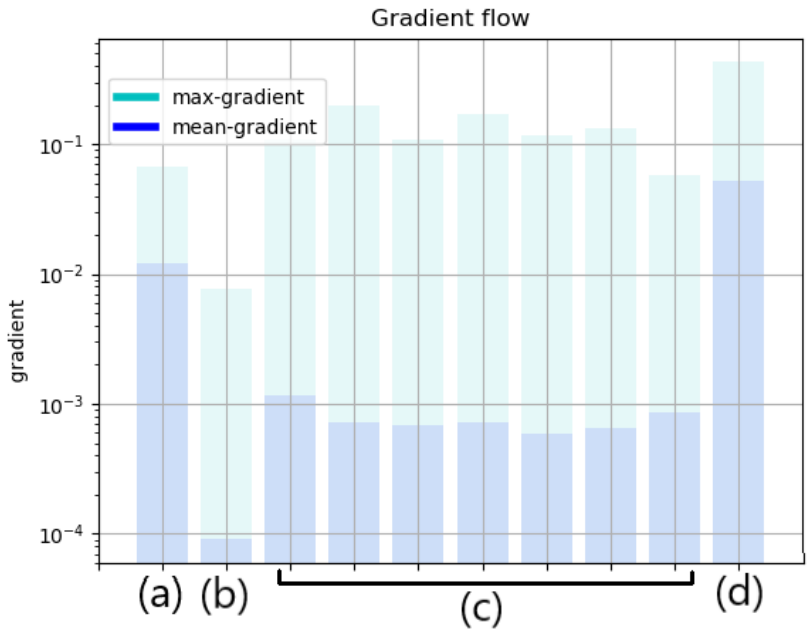


Figure 5.4: Example of gradient flow in the 2D convolutional model that uses an occupancy grid to represent social information. The gradient flow in the social occupancy grid embedding layer is 100 times less than the gradient flow in the position embedding layer. This means that the network does not value social information nearly as much as position information. (a) Position embedding layer (b) Social occupancy grid embedding layer (c) Convolutional layers (d) Final fully connected layer.

Chapter 6

Conclusion

Pedestrian trajectory prediction is one of the many fields in which the advent of deep learning completely changed the approach to the problem. Before 2016 physics-based models were the only way to accurately predict future trajectories of pedestrian, but in 2016 it was shown that deep learning could do it better. Since then, various new architectures have been proposed with increasingly better results.

Data-driven models such as deep learning models are very sensitive to the quantity and the quality of the training data and how this data is then presented to the model. That is the reason why the first contribution of this thesis is a study on the coordinate types and on the data augmentation. After comparing a simple LSTM trained with absolute coordinates, coordinates with the origin in the first observation point, coordinates with the origin in the last observation point and relative coordinates, Table 4.3 shows that coordinates with the origin in the last observation point are the ones that obtain the best results. This is because that type of coordinate has a structure that makes the network able to understand the order of the coordinates, and because the last observation point is the most important one, since it is the most recent one.

It is also possible to affirm that it is better to train a network with the Average Displacement Error(ADE) as loss instead of using the Mean Square Error as loss, as can be noticed comparing Table 4.2 and Table 4.3.

Since input samples on the ETH-UCY dataset and in the TrajNet dataset are limited in number, various data augmentation techniques have been tested. It was found that the additions of Gaussian noise with mean 0 to every point and random rotations can dramatically improve the results of a simple LSTM, especially in the ETH and Hotel scenes of the ETH-UCY dataset. Results can be found in Table 4.4.

The baseline LSTM is able to reach 0.446 as average ADE on the ETH-UCY dataset (placing the base LSTM model as one of the best models in ETH-UCY dataset) and 0.356 on the TrajNet dataset (outperforming all models from literature in the TrajNet dataset). This result is obtained thanks to the use of coordinates with the origin in the last observation point, the use of the ADE as loss and the use of Gaussian noise and random rotations as data augmentation techniques. Thus, this result shows how important is the choice of coordinates, loss and data augmentation techniques.
As a second contribution, this thesis presents a novel convolutional model able to outperform the already excellent result of the LSTM baseline. This demonstrates that convolutional neural networks can, in the field of pedestrian trajectory prediction, obtain better results than recurrent models with less computational load. The proposed convolutional model fist embeds the eight pairs of input coordinates to form an 8x64 matrix that can be viewed as a one-channel image. Then various groups of 2D convolutions with and without padding are applied to the image. At the end, a final fully connected layer transforms the row of the matrix into output coordinates. The full architecture can be viewed in Figure 4.6. As Table 4.6 shows, a bigger kernel size improves the results, and the 2D convolutional model outperforms the 1D convolutional model.
The final 2D convolutional model achieves an average ADE of 0.436 on the ETH-UCY dataset (placing it as one of the best models on that dataset), and it obtains an ADE of 0.352 on the TrajNet dataset: the best result ever achieved on that dataset.
The final contribution of this thesis is a study on the effectiveness of different techniques to include social information. The techniques presented in Figure 4.7 and in Section 4.6.2 did not achieve better results than the Encoder-Decoder baseline and the 2D convolutional model. Therefore, it is possible to conclude that the implemented techniques are not adequate to model social information and that other more complex techniques might be better, such as the use of spatial-temporal neural networks or state refinement.
However, from a comparison of the developed models without social information and social models from literature (in Table 4.9 and Table 4.10), it is possible to conclude that social models perform better only on the UCY dataset (in the scenes Univ, Zara1 and Zara2), in which there is more social interaction. Meanwhile, trajectory-only models outperform social models on datasets where social interaction is more sparse, such as the ETH dataset (in the scenes ETH and Hotel) and on the TrajNet dataset.

Bibliography

[1] Enrico Brivio and Stephan Meder. “2018 Road Safety Statistics: What Is behind the Figures?” In: *Europena Commission press* (2018).

[2] Dirk Helbing and Péter Molnár. “Social Force Model for Pedestrian Dynamics”. In: *Physical Review E* 51.5 (May 1995), pp. 4282–4286. ISSN: 1095-3787. DOI: 10.1103/physreve.51.4282.

[3] Sujeong Kim et al. “BRVO: Predicting Pedestrian Trajectories Using Velocity-Space Reasoning”. In: *The International Journal of Robotics Research* 34.2 (2015), pp. 201–217. DOI: 10.1177/0278364914555543.

[4] Alexandre Alahi et al. “Social LSTM: Human Trajectory Prediction in Crowded Spaces”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[5] Agrim Gupta et al. “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 2255–2264.

[6] Anirudh Vemula, Katharina Muelling, and Jean Oh. “Social Attention: Modeling Attention in Human Crowds”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2017), pp. 1–7.

[7] Tharindu Fernando et al. “Soft + Hardwired Attention: An LSTM Framework for Human Trajectory Prediction and Abnormal Event Detection”. In: *Neural networks : the official journal of the International Neural Network Society* 108 (2017), pp. 466–478.

[8] Mark Pfeiffer et al. “A Data-Driven Model for Interaction-Aware Pedestrian Motion Prediction in Object Cluttered Environments”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018).

[9] Amir Sadeghian et al. “SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints”. In: CVPR2019, June 2019, pp. 1349–1358. DOI: 10.1109/CVPR.2019.00144.

[10] Nishant Nikhil and Brendan Tran Morris. “Convolutional Neural Networks for Trajectory Prediction”. In: *arXiv* abs/1809.00696 (2018).

[11] Jonas Gehring et al. “Convolutional Sequence to Sequence Learning”. In: *ArXiv* abs/1705.03122 (2017).

[12] J. Aneja, A. Deshpande, and A. G. Schwing. “Convolutional Image Captioning”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5561–5570.

[13] S. Pellegrini et al. “You’ll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking”. In: *2009 IEEE 12th International Conference on Computer Vision*. Sept. 2009, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

[14] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. “Crowds by Example”. In: *Computer Graphics Forum* 26 (2007), pp. 655–664.

[15] Amir Sadeghian et al. “TrajNet: Towards a Benchmark for Human Trajectory Prediction”. In: *arXiv preprint* (2018).

[16] Christoph Schöller et al. “The Simpler the Better: Constant Velocity for Pedestrian Motion Prediction”. In: *arXiv* (Mar. 2019).

[17] *Social Forces Blog Post*. http://futurict.blogspot.com/2014/12/social-forces-revealing-causes-of.html.

[18] Jur van den Berg et al. “Reciprocal N-Body Collision Avoidance”. In: *Robotics Research*. Springer Berlin Heidelberg, 2011, pp. 3–19. ISBN: 978-3-642-19457-3.

[19] Javier Alonso-Mora et al. “Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots”. In: *Distributed Autonomous Robotic Systems*. Vol. 83. Jan. 2013, pp. 203–216. DOI: 10.1007/978-3-642-32723-0_15.

[20] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural computation* 9 (1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[21] Paul. J. Werbos. “Backpropagation through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. ISSN: 1558-2256. DOI: 10.1109/5.58337.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386.

[23] Zachary Chase Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *ArXiv* (2015).

[24] Kyunghyun Cho et al. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv* abs/1406.1078 (2014).

[25] Y. Li. "A Deep Spatiotemporal Perspective for Understanding Crowd Behavior". In: *IEEE Transactions on Multimedia* 20.12 (Dec. 2018), pp. 3289–3297. ISSN: 1941-0077. DOI: 10.1109/TMM.2018.2834873.

[26] Stefan Becker et al. "An Evaluation of Trajectory Prediction Approaches and Notes on the TrajNet Benchmark". In: *ArXiv* abs/1805.07663 (2018).

[27] H. Xue, D. Q. Huynh, and M. Reynolds. "SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Mar. 2018, pp. 1186–1194. DOI: 10.1109/WACV.2018.00135.

[28] Hao Xue, Du Huynh, and Mark Reynolds. "Pedestrian Trajectory Prediction Using a Social Pyramid". In: *PRICAI 2019: Trends in Artificial Intelligence*. Ed. by Abhaya C. Nayak and Alok Sharma. Vol. 11671. Springer International Publishing, 2019, pp. 439–453. ISBN: 978-3-030-29910-1.

[29] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *ArXiv* 1409 (Sept. 2014).

[30] H. Xue, D. Huynh, and M. Reynolds. "Location-Velocity Attention for Pedestrian Trajectory Prediction". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2019, pp. 2038–2047. DOI: 10.1109/WACV.2019.00221.

[31] Y. Xu, Z. Piao, and S. Gao. "Encoding Crowd Interaction with Deep Neural Network for Pedestrian Trajectory Prediction". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 5275–5284. DOI: 10.1109/CVPR.2018.00553.

[32] Tharindu Warnakulasuriya et al. "GD-GAN: Generative Adversarial Networks for Trajectory Prediction and Group Detection in Crowds". In: *Computer Vision - ACCV 2018: 14th Asian Conference on Computer Vision*. Ed. by H Li et al. Springer, 2019, pp. 314–330. DOI: 10.1007/978-3-030-20887-5_20.

[33] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. "Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories with GANs". In: *CVPR19 Workshops*. 2019.

[34] Pu Zhang et al. "SR-LSTM: State Refinement for LSTM towards Pedestrian Trajectory Prediction". In: *CVPR2019* (Mar. 2019).

[35] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS14. MIT Press, 2014, pp. 2672–2680.

[36] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *arXiv* abs/1411.1784 (2014).

[37] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS16. Curran Associates Inc., 2016, pp. 2180–2188. ISBN: 978-1-5108-3881-9.

[38] Zhao Pei et al. "Human Trajectory Prediction in Crowded Scene Using Social-Affinity Long Short-Term Memory". In: *Pattern Recognition* 93 (2019), pp. 273–282.

[39] Hao Cheng and Monika Sester. "Mixed Traffic Trajectory Prediction Using LSTM-Based Models in Shared Space". In: *AGILE Conf.* 2018.

[40] Xiaodan Shi et al. "Pedestrian Trajectory Prediction in Extremely Crowded Scenarios". In: *Sensors* 19 (Mar. 2019), p. 1223. DOI: 10.3390/s19051223.

[41] Fangkai Yang, Himangshu Saikia, and Christopher Peters. "Who Are My Neighbors?: A Perception Model for Selecting Neighbors of Pedestrians in Crowds". In: *ACM* (Nov. 2018), pp. 269–274. DOI: 10.1145/3267851.3267875.

[42] Kaiping Xu et al. "Collision-Free LSTM for Human Trajectory Prediction". In: *MMM*. 2018.

[43] Shuai Yi, Hongsheng Li, and Xiaogang Wang. "Pedestrian Behavior Understanding and Prediction with Deep Neural Networks". In: *EECV 16*. Vol. 9905. EECV 16, Oct. 2016, pp. 263–279. ISBN: 978-3-319-46447-3. DOI: 10.1007/978-3-319-46448-0_16.

[44] Niccoló Bisagno, Bo Zhang, and Nicola Conci. "Group LSTM: Group Trajectory Prediction in Crowded Scenarios". In: *ECCV 2018 Workshop*. Jan. 2019, pp. 213–225. ISBN: 978-3-030-11014-7. DOI: 10.1007/978-3-030-11015-4_18.

[45] Yanliang Zhu et al. "StarNet: Pedestrian Trajectory Prediction Using Deep Neural Network in Star Topology". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 8075–8080.

[46] Federico Bartoli et al. "Context-Aware Trajectory Prediction". In: *2018 24th International Conference on Pattern Recognition (ICPR)* (2017), pp. 1941–1946.

[47] Hiroaki Minoura et al. "Path Predictions Using Object Attributes and Semantic Environment". In: *14th International Conference on Computer Vision Theory and Applications*. 14th International Conference on Computer Vision Theory and Applications, Jan. 2019, pp. 19–26. DOI: 10.5220/0007297500190026.

[48] Matteo Lisotto, Pasquale Coscia, and Lamberto Ballan. "Social and Scene-Aware Trajectory Prediction in Crowded Spaces". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019.

[49] Alexandre Robicquet et al. "Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes". In: *ECCV 2016* 9912 (2016), pp. 549–565. DOI: 10.1007/978-3-319-46484-8_33.

[50] B. Zhou, X. Wang, and X. Tang. "Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 2871–2878. DOI: 10.1109/CVPR.2012.6248013.

[51] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).

[52] A. Syed and B. T. Morris. "SSeg-LSTM: Semantic Scene Segmentation for Trajectory Prediction". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. June 2019, pp. 2504–2509. DOI: 10.1109/IVS.2019.8813801.

[53] Hao Cheng et al. "MCENET: Multi-Context Encoder Network for Homogeneous Agent Trajectory Prediction in Mixed Traffic." In: *arXiv* abs/2002.05966 (2020).

[54] Vineet Kosaraju et al. “Social-Bigat: Multimodal Trajectory Forecasting Using Bicycle-Gan and Graph Attention Networks”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 137–146.

[55] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. “Conditional Generative Neural System for Probabilistic Trajectory Prediction”. In: *ArXiv* abs/1905.01631 (2019).

[56] Xingjian SHI et al. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 802–810.

[57] Kai Chen, Xiao Song, and Hang Yu. “Conv-Lstm: Pedestrian Trajectory Prediction in Crowded Scenarios”. In: *AsiaSim*. 2019.

[58] Wenpeng Yin et al. “Comparative Study of CNN and RNN for Natural Language Processing”. In: *arXiv* abs/1702.01923 (2017).

[59] Namhoon Lee et al. “DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents”. In: *CVPR17*. July 2017, pp. 2165–2174. DOI: 10.1109/CVPR.2017.233.

[60] Atanas Poibrenski et al. “M2P3: Multimodal Multi-Pedestrian Path Prediction by Self-Driving Cars with Egocentric Vision”. In: *Proceedings of 35th ACM Symposium on Applied Computing. ACM Symposium on Applied Computing (SAC-2020)*. ACM Press, 2020.

[61] J. Mänttäri and J. Folkesson. “Incorporating Uncertainty in Predicting Vehicle Maneuvers at Intersections with Complex Interactions”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. June 2019, pp. 2156–2163. DOI: 10.1109/IVS.2019.8814159.

[62] Allan Wang, Zirui Wang, and Wentao Yuan. “Pedestrian Trajectory Prediction with Graph Neural Networks”. In: (2019).

[63] Sirin Haddad et al. “Situation-Aware Pedestrian Trajectory Prediction with Spatio-Temporal Attention Model”. In: *24th Computer Vision Winter Workshop*. 24th Computer Vision Winter Workshop, Jan. 2019. DOI: 10.3217/978-3-85125-652-9.

[64] Yingfan Huang et al. “STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019.

[65] Boris Ivanovic and Marco Pavone. "The Trajectron: Probabilistic Multi-Agent Trajectory Modeling with Dynamic Spatiotemporal Graphs". In: *ICCV 2019*. 2018.

[66] Y. Sun et al. "Socially-Aware Graph Convolutional Network for Human Trajectory Prediction". In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Mar. 2019, pp. 325–333. DOI: 10.1109/ITNEC.2019.8729387.

[67] Lidan Zhang, Qi She, and Ping Guo. "Stochastic Trajectory Prediction with Social Graph Network". In: *ArXiv* abs/1907.10233 (2019).

[68] Abduallah A. Mohamed et al. "Social-Stgcnn: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction". In: *ArXiv* abs/2002.11927 (2020).

[69] Agrim Gupta. *Social GAN Github Repository*. https://github.com/agrimgupta92/sgan.

[70] J. Ferryman and A. Shahrokni. "PETS2009: Dataset and Challenge". In: *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. Dec. 2009, pp. 1–6. DOI: 10.1109/PETS-WINTER.2009.5399556.

[71] *Stanford Drone Dataset Website*. https://cvgl.stanford.edu/projects/uav_data/.

[72] *Trajnet Official Website*. http://trajnet.stanford.edu/.

[73] J. Shao, C. C. Loy, and X. Wang. "Scene-Independent Group Profiling in Crowd". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 2227–2234. DOI: 10.1109/CVPR.2014.285.

[74] Barbara Majecka. "Statistical Models of Pedestrian Behaviour in the Forum". In: *Master's thesis, School of Informatics, University of Edinburgh* (Jan. 2009).

[75] F. Bartoli et al. "MuseumVisitors: A Dataset for Pedestrian and Group Detection, Gaze Estimation and Behavior Understanding". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2015, pp. 19–27. DOI: 10.1109/CVPRW.2015.7301279.

[76] Andreas Geiger et al. "Vision Meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[77] Rohan Chandra et al. “TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 8475–8484.

[78] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. “Joint Attention in Autonomous Driving (JAAD)”. In: *ArXiv* abs/1609.04741 (2016).

[79] R. J. Williams and D. Zipser. “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks”. In: *Neural Computation* 1.2 (1989), pp. 270–280.

[80] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *ArXiv* abs/1512.03385 (2015).

[81] *Kaggle*. https://www.kaggle.com/.

[82] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *ArXiv* abs/1405.0312 (2014).

[83] Baran Nama. *Social-LSTM Implementation on the TrajNet Dataset*. https://github.com/quancore/social-lstm.

TRITA-EECS-EX-2020:331