Deep Networks for Human Trajectory Forecasting: LSTM versus Transformer

Xinxin Tan, Matthieu Larnouhet

***Abstract*—When navigating crowded spaces, human beings will adapt their trajectory so that they do not collide with either obstacles or other pedestrians. The task of predicting these trajectories can be seen as a sequence generation problem, where we use the past positions of a given individual to predict his future positions. Our goal is to implement two different kind of models (LSTM and transformer) and compare their performance when it comes to this problem.**

I. Introduction

Human trajectory forecasting is an important module in applications such as automatic driving and surveillance systems. Long Short-Term Memory Networks (LSTM), have shown improvements over RNNs when it comes to dealing with long-distance dependencies and have been widely applied and shown good results. However, due to the excellent performance achieved by Transformers in various tasks related to sequential data, a question arises: Can the Transformer architecture generate better results than LSTM when it comes to predicting human trajectory?

In this project, we make a comprehensive comparison between LSTM and Transformer networks on the task of predicting human trajectories. We attempt to implement these two different networks from scratch to get an deeper understanding of their architecture as well as training methods and how they process sequential data and to further compare the advantages and disadvantages of the two models on this task.

II. Background Theory and Related Work

A. Physics based models

Prior to using data driven models for trajectory predictions, physics base models were used. An example of such a model is the Social Forces model which uses attractive and repulsive forces to model interactions between pedestrians. The major shortcoming of these models is that they use hand crafted features and cannot anticipate future actions based on previously seen patterns, which is why we focus here on data driven models.

B. LSTM and Transformer

Human trajectory is a kind of typical time series data, so it's a natural idea to process them with recurrent neural networks. In order to deal with the vanishing gradient problem present when using RNNs, long short term memory networks (LSTM) introduce a different architecture which consists in a cell, an input gate, an output gate and a forget gate. The cell can remember values over extended periods of time. The input, forget and output gates respectively control what information is added to, removed, or output from the cell. Let f_t, i_t, o_t be the activation vectors of the forget, input and output gates and $\tilde{c}_t$ be the cell input activation vector. We have the following equations:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad h_t = o_t \odot \sigma_h(c_t)$$

Likewise, the Transformer, a widely used architecture in the field of natural language processing, has been frequently used in machine learning tasks related to sequential data. What is special about Transformers is that they improve on the shortcomings of RNN regarding long-distance dependencies and parallel computing. Its unique multi-head self-attention module allows the Transformer model to process data at different positions in parallel and to capture long-distance dependencies in input sequences through a combination of embeddings and positional encodings.

III. Method

A. Goal

We seek to predict future positions of an individual by using its previous positions. The data that we have at our disposal consists of scenes where, for each time step t, the coordinates of the i^{th} pedestrian are given by (x_t^i, y_t^i). We use the 8 previous positions (from T_0 to T_{obs}) to compute the next 12 ones (from T_{obs+1} to T_{pred}). In order to do so, we will implement two different kinds of models, a Long Short Term Memory (LSTM) network and a transformer network. In order to assess the quality of the predictions, we use metrics such as average displacement error (ADE) and final displacement error (FDE). The ADE represents the mean squared error between the estimated points of a trajectory and their ground truth while the FDE represents the distance between the predicted final destination and the true final destination.

B. Dataset: TrajNet

The TrajNet dataset is a large multi-scene trajectory dataset introduced by A. Sadeghian et al. from Stanford University in their paper ”Trajnet: Towards a benchmark for human trajectory prediction” in 2018 [1]. This dataset is synthesised from several classical trajectory datasets in common use, such as the ETH dataset, the UCY dataset and the Stanford

Drone Dataset (SDD), which provide different scenarios from different views, e.g. the data for the SDD includes orthogonal bird's eye views provided by high-altitude UAVs (Fig.1).

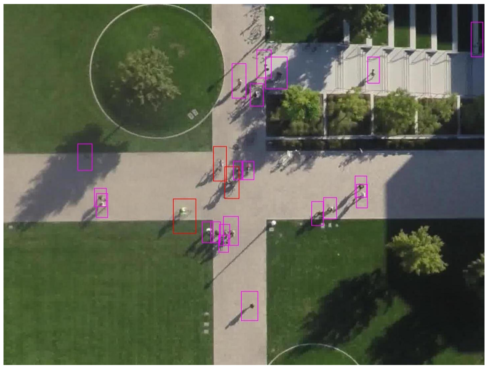

Fig. 1: The human trajectory detection from Stanford Drone Dataset (SDD). Image source: [2].

C. LSTM

We started by using a simple LSTM to predict trajectories. We attached one LSTM model to each individual and the weights are shared across individuals. The input to the LSTM consists in the 8 previous positions of the pedestrian. These input positions are embedded in a higher dimensional space (64) and then processed by the network which learns to predict future positions. We then compute the L2-loss between the predictions and the ground truth and update the weights of the network through back-propagation. During inference time, from times T_{obs+1} to T_{pred}, the previously predicted positions are used to predict the position during the next time step.
We pre-process the data by performing data augmentation such as symmetry relative to the x or y axis or by rotating the trajectory by a random angle. The hidden state and cell state dimensions are chosen to be 128. We use the Adam optimizer and a learning rate of 10^{-4} which is reduced later during training to help the network converge.

One downside of using a regular LSTM network is that it considers that the trajectories within a given scene are independent from each other and thus doesn't take into account social interactions. In order to do that, we use a model called social LSTM (Fig.2) which uses a social pooling grid to take into account the interactions between individuals. This grid allows neighboring pedestrians to share their hidden states h_t^i (hidden state of person i at time step t) with each other. In order to create this grid, we assign a LSTM network to each individual within a scene, and for each of those individuals we compute a grid that contains the hidden states of nearby pedestrians. For a given neighborhood size N_0 and a hidden state dimension D, let H_t^i be the grid of pedestrian i at time step t, its shape is $N_0 \times N_0 \times D$ and its elements are given by:

$$H_t^i(m,n,:) = \sum_{j \in \mathcal{N}_i} 1_{mn}[x_t^j - x_t^i, y_t^j - y_t^i] h_{t-1}^j$$

This grid is then embedded into a 64 dimensional vector and used as input to the LSTM in addition to the positions, which are also embedded like explained previously.

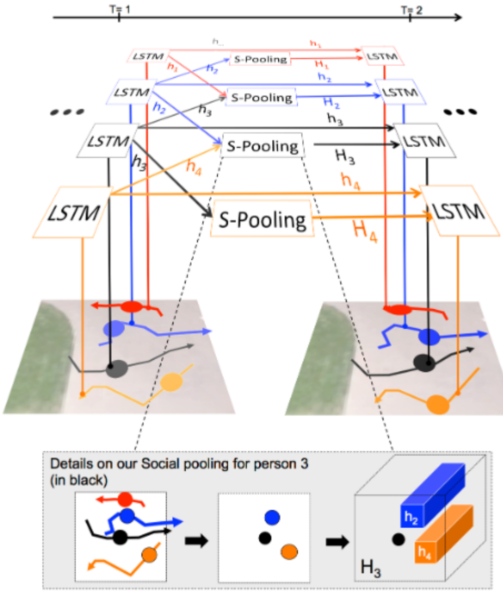

Fig. 2: Social Pooling Mechanism [3]

The model then updates the hidden state and uses it to output 5 parameters, which are the parameters of a 2D gaussian distribution (mean, variance and correlation). Those parameters are learned by minimizing the negative log-Likelihood loss (L^i is the loss for the pedestrian i):

$$[\mu_t^i, \sigma_t^i, \rho_t^i] = W_p h_{t-1}^i$$

$$L^i = -\sum_{T_{obs+1}}^{T_{pred}} \log(\mathbb{P}(x_t^i, y_t^i | \mu_t^i, \sigma_t^i, \rho_t^i))$$

The coordinates for the next time step are then predicted by sampling from this distribution. The sampling process expresses the uncertainty associated with the prediction. During inference, from T_{obs+1} to T_{pred}, we use the predicted positions to compute the grid for the next positions. The neighborhood size is set to 32, the grid size is 8 × 8, the hidden state and cell state dimension in still 128 and we use the Adagrad optimizer with a weight decay of 0.005.

D. Transformer

We implemented the classical Transformer network described in the paper "Attention Is All You Need" [4] from scratch, as shown in the Figure 3. A number of 8 heads are implemented for the multi-head self-attention layer, and both the encoder and decoder are made of six layers stacked together. Each of these layers is composed of an attention module, a feed-forward layer and 2 residual connections.
In order to utilize more information, the [Frame, Pedestrian ID, X-coordinate, Y-coordinate] feature set of the original data is used to compute the [X-coordinate, Y-coordinate, X-velocity, Y-velocity] vector during the dataloading process which is then

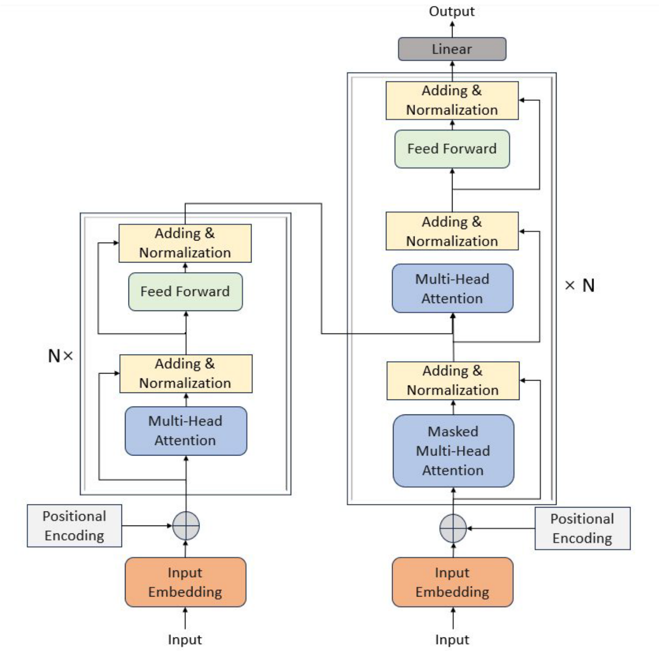

Fig. 3: The classical structure of Transformer network we implemented.

embedded into a higher dimensional space. Unlike LSTM, the Transformer doesn't process the input sequentially and thus needs to use positional encoding. Let e_t^i be the input embedding of trajectory i at time t, we add to it a positional embedding vector $p^t = (p_{t,d})_{d=1}^D$ of the same dimensionality D given by:

$$p_{t,d} = \begin{cases} \sin(\frac{t}{10000^{d/D}}) & \text{if d is even} \\ \cos(\frac{t}{10000^{d/D}}) & \text{if d is odd} \end{cases} \quad (1)$$

We choose D to be equal to 512. The dimension of the feedforward layer was then chosen to be 2048, the same value used in the original Transformer model. Meanwhile, to reduce overfitting and improve the convergence ability, we use a cosine decay strategy with a period of 20 epochs to dynamically adjust the learning rate (beginning with 1e-4), as shown in Fig.4. Dropout with probability of 0.1 is also applied although it is not shown in the structure.

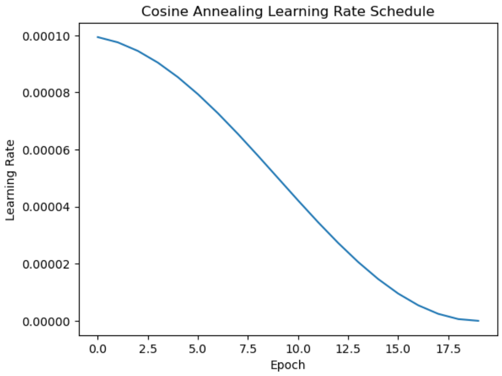

Fig. 4: Learning rate variation over 20 epochs of the Transformer training process.

IV. Results and Discussions

A. Metrics and discussion

The testing phase is done using a leave-one-out-cross-validation approach, which means that for each scene that we want to test the model on, we are first going to train the model on all of the remaining scenes. This is done five times in total, once for each scene. The results are summarized in the table Tab I. Overall, we can see that the transformer model shows better performance than the LSTM models and that the social LSTM model performance is slightly better than the regular LSTM. A reason why the transformer model outperforms the LSTM based models is that LSTMs process the input positions sequentially and then predict future positions while a transformer based model will make use of the attention mechanism and will weigh the previous positions differently based on how useful they are in predicting the next positions. The most useful points for predicting future positions aren't always the last ones; see for example Fig.5

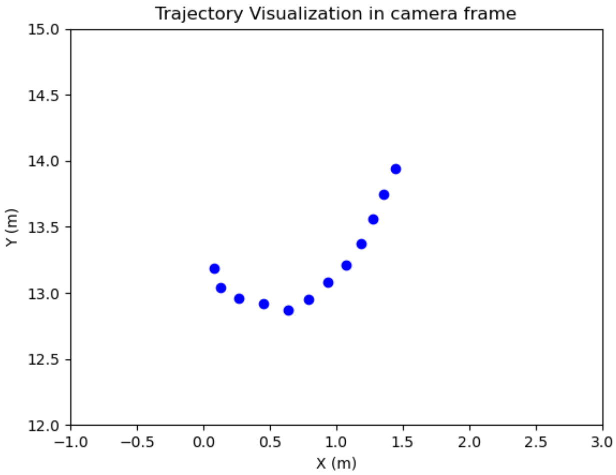

Fig. 5: Observation sample

Metric(m)	Dataset	LSTM	Social LSTM	Transformer
ADE	ETH	1	0.9	0.65
	HOTEL	0.6	0.52	0.62
	ZARA 1	0.5	0.6	0.66
	ZARA 2	0.6	0.5	0.65
	UNIV	1	1	0.71
FDE	ETH	2.1	2	1.25
	HOTEL	1.3	1.2	1.22
	ZARA 1	1.1	1.4	1.28
	ZARA 2	1.2	1	1.27
	UNIV	2.1	1.9	1.33

TABLE I: Performance metrics table

In this example, a LSTM based model will focus on the last observations whereas a transformer based one will focus on the observations that are part of the turn (the ones whose x coordinate lies between 0 and 1). In general the models tend to perform better on ZARA1, ZARA2 and HOTEL than on ETH and UNIV. This is because the former datasets contain more trajectories that are regular and linear whereas a dataset such as ETH contains a lot of non linear trajectories where a pedestrian will either stop walking or make a sharp turn.

B. Visualization

For each model, we can plot the observations next to the ground truth and the predictions (see Fig 6).

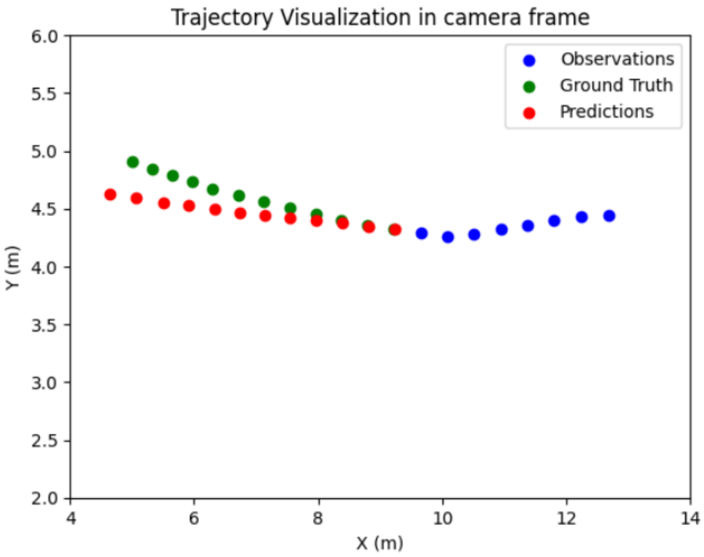

Fig. 6: LSTM prediction sample

We can also do the same thing with the transformer (see Fig 7).

C. Failure cases

By looking at failure cases, we can identify several recurring scenarios were the networks fail to predicting the trajectory. Common failure cases include instances were the pedestrian abruptly stops in the middle of their trajectory, or sometimes stops and then starts walking again. in general, non linear trajectories involving sharp turns have proven difficult to predict. Theses scenarios usually correspond to cases were the trajectory of the individual was most likely influenced by something other than the trajectory of other individuals. Indeed, a pedestrian could stop walking to look at something in their environment, or they could make a sharp 90° turn because they're at a crossroad. These trajectories are difficult to predict because they are influenced either by the environment or something else entirely. A model such as social LSTM encodes spatial information related to nearby pedestrians but it doesn't encode spatial information related to the scene and cannot predict this kind of behaviour.

V. Conclusion

We have presented 2 models, one is based on LSTM networks and was improved using social pooling. The other is based on Transformers. Overall the Transformer model outperforms LSTM based methods thanks to the attention mechanism. Improving these models would consist in trying to encode spatial information related to the environment so that the model is able to deal with heavily non-linear trajectories.

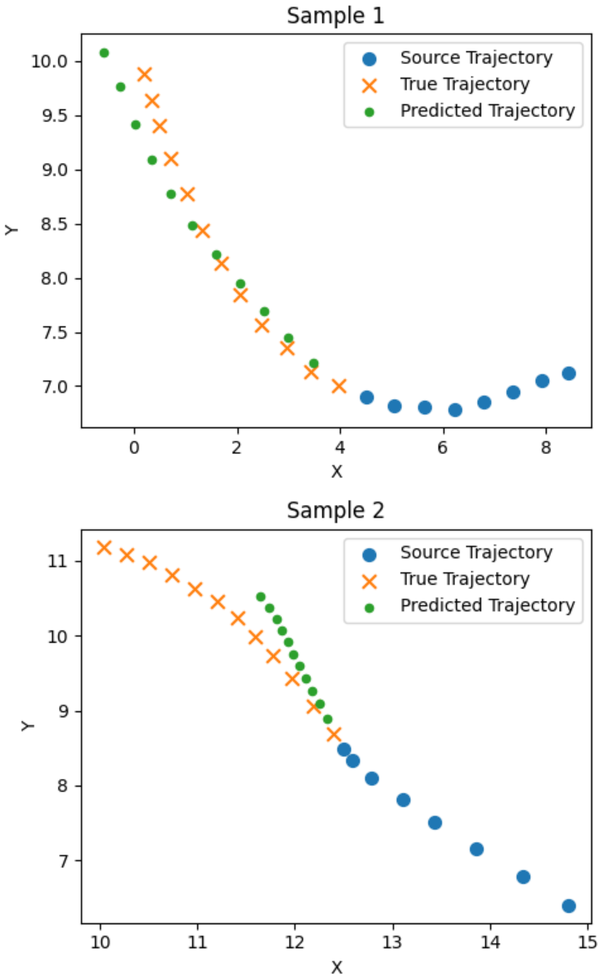

Fig. 7: Predictions for two randomly selected samples.

References

[1] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.

[2] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Stanford Drone Dataset, 2016. Available at: http://cvgl.stanford.edu/projects/uav_data/.

[3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM:Human Trajectory Prediction in Crowded Spaces, 2016. Available at: https://ieeexplore.ieee.org/document/7780479.

[4] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.