On the reliability of LSTM-MDL models for pedestrian trajectory prediction

Ronny Hug, Stefan Becker, Wolfgang Hübner, and Michael Arens

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation,
Gutleuthausstr. 1, 76275 Ettlingen, Germany
{ronny.hug,stefan.becker,wolfgang.huebner,
michael.arens}@iosb.fraunhofer.de

Abstract. Recurrent neural networks, like the LSTM model, have been applied to various sequence learning tasks with great success. Following this, it seems natural to use LSTM models for predicting future locations in object tracking tasks. In this paper, we evaluate an adaption of a LSTM-MDL model and investigate its reliability in the context of pedestrian trajectory prediction. Thereby, we demonstrate the fallacy of solely relying on prediction metrics for evaluating the model and how the models capabilities can lead to suboptimal prediction results. Towards this end, two experiments are provided. Firstly, the models prediction abilities are evaluated on publicly available surveillance datasets. Secondly, the capabilities of capturing motion patterns are examined. Further, we investigate failure cases and give explanations for observed phenomena, granting insight into the models reliability in tracking applications. Lastly, we give some hints how demonstrated shortcomings may be circumvented.

Keywords: recurrent neural networks, pedestrian trajectory prediction, generative models

1 Introduction

One component of an object tracking system is the estimation and prediction of object motion based on observed measurements. Traditionally, this process is modeled using a Bayesian formulation [1] in approaches like the Kalman filter [2] or nonparametric methods, such as particle filters [3]. Alternatively, the inference problem can be construed as a sequence generation problem. This way, we can step in the direction of recurrent neural networks, like Long Short Term Memory (LSTM) networks [4], which are commonly used for sequence generation and processing. Due to the recent success of LSTM models in a variety of sequence processing tasks, like speech recognition [5][6] and caption generation [7][8], these models seem like a natural choice for the task of pedestrian trajectory prediction. In particular, we focus on the model introduced by Alex Graves [9], which was originally designed for the generation and prediction of handwriting. His proposed model (subsequently referred to as LSTM-MDL model),

which consists of a LSTM network with a mixture density layer (MDL) [10] stacked on top, gained popularity in recent years and is applied in a wide range of applications, for example the prediction of basketball trajectories [11]. In the context of motion prediction in video surveillance data, the work of Alahi et al. [12][13] or Bartoli et al. [14] utilizes the aforementioned model to generate single trajectory predictions based on multiple correlated trajectories. Due to the capabilities of recurrent neural networks to model arbitrary functions and the results of the LSTM-MDL model in handwriting generation, we expect an adaptation of the model to be well suited for modeling complex trajectories in video surveillance data. Here, a trajectory with a significant number of state changes, which are mainly influenced by statistical long-term dependencies, is defined as complex. Since the domain of object tracking is particularly different to handwriting prediction in certain aspects, like position-dependent movement patterns, an adaptation of the model to use positional information is crucial.

The main contribution of this paper is an extensive evaluation of a slightly modified version of the LSTM-MDL model, combined with an investigation on the reliability of this model in the context of pedestrian trajectory prediction. Thereby, when it comes to evaluating the model, the fallacy of solely relying on prediction metrics, such as the final or average displacement error [12][15][16], is demonstrated. Further, it is shown how the overall capabilities of the model can lead to suboptimal prediction results. Here, such prediction metrics can lead up to false conclusions in evaluating the model as a whole, as these metrics are unable to describe the full range of the models capabilities to capture and represent motion in a scene. Towards this end, two-folded extensive experiments are provided. In a first step, the models abilities in predicting pedestrian trajectories is evaluated on several publicly available surveillance datasets. This evaluation shows that the model performs poorly, especially on datasets that consist mainly of complex trajectories, which is not obvious at a first glance. To ensure that this is not a cause of the models inability to capture complex, non-systematic motion from data, we examine the models modeling capabilities in our second series of experiments. As a last step, we further investigate failure cases of the model in the prediction task based on both of our experiments and give explanations for observed phenomena through synthetic toy examples. Furthermore, we demonstrate some problems and shortcomings that might occur when this model is applied to video surveillance tasks and give some hints on how these may be circumvented.

The remainder of this paper is structured as follows. Following this section, we introduce the variation of the LSTM-MDL model that we have used in this paper and discuss the adaptations made (section 2). After that, we are investigating the reliability of this model for pedestrian trajectory prediction, going through the experimental section (3). Finally, we showcase and discuss different problems and shortcomings with the model, that occur regarding prediction tasks (section 4). Section 5 provides some final conclusions.

2 Preliminaries

For modeling human motion in a generative fashion, we are using a slight modification of the LSTM-MDL model described in [9]. Recurrent neural networks are a natural choice for modeling sequence data, such as human motion represented as timely-ordered sequences of spatial positions. Additionally, regarding the context of tracking pedestrians in a surveillance scenario, generative modeling provides advantages over discriminative training, as neither labeled training data nor negative examples are required. Especially the latter is rather difficult to obtain, as the concept of a negative example concerning pedestrian motion is not well-defined. In its original form, the network was used to model handwritten text, represented as a sequence of pen position offsets. Therefore, the network learns an offset distribution conditioned on previous offsets. Consequently, sequence prediction, like endpoint prediction, is modeled implicitly by this model.

While utilizing offsets helps stabilizing the learning process, presumably due to the limitation of the input and output spaces, spatial information gets lost (which does not harm handwriting generation). More specifically, spatial information only persists in an implicit fashion by performing path integration. Given this fact, it is not ensured that the model captures spatial points of interest when learning the model. In the context of handwriting generation, this particular feature is not necessary, as generated handwriting does not have to consider e.g. obstacles in space. As a consequence, prediction in the original LSTM-MDL model mostly relies on the previously observed sequence. This is where a major difference between handwriting generation and pedestrian trajectory prediction comes into play, where positions in a scene may lead to previous information being disregarded in favor of immediate predictions given the current location, e.g. for avoiding a static object. In this respect, the actual position of a person in a scene is important when modeling future movement and must be used as an input to the model. Following this, there are two possible ways to further adapt the original model given the current position: either by trying to predict the next *position* $\mathbf{x}^{t+1}$ (where will the person *be* next) or the next *offset* $\boldsymbol{\delta}^t$ (where will the person *go* next). During our experiments, we found that predicting an offset distribution increases numerical stability while training, and also improves results in general, as opposed to predicting a positional distribution. Such changes were, for example, also made in [12], where the network is used in that way to predict motion in crowded environments, generating a positional distribution. In their work they stacked another neural network on top to model crowd interactions, which may help in coping with instable positional predictions. Since their main focus was on predicting motion, a single Gaussian is used instead of a multi-component Gaussian mixture, thus further reducing the problem complexity and modeling capabilities. In contrast, we examined the prediction and modeling capabilities of the model. Ultimately, we adapt the model by changing the input layer to take positions instead of offsets. The adapted model is depicted in figure 1. The output $\mathbf{y}$ of the model is a $6 \times K$ dimensional vector and consists of the parameters of a K component Gaussian mixture model: $\mathbf{y} = (\mu_k, \sigma_k, \rho_k)_{k=1,\dots,K}$. This model generates a distribution with respect to the

next offset $\boldsymbol{\delta}^t$. The next trajectory position can then be obtained by calculating $\mathbf{x}^t = \mathbf{x}^{t-1} + \boldsymbol{\delta}^{t-1}$ with $\boldsymbol{\delta}^{t-1} \sim \mathbf{y}^{t-1}$.

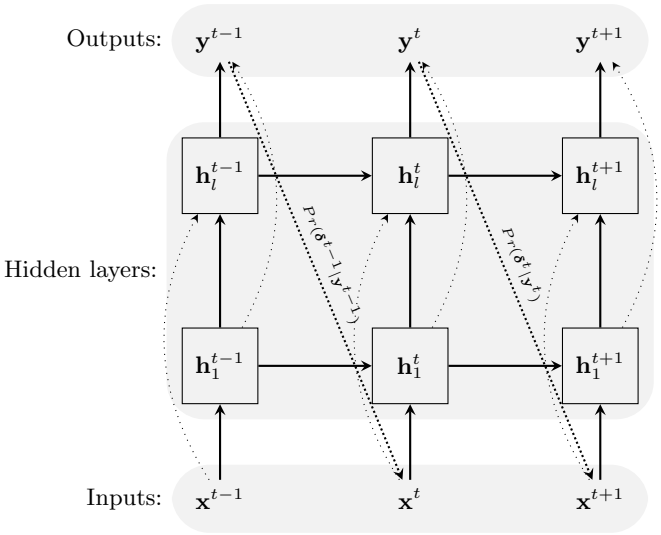

Fig. 1. Recurrent neural network prediction architecture with skip connections as described by Graves [9]. The model has been modified to take positions as input rather than offsets.

This model can be trained in a generative way, by maximizing the likelihood of the data given the output Gaussian mixture parameters. Therefore, the loss function $\mathcal{L}$ is defined as:

$$\begin{aligned}\mathcal{L}(\mathbf{x}) &= \sum_{t=1}^{T} -\log\left\{\sum_k \pi_k^t \mathcal{N}(\boldsymbol{\delta}^{t+1}|\mu_k^t, \sigma_k^t, \rho_k^t)\right\} \\ &= \sum_{t=1}^{T} -\log\left\{\sum_k \pi_k^t \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left\{\frac{-Z}{2(1-\rho^2)}\right\}\right\} \qquad (1)\end{aligned}$$

with

$$\boldsymbol{\delta}^{t+1} = \mathbf{x}^{t+1} - \mathbf{x}^t,$$

$$Z = \frac{(\delta_1-\mu_1)^2}{\sigma_1^2} + \frac{(\delta_2-\mu_2)^2}{\sigma_2^2} - \frac{2\rho(\delta_1-\mu_1)(\delta_2-\mu_2)}{\sigma_1\sigma_2}.$$

It is important to note, that $\mu_k^t, \sigma_k^t, \rho_k^t$ themselves are dependent on the whole history of inputs $(\mathbf{x}^t, \mathbf{x}^{t-1}, ..., \mathbf{x}^0)$. Further, we opt to disregard the skip connections in the following, as gradient problems are less present during training in our rather small network. With this decision, we also follow other works (e.g. [12]) that do not use skip connections in their models.

3 Evaluation

In this section the effectiveness of the adapted LSTM-MDL model is evaluated for the task of pedestrian trajectory prediction (section 3.1). Additionally, we ran an experiment to test the capabilities of the model to capture the statistics present in the data (section 3.2). For evaluation, a selection of widely-used, publicly available surveillance datasets are used. Each of these datasets observes one or multiple scenes from a bird eye view. The selected datasets cover real world scenarios under varying crowd densities and varying complexity of trajectory patterns.

In order to differentiate between varying complexities of trajectory patterns, and thereby also ranking these datasets, a categorization of observed motion is required. The motion of an object can not only differ in the way the object follows a path, but also in the form of the path. According to the motion type, it is possible to differentiate between constant and accelerated motion. In case of accelerated motion, it can further be distinguished between uniformly accelerated and unequally accelerated motion. Depending on the form or shape of the trajectory, it is common to differentiate between rectilinear and curvilinear motion. Furthermore, the dynamics of an object can vary over time and also depend on location information. In accordance with the introduced motion categorization and the spatio-temporal context of motion, we define a simple motion as a constant, rectilinear, and temporally constant motion. In return, an object performs a very complex motion when it is unequally accelerated, curvilinear, and spatio-temporal varying.

Our entire evaluation was initially based on 3 datasets that together contain 5 scenes. Details of these datasets are summarized in table 1.

Dataset	UCY[17]	UCY	EWAP[16]	EWAP	SSD[18]
Details	Students	Zara	ETH	Hotel	Hyang
video frame rate	25	25	25	25	29
annotation rate	non-equidistant	non-equidistant	2.5	2.5	non-equidistant
interpolated	✓	✓	✗	✗	✓
number of trajectories	967	489	360	390	219
mean trajectory length	13.481	11.043	24.744	16.779	1581.087

Table 1. Details of pedestrian trajectories datasets.

For the purpose of this paper, we focus on the scenes *ETH*, taken from the ETH Walking Pedestrians (EWAP) dataset [16], and *Hyang*, taken from the Stanford Drone Dataset (SDD) [18]. We assume that these two scenes suffice to show and discuss capabilities of the model, as these scenes represent the two opposite ends of the covered range regarding the complexity of the underlying

data. Although in the chosen exemplary surveillance scenarios with an aerial view the complexity is limited compared to other scenarios like video network data or data captured on board a vehicle, it still allows for a variety in complexity of observed motion. This holds true for the scenes chosen for our evaluation. The trajectories from the *ETH* scene can be described as rather simple, where the amount of spatio-temporal variation is very limited and most trajectories can be adequately described with a constant velocity model following a straight line. However for *Hyang* the scene context strongly influences the person trajectories. For example, there are sidewalks with junctions leading to spatially depending changes in the curvature of the trajectories. With the occurring switches between rectilinear and curvilinear motion and an increased amount of walking paths this scenario can be rated as complex. To provide a better understanding of both scenes, example trajectories are depicted in figure 2.

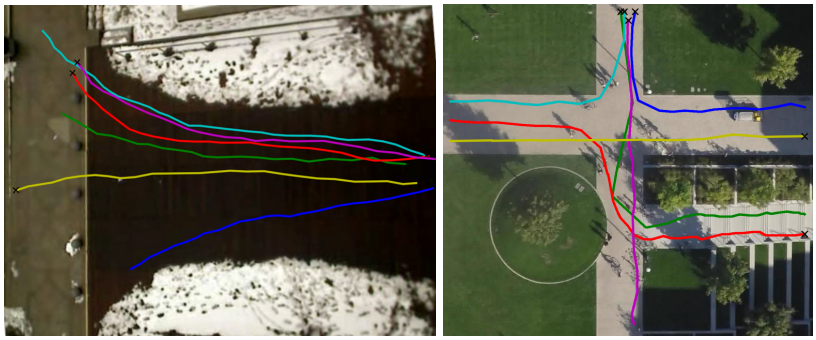

Fig. 2. Example trajectories taken from *ETH* (left) and *Hyang* (right) scenes.

For our experiments, we slightly modified the data of the *Hyang* scene[1]. First, the scene is recorded with a drone from different positions and split into 15 separate recordings containing different trajectories. Seven of these have a large overlap in the observed parts of the whole scenery. Because of that, we are able to project the trajectory data of these recordings into a single coordinate system, using a homography matrix for the transformation, to create a larger set of trajectories for training and evaluation. Secondly, the provided data provides manual annotation, roughly every 30 frames. Between these manual annotations, automatic annotations have been generated. We have replaced these automatic annotations with a linear interpolation to gain an equidistant annotation rate. Using a linear interpolation seemed to improve the results produced by a learned model. Lastly, we removed some trajectory association errors and annotation drift at the borders of the scene. In this paper we are using this version of the *Hyang* scene dataset.

[1] The modified ground truth will be provided upon request.

3.1 Applying the LSTM-MDL model for prediction

For evaluating the predictive capabilities of the model, we are measuring the performance on the task of predicting the endpoint of a pedestrian trajectory, given a portion of the respective trajectory. We trained 27 different model configurations for each scene in our evaluation, with a varying number of layers (1, 2, and 3), LSTM cell state size (128, 256, and 512 dimensional) and number of components (4, 8, and 16) in the Gaussian mixture model that is output by the model. Each configuration was trained 3 times on a different random subset of the respective dataset, using stochastic gradient descent with a learning rate of 0.005 and an exponential learning rate decay rate of 0.95. The unroll length is set to the lower percentile of the trajectory lengths in the given dataset. This provides a trade-off between the observation length and the number of training examples. In the evaluation, the results of these 3 iterations are averaged to cope with random effects in the trained models. No skip connections, as proposed by Graves [5], were used. The model is implemented in tensorflow [19].

To measure the endpoint prediction we are using the *final displacement error*

$$FDE = \frac{\sum_i \sqrt{(\hat{y}_i - y_i)^2}}{n} \tag{2}$$

where $\hat{y}_i$ and y_i are the predicted and real endpoints of all n trajectories T_i in the test dataset. These trajectories were not part of the training dataset and make up for 20 percent of the whole dataset. The predicted endpoint is determined by passing the first t trajectory points into the model and then generating the remaining l points by using a maximum likelihood estimate on the output mixtures. Here, the parameter t is set to be equal to the unroll length used at training time. The performance of the model is compared against a simple linear predictor that takes the last two of the t trajectory points to calculate a constant offset. This offset is added to the last given trajectory point l times to predict the trajectory endpoint. The prediction results for the *ETH* scene are depicted in figure 3.

In this simple scene, the model expectedly performs better than the linear predictor for nearly all trained configurations. This scene mostly features straight or slightly bent trajectories. For the latter, the linear predictor performs worse, as it doesn't capture curvature. The LSTM-MDL model on the other hand is capable of learning and representing more complex paths, meaning it can capture curved paths. Additionally, the linear predictor propagates a constant velocity in its prediction, whereas the LSTM-MDL model is, generally speaking, capable of modeling several velocity profiles and applying these according to given observations. Given these results, we expect the model to excel in more complex scenes, where curved paths are more common. Following this, we proceeded and scaled up the complexity of the data used. Figure 4 shows the results of the endpoint prediction for the *Hyang* scene.

This dataset contains a lot more major walking paths including two junctions, where the predictor has to choose from several continuations. Given the fact that the adapted LSTM-MDL model is capable of capturing a large variety of different

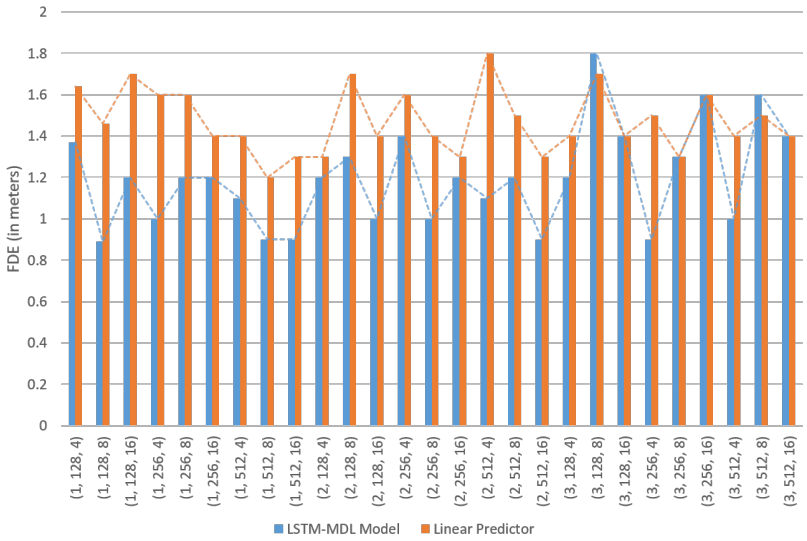

Fig. 3. Endpoint prediction results for *ETH* for the trained configurations (number of layers, LSTM cell state size, and number of gaussian components) of the LSTM-MDL model and the linear predictor.

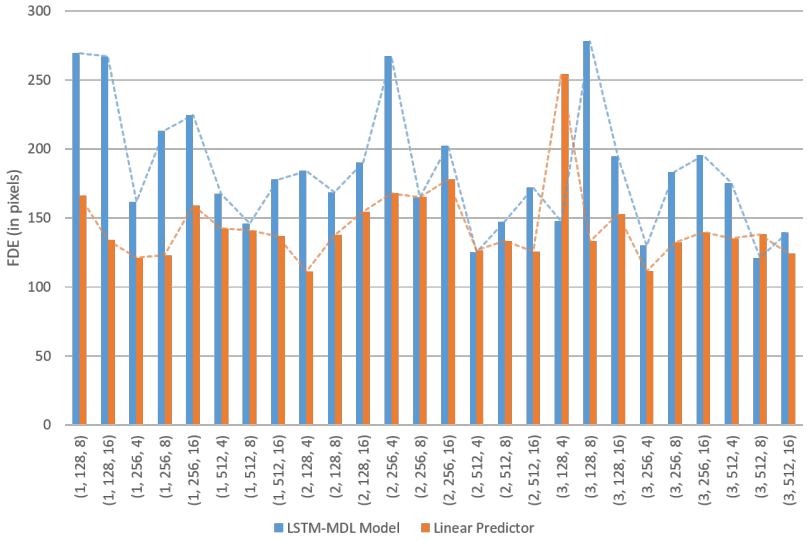

Fig. 4. Endpoint prediction results for *Hyang* for the trained configurations (number of layers, LSTM cell state size, and number of gaussian components) of the LSTM-MDL model and the linear predictor.

paths and tendencies in choosing between different continuations at junction points, we expected the model to outperform the linear prediction by a large magnitude. Against the odds, the model fails to surpass the linear predictor. In fact, except in one case, the linear predictor outperforms the model. Furthermore, the results vary strongly depending on the trajectories used for testing, making the results supposedly random.

In order to give an explanation about why the adapted LSTM-MDL model outperforms the linear predictor for the first scene but fails on the second one, we first evaluate the models representation capacity in section 3.2. Following this, we try to give explanations for observed phenomena in section 4.

3.2 Evaluating the models representation capacity

Looking at the results of the previous section, it appears that the adapted LSTM-MDL model is not capable of capturing motion in more complex scenes. This raises the question if the model is incapable of capturing more versatile motion or if the endpoint prediction is not well suited to measure the performance of this model. As a first step in answering this question, we ran another experiment to test the models capability to capture the statistics of the data. In terms of pedestrian motion these statistics include offsets, magnitudes, and the so called path coverage. The last term describes the number of paths from specified sources to sinks in the scene, and the distribution of trajectories with respect to these paths.

For calculating these measures, the models generative capabilities can be used to generate sample trajectories from different starting positions in the scene. In the context of this paper, we are again focusing on the two datasets *ETH* and *Hyang*. We provide sources and sinks for both scenes in the form of rectangles. The set of sources is equal to the set of sinks in each dataset. A rough representation of the paths present in the datasets is depicted in figure 5.

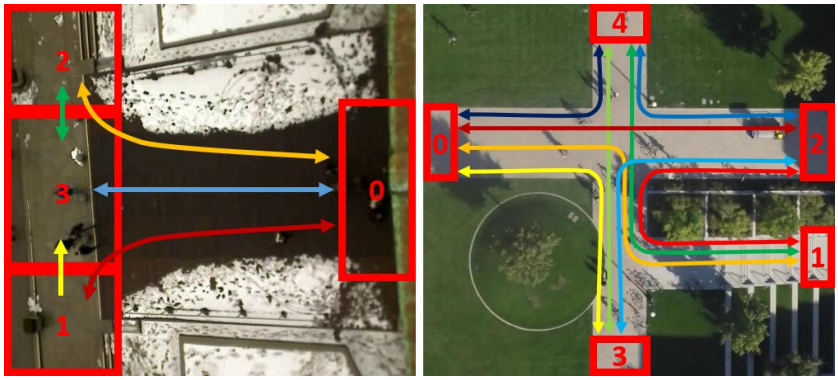

Fig. 5. Sources, sinks and main walking paths in the *ETH* (left) and the *Hyang* (right) scenes.

For generating sample trajectories from the model, we draw 100 positions from each source in the scene. Given these starting positions, we generate sample trajectories using the model by iterating the following steps:

1. Feed the current position into the model to generate a distribution for the next offset
2. Draw an offset from that distribution
3. Calculate the next trajectory point by adding the offset to the current position
4. Set the calculated point to be the next current position

We stop generating new positions when the trajectory enters one of the sinks (i.e. one of the other sources) in the scene or when we exceed a set limit of trajectory points. All sampled trajectories for one dataset are collected in a sample dataset $\mathcal{D}_S$, which will be compared to the training dataset $\mathcal{D}_T$.

For the comparison between $\mathcal{D}_S$ and $\mathcal{D}_T$ we calculate several statistics on both datasets:

1. Histogram of magnitudes
2. Histograms of x and y offsets
3. 2D Histogram of offsets
4. Path coverage

We compare the histograms by using the absolute Pearson correlation as a measure of similarity. These histograms will be referred to as the motion profile of the model. The path coverage is tested by checking which and how many of the paths in the original dataset are represented by the model. Additionally, the distribution of the samples over the paths is again compared by using the absolute Pearson correlation. As we have evaluated a total of 81 learned model checkpoints, table 2 depicts an example for a *good* (subscripted with plus) and a *bad* (subscripted with minus) performing checkpoint. Here, the *good* and *bad* checkpoints perform accordingly in the measured dimensions. The given values are the mean of 3 runs of the same configuration with a different random selection of the training set.

Checkpoint	**Motion profile**			**Path coverage**		
	Magnitudes	Offsets (x, y)	Offsets	Paths	Distribution	Outlier ratio
ETH_-	.318	.573, .934	.352	6/9	.827	18.7 %
ETH_+	.792	.818, .885	.422	5.67/9	.704	1.7 %
Hyang_-	.122	.870, .995	.844	7.33/20	.529	10.5 %
Hyang_+	.539	.964, .997	.929	10/20	.633	6.2 %

Table 2. Statistics for a good (subscripted with plus) and a bad (subscripted with minus) checkpoint of the *ETH* and *Hyang* scenes.

Figure 6 shows samples generated from these checkpoints. The left column depicts samples from a *bad* and the right columns from a *good* model checkpoint.

The two rows show samples from models learned on the *ETH* and the *Hyang* dataset, respectively. It is visible, that the *bad* models produce much more random trajectories. Still, for *ETH* both models capture the major walking paths well. In contrast, for *Hyang*, the *bad* model captures less paths and both junctions are biased towards the left border of the scene, causing samples to deviate from the usual walking paths.

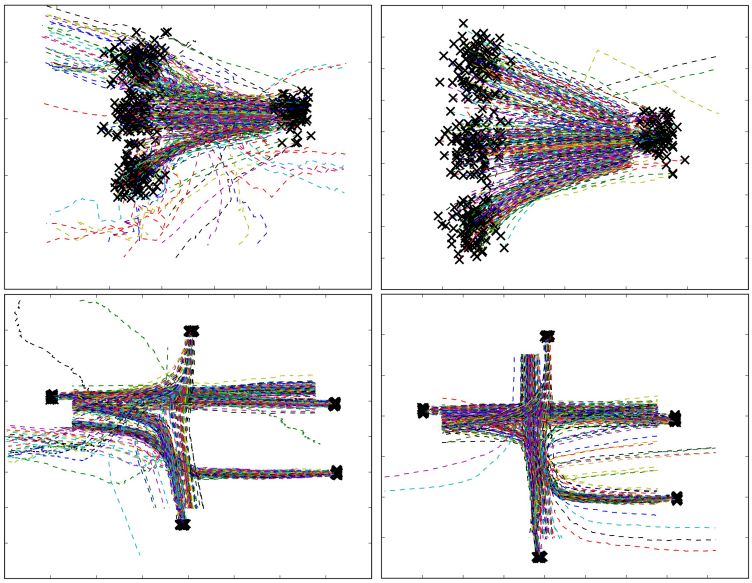

Fig. 6. Samples drawn from a good (right) and a bad (left) checkpoint learned on the *ETH* (first row) and *Hyang* (second row) scenes.

Concluding this evaluation of the data modeling capabilities of the adapted LSTM-MDL model, we figured that it is indeed capable of modeling complex motion in given trajectory data. On the other hand this also means, that our question, regarding the failure in the prediction task, is still unanswered and must be further investigated. Following this, we try to give reasons for this in section 4.

4 On the reliability of LSTM-MDL models for pedestrian trajectory prediction

Regarding the results shown in section 3.2, the adapted LSTM-MDL model is capable of capturing the statistics in the data, thus still leaving our question concerning the poor prediction results unanswered. Thinking further, the reason why the endpoint prediction is not particularly good and somehow appears to become more random the better the model is, lies in the task and the common

approach itself. Even if we have multiple hypotheses to choose from when predicting the endpoint of a given trajectory segment, we usually decide to go with the most probable path (maximum likelihood estimation), also if its probability is only slightly higher. That means, with an increasing number of paths, and especially junctions in those paths, the prediction will become more instable, the better the model is capable of representing the data. To clarify, this can be pictured using a small toy example. Take for example 3 synthetic datasets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ that contain trajectories that pass a 4-way junction. In $\mathcal{D}_1$, all trajectories start at the bottom and pass the junction to the left. Dataset $\mathcal{D}_2$ also contains trajectories starting at the bottom, but they pass the junction either to the left or to the right. The last dataset $\mathcal{D}_3$ adds the possibility of going straight over the junction. Now, if the model is large enough to cover all these paths, it may generate sample trajectories starting from the exact same position that end in very different positions. This is depicted in figure 7.

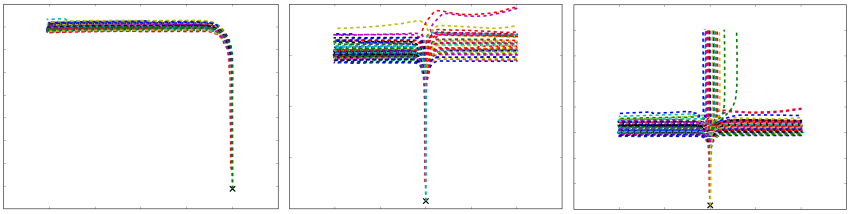

Fig. 7. Toy example datasets to showcase the problem with the modeling power of the adapted LSTM-MDL model for prediction tasks.

Concerning endpoint prediction, we have a chance of completely missing the true endpoint of 0, 50, and 66 percent. Depending on the absolute positioning of the trajectory itself, it may be biased more or less towards a specific endpoint region. This, in fact, still does not need to hold true for many trajectories in a real world dataset.

Another aspect to consider is the number t of observed points prior to the prediction. If there is at least one junction on a trajectory's path, the observed points *before* that junction do not necessarily provide useful information about the continuation of that particular trajectory. Different observation lengths could indeed lead to some bias towards a specific direction, but depending on the variation in the training data, this is more or less helpful. In general, this fact has to be taken into account, as trajectories generally vary in length, due to different walking speeds of pedestrians. This also holds true, even if two trajectories follow roughly the same path through the scene, when these are progressing at different velocities. Here, the model will eventually have richer information for predicting the continuation of the faster moving target, as for the slower moving target, when the number of observed trajectory points is fixed, because the former has progressed much further into the scene. The impact of the observation length is illustrated in figure 8. In this figure, a trajectory (blue) from $\mathcal{D}_3$ is chosen, which

is heading towards the left side crossing the junction. The learned model should predict the most probable continuation of that trajectory (red), given trajectory points up to the junction, a few steps into the junction and mostly through the junction.

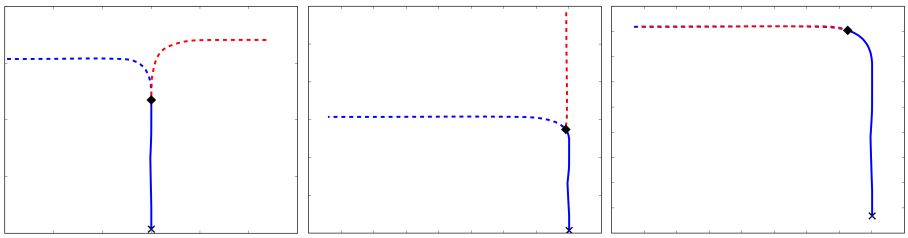

Fig. 8. Predicted continuation (red) of a partly observed trajectory (blue, starting at black cross) given 30 (right before the junction), 35 (few steps into the junction, already shows some tendencies), and 40 (almost through the junction, direction is now clear) trajectory points (up to the black diamont).

It can be clearly seen how the choice of the observation length affects the predictions of the model. In this particular case, having only observed trajectory points just before entering the junction, the model does not even consider going to the left. Given some more points that provide some information about the continuation, the model switches to predicting a straight motion, which still is not as expected. This may happen because the model is robust to noise, at least to some extent. In the last case, it is clear how the trajectory should continue: to the left. The model also captures this and does an appropriate prediction.

In conclusion, the adapted LSTM-MDL models capability of capturing and representing a variety of complex paths is great for generating data, but somewhat obstructive for prediction tasks, such as endpoint prediction. Regarding this, another question arises that should be investigated further in future works: How can we possibly fix this? No matter what the exact solution will be, it will most probably involve a more sophisticated approach to generate predictions that do not solely rely on the most probable hypothesis. In fact, to exhaust all possibilities given by the LSTM-MDL, an approach that tracks and incorporates all hypotheses generated by the model should be preferred over the common approach. Also, another adaption of the model that incorporates the positions and offsets may provide the model with richer possibilities to model pedestrian motion. Besides that, when using this model as a predictor, the restrictions and problems discussed in this paper should be kept in mind.

5 Conclusions

In this paper, we explored how naïvely applying a LSTM-MDL model for predicting human trajectories can lead to unreliable results. While these excel in handwriting generation, model adaptations that consider explicit spatial information can unexpectedly collapse in prediction tasks. Especially in the context of surveillance data, common metrics like endpoint prediction result in the fallacy that the model cannot correctly represent the data. In order to demonstrate the capabilities of the model to capture the training data characteristics, an extensive comparison of different underlying data statistics was provided. As the model proves to be capable of capturing the data statistics, we further revealed the occurring reasons for the problems and shortcomings of the model on synthetic data. Finally, we provided research directions on how to overcome some limitations of the model.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005)
2. Kalman, R.: A new approach to linear filtering and prediction problems. ASME Journal of Basic Engineering (1960)
3. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing **50**(2) (Feb 2002) 174–188
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8) (1997) 1735–1780
5. Graves, A., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: International Conference on Acoustics, Speech and Signal Processing. (May 2013) 6645–6649
6. Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., Bengio, Y.: A recurrent latent variable model for sequential data. In: Advances in Neural Information Processing Systems (NIPS). (2015)
7. Donahue, J., Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Conference on Computer Vision and Pattern Recognition, IEEE (2015)
8. Xu, K., Ba, J., Kiros, R., K.Cho, Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In Bach, F., Blei, D., eds.: Proceedings of the 32nd International Conference on Machine Learning. Volume 37 of Proceedings of Machine Learning Research., Lille, France, PMLR (2015) 2048–2057
9. Graves, A.: Generating sequences with recurrent neural networks. arXiv:1308.0850 (2013)
10. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
11. Shah, R., Romijnders, R.: Applying deep learning to basketball trajectories. arXiv:1608.03793 (2016)

12. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Conference on Computer Vision and Pattern Recognition, IEEE (2016) 961–971
13. Alahi, A., Ramanathan, V., Goel, K., Robicquet, A., Sadeghian, A.A., Fei-Fei, L., Savarese, S.: Learning to predict human behaviour in crowded scenes. In: Group and Crowd Behavior for Computer Vision. Elsevier (2017)
14. Bartoli, F., Lisanti, G., Ballan, L., Bimbo, A.D.: Context-aware trajectory prediction. arXiv:1705.02503 (2017)
15. Vemula, A., Muelling, K., Oh, J.: Modeling cooperative navigation in dense human crowds. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). (May 2017) 1685–1692
16. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: International Conference on Computer Vision. (2009) 261–268
17. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphic Forum **26**(3) (2007) 655–664
18. Robicquet, A., Sadeghian, A., Alahi, A., Savarese, S. In: Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes. Springer International Publishing (2016) 549–565
19. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.