Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

Original research article

Applying deep bidirectional LSTM and mixture density network for basketball trajectory prediction

Yu Zhao[a,*], Rennong Yang[a], Guillaume Chevalier[b], Rajiv C. Shah[c], Rob Romijnders[d]

[a] *Air Force Engineering University, Xi'an, China*
[b] *Laval University, Quebec G1V 0A6, Canada*
[c] *University of Illinois, Chicago, United States*
[d] *Eindhoven University of Technology, Eindhoven, Netherlands*

ARTICLE INFO

Article history:
Received 5 November 2017
Accepted 13 December 2017

Keywords:
Bidiretional LSTM
Mixture density network
Basketball trajectory
SportVu
Classification and prediction

ABSTRACT

Data analytics helps basketball teams to create tactics. However, manual data collection and analytics are costly and ineffective. Therefore, we applied a deep bidirectional long short-term memory (BLSTM) and mixture density network (MDN) approach. This model is not only capable of predicting a basketball trajectory based on real data, but it also can generate new trajectory samples. It is an excellent application to help coaches and players decide when and where to shoot. Its structure is particularly suitable for dealing with time series problems. BLSTM receives forward and backward information at the same time, while stacking multiple BLSTMs further increases the learning ability of the model. Combined with BLSTMs, MDN is used to generate a multi-modal distribution of outputs. Thus, the proposed model can, in principle, represent arbitrary conditional probability distributions of output variables. We tested our model with two experiments on three-pointer datasets from NBA SportVu data. In the hit-or-miss classification experiment, the proposed model outperformed other models in terms of the convergence speed and accuracy. In the trajectory generation experiment, eight model-generated trajectories at a given time closely matched real trajectories.

1. Introduction

Basketball has hundreds of years of history. Today, it is one of the core competitions in the Olympic Games. There is an increasing trend toward participation in basketball. In professional competitions, each team has a professional coaching team, who design a scientific training plan. Three-point shots, which are always the key to victory, require very important technical tactics. For players, mastering the three-pointer can greatly increase scoring opportunities. In the NBA game, the SportVu player tracking analysis system records everything using sensors and has thus replaced traditional analysis methods. SportVu can collect quantitative statistics regarding player performance, which is useful for post-evaluation and tactical decision-making. Even though the prediction of the basketball's trajectory is a toy problem, this model could contribute to progress in many other areas, for example, behavior recognition [1,2], robot control [3,4], or path planning of unmanned aerial vehicles (UAV) [5,6].

* Corresponding author.
E-mail address: zhaoyuafeu@gmail.com (Y. Zhao).

https://doi.org/10.1016/j.ijleo.2017.12.038
0030-4026/© 2017 Elsevier GmbH. All rights reserved.

The study of a basketball trajectory system can be approached in two ways: i) using mechanical models [7,8] or ii) using statistical models [9,11]. Mechanical models are simple and easy to use, but they require a solid theoretical foundation and usually require assumptions that limit the scope of application. Statistical models have high generalizability, but require a lot of data and rich feature-processing experience. Moreover, they usually require high computing power and long runtimes. In the 1990s, Hamilton and his collaborators establish a free basketball model based on kinematics that can calculate the best shot angle and speed [7]. Based on previous work, Tran and Silverberg [9] also considered the release height, side angle, and back spin. They concluded that players should aim the ball toward the back of the ring. As long as is the shooter is not covered by an opponent, the shooting height makes no differences to the hit rate. In recent years, people have tried to apply deep learning to trajectory prediction. In a study by Wang and Zemel [10], recurrent neural networks (RNNs) did not perform well in predicting players' motion due to the role of subjective consciousness. However, it achieved good results in classifying players' roles. Subsequently, Zheng [12] used deep hierarchical networks to model player macro-goals and micro-actions. He demonstrated significant improvement over non-hierarchical baselines in the experiment. But his model did not consider competition and cooperation, and it only used an imitation learning framework. Compared with predicting players' motion, the three-dimensional basketball's trajectory is much more complex. Shah and Romijnders [11] proposed an LSTM + MDN model for classifying hit-or-miss outcomes. In contrast to the conventional methods, such as the general linear model (GLM) and the gradient boost model (GBM), LSTM + MDN does not initially require feature extraction. In their paper, GLM and GBM could incorporate only the previous point in generating the trajectory sequence, whereas the proposed model can incorporate the whole sequence, which includes temporal information. The results show that using LSTM + MDN significantly improves hit or miss classification.

BLSTM is a variant of the RNN model, and its structure is particularly suitable for solving time series problems. Unlike LSTM, BLSTM can use forward and backward information. Graves and Schmidhuber [13] firstly proposed BLSTM in 2005, applying it for framewise phoneme classification. Since then, BLSTMs have shown state-of-the-art performance in speech recognition [14,15], natural language processing [16,17] and other areas [18,19]. In fact, researchers often use deep BLSTMs due to their strong learning ability. As deep BLSTMs are created by simply stacking multiple BLSTMs, they are very easy to construct.

In real life, there is always uncertainty, which can be described in terms of probabilities. Unlike conventional networks, which directly outpus real labels, MDN proposed by Bishop [20] can output the weighted sum of multiple probability distributions. Graves [21] showed detailed MDN calculation. Additionally, he used RNN and MDN for handwriting synthesis. The results showed that ordinary people could hardly distinguish personal handwriting from the artificially generated version. Zen et al. [22] used deep MDN to solve the problem of generating multiple outputs in speech synthesis to achieve more natural sound. The paper published in ICLR 2017 by Bazzani [23] made further significant contributions. They modeled visual attention with a mixture of Gaussians at each frame. Without a priori knowledge, the model could recognize human behavior in the video, an ability that can be leveraged to improve the accuracy of baseline action classification.

Based on this previous research, we propose a new model, deep BLSTM-MDN. In the next section, we describe the principles and structure of deep BSLTM-MDN, and then introduce the training process. In the third section, we present the experimental process and compare different models in hit-or-miss classification performance. Finally, based on the proposed model, we generated the basketball trajectory. In the fourth section, we summarize the contribution and the limitations of this approach, and make suggestions for future research.

2. Deep BLSTM-MDN

In this section the motivation behind the BLSTM-MDN model is described. After the structure of model is elaborated, we explain how layers are stacked and the reshaped parameters that result. Finally, details about training and future directions are discussed.

2.1. Motivation

LSTM is a variant of RNN, which has the same type of input and output. But in contrast to RNN, LSTM has an input gate, a forget gate, and an output gate. Thus, it can control what needs to be preserved and what needs to be forgotten. This is why LSTM can retain information from a long time ago, whereas RNN cannot. BLSTM is derived from LSTM. Its main idea is that the output of each layer can process information from both forward units and backward units. A simple example is that when we want to understand the meaning of a word, the best way is to guess it according to the context.

In contrast to conventional BLSTM networks, which only give a point estimation for the target data, BLSTM-MDN outputs a subset of values based on their probability density function (PDF). The appropriate number of PDFs needs to be specified initially. Even though the weighted sum of PDFs can, in principle, represent any arbitrary distributions, too many PDFs will in fact lead to overfitting. Weights are calculated through maximum likelihood estimation.

Given the valuable features of both BLSTM and MDN, we offer the deep BLSTM and MDN model. It has the following advantages: 1) It does not require feature extraction; 2) Deep BLSTMs have strong spatiotemporal learning ability; 3) MDN can represent the true probability of the full probability distribution; 4) It can generate the basketball trajectory in three dimensions.

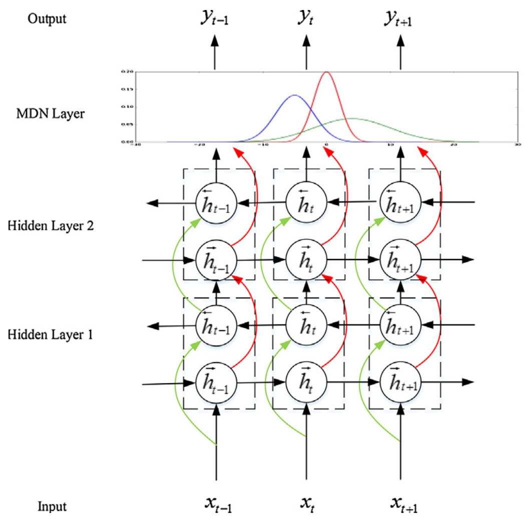

Fig. 1. Unfolded structure of Deep BLSTM and MDN.

2.2. Structure

In Fig. 1, the network has two hidden layers, and the output comprises three weighted MDNs. The horizontal arrows represent bidirectional flow in the temporal axis. The vertical arrows represent the one-way flow from input layer to hidden layer and from hidden layer to output layer. The red lines and green lines are forward and backward unit flows, respectively.

Let $S = \{(y^i, \mathbf{x}^i)\}_{i=1}^{N}$ represent the set of N samples. For the sample i, the input $\mathbf{x}^i$ has four features: trajectory in three dimensions, and a time clock. But the output y^i depends on different tasks. For the hit-and-miss classification task, y^i has a binary value of hit and miss. For the generation task, y^i is the evaluation of the next point $\mathbf{x}^{i+1}$.

LSTM adds three gates to avoid gradient vanishing and to improve memory over long periods of time. The input gate (i_t), forget gate (f_t), output gate (o_t), internal memory (c_t), and LSTM unit output (h_t) at time step t are computed as follows:

$$\begin{aligned} f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{hc}h_{t-1} + W_{xc}x_t + b_c) \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{1}$$

where $\sigma(\bullet)$ is the sigmoid activation function, and $\odot$ is elementwise multiplication.

For clarity, $LSTM(\bullet)$ is used to represent all LSTM's functions from Eq. (1). Therefore, a single BLSTM layer can be concatenated with a forward sequence and a backward sequence.

$$\begin{aligned} \overrightarrow{h}_t &= LSTM(x_t, \overrightarrow{h}_{t-1}) \\ \overleftarrow{h}_t &= LSTM(x_t, \overleftarrow{h}_{t+1}) \\ y_t &= g(W_{\overrightarrow{h}y} \overleftarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y) \end{aligned} \tag{2}$$

where $g(\bullet)$ represents the activation function, W_{mn} represents the weight from m to n, and b_n represents bias at layer n. Additionally, we choose rectified linear unit (ReLU) [25] as the activation function.

For the output layer, the Gaussian function is chosen as PDF, which can be defined as

$$P(y_t|N_t) = \sum_{c=1}^{C} \omega_t^c N(y_t|\mu_t^c, \sigma_t^c, \rho_t^c) \tag{3}$$

where y_t is the ground truth, C is the number of PDF, ω_t^c is the weight of the c^{th} PDF, and $N(\bullet)$ is the Gaussian function.

Because the probability distribution should be valid, the parameters in $N(\bullet)$ are normalized as follows:

$$\begin{aligned} \mu_t^c &= \tilde{\mu}_t^c \\ \omega_t^c &= \frac{\exp(\tilde{\omega}_t^c)}{\sum_{i=1}^{C} \exp(\tilde{\omega}_i^c)} \\ \sigma_t^c &= \exp(\tilde{\sigma}_t^c) \\ \rho_t^c &= \tanh(\tilde{\rho}_t^c) \end{aligned} \tag{4}$$

where $\tilde{\mu}_t^c$, $\tilde{\sigma}_t^c$, $\tilde{\omega}_t^c$ and $\tilde{\rho}_t^c$ are mean value of output, variance of output, weight of the PDF, and the correlation of the c^{th} Gaussian component, respectively.

2.3. Training

To make the output as close as possible to the ground truth, we need to maximize the probability likelihood:

$$L(\mathbf{x}) = \sum_{t=1}^{T} -\log\left(\sum_{c=1}^{C} \omega_t^c N(y_t|\mu_t^c, \sigma_t^c, \rho_t^c)\right) \tag{5}$$

where T is the sequence length. To calculate the loss function, we choose the Adam optimizer [24] function. The Adam optimizer can improve the traditional gradient by using momentum (the moving average of the parameters). To simplify the calculation, we assumed that the trajectory in the z-axis was uncorrelated with that in the x- and y-axes.

3. Experiment

The computer used to test the model had an i7 CPU with 8 GB RAM and an NVIDIA GTX 960 m GPU, which has 640 CUDA cores and 4 GB RAM. Both GPU and CPU were used, depending on the size of the neural network, which sometimes exceeded the available amount of memory on the graphics card during training.

3.1. Data pre-processing and tricks

SportVu is an optical tracking system that can record the spatial position of the ball and players on the court 25 times a second during a game. We used the NBA 2015–2016 season, a total of 631 games with 20,780 three-point shots, as data for the experiment. For the shots, most sequence lengths ranged from 30 to 70 points. To deal with the unequal lengths, the model disregards the shots that are less than 12 points in length, and it cuts off the others at 12 points. This reveals that the eliminated shots account for 1.04% only of all shots, which has a negligible effect on the final results. 12 points is a reasonable number, as a few points relative to the basket determine whether each shot will be a hit or a miss; additionally, it makes efficient use of the computing resources.

We define the tensor $[X, Y, Z, T]$ as input, where X refers to the length of the court, Y is the width of the court, Z is the height of the ball, and T is the game clock at the point of measurement. For the first (hit or miss) task, the output is a hit or miss on one shot. For the second (trajectory generation) task, the output is the next location of the ball over time in three dimensions. In the experiment, we set the basket as the center point, and transformed each point in sequence to a relative value. The dataset was divided into a training set and a test set based on Pareto principle.

When the model has been trained for several epochs, it begins to overfit. Therefore, we used an early stop to overcome the problem. This will stop the process if the present loss is lower than 90% of the mean value of the last 10 losses. In contrast to the trial-and-error method, we used grid search [26] to decrease the range of hyper-parameters, and then used the Python Hyperopt library [27] to find the best values. The Hyperopt library provides a parallel solution for model selection and parameter optimization in Python. Hyperopt is like a black box, where users only need to input an evaluation function and parameter space, and from these, they can obtain the best value within the space. In this paper, we chose the Tree of Parzen Estimators (TPE) algorithm as an optimization algorithm.

3.2. Hit or miss

In this section, we describe how we tested several models for the hit or miss classification task and analyzed the result. The hit rate for the whole dataset was 35.7%, which means that the two classes (hit and miss) were disproportionally represented. So the area under the curve (AUC) was a more appropriate tool for evaluating the different models' performance.

For GLM and GBM, extra information is required in the form of additional variables based on the physics of ball trajectories, such as the angle of the ball with respect to the rim, and the distance to the rim. We believe that the additional information can improve the classification accuracy of the models.

For the other models, we built two hidden layers. Specifically, we set three Gaussian functions as MDN for LSTM-MDN and BLSTM-MDN. The epoch is 300, and it loops 50 times, with Hyperopt set for the best AUC.

Table 1
AUC at different distances to the basket for each model.

Distance to basket	Previous models [14]			Our models			
	GLM	GBM	RNN	CNN	LSTM-MDN	BLSTM	BLSTM-MDN
2 feet	0.875	**0.942**	0.930	0.800	0.923	0.926	0.933
3 feet	0.807	0.902	0.913	0.796	0.918	0.924	**0.925**
4 feet	0.721	0.848	0.906	0.790	0.915	0.918	**0.922**
5 feet	0.659	0.796	0.880	0.763	0.887	0.890	**0.910**
6 feet	0.604	0.746	0.873	0.751	0.877	0.879	**0.903**
7 feet	0.583	0.742	0.841	0.748	0.848	0.863	**0.882**
8 feet	0.558	0.719	0.843	0.740	0.846	0.851	**0.869**

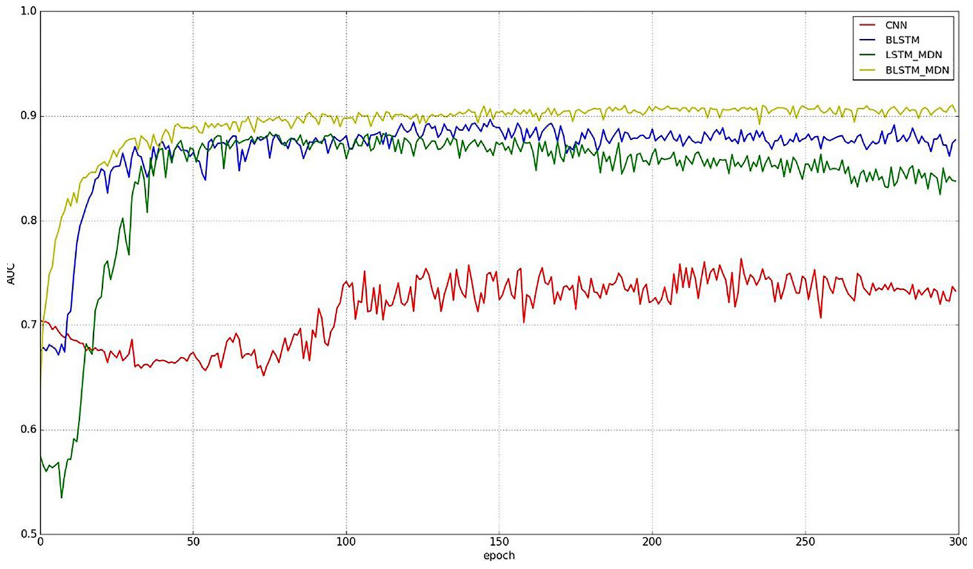

Fig. 2. AUC for models at 5 feet from the basket.

Table 2
Performance of deep learning models at 5 feet.

Model	AUC	Best epoch	Variable number	Spend time (average) per second
CNN	0.763	229	2532	300
LSTM-MDN	0.887	75	52378	2357
BLSTM	0.890	147	34434	1642
BLSTM-MDN	**0.910**	298	35994	1859

The best AUC is marked in bold (Table 1). Generally, the closer to the basket, the better is the AUC. For the same distance, deep learning models perform better than conventional models (GLM and GBM). However, convolutional neural network (CNN) performs no better than conventional models for this exercise, even though it performs well in many other fields. BLSTM-MDN gets the best AUC at 3–8 feet (Table 1).

To compare the performance of the models in more detail, we selected the distance of 5 feet to analyze the AUC for each epoch. In Fig. 2, CNN, BLSTM, LSTM-MDN, and BLSTM-MDN are drawn with red, blue, green, and yellow lines, respectively. Initially, BLSTM, LSTM-MDN, and BLSTM-MDN had the same upward trend, the AUCs increased sharply, and they started to oscillate from epoch 40 at close range. CNN had the same initial value as the others, but the AUC increased much more slowly. The best AUC was around epoch 100. Overall, BLSTM-MDN performed both well in the convergence rate and final AUC.

Table 2 shows the AUC, epoch, variable number, and time spent for each model. It should be noted that each model was re-run 50 times with Hyperopt, and the table shows the best AUC. The four models all have two hidden layers, and each layer has 64 units. CNN has the simplest structure, with only 2532 variables, and only 300 s-run cost per run with Hyperopt. The BLSTM model splits hidden units into forward LSTMs and backward LSTMs on average. In other words, each LSTM has 32 units. Therefore, the number of variables in BLSTM-MDN has fewer variables than LSTM-MDN. This means that running BLSTM-MDN costs less than running LSTM-MDN does (1859 vs. 2357 variables).

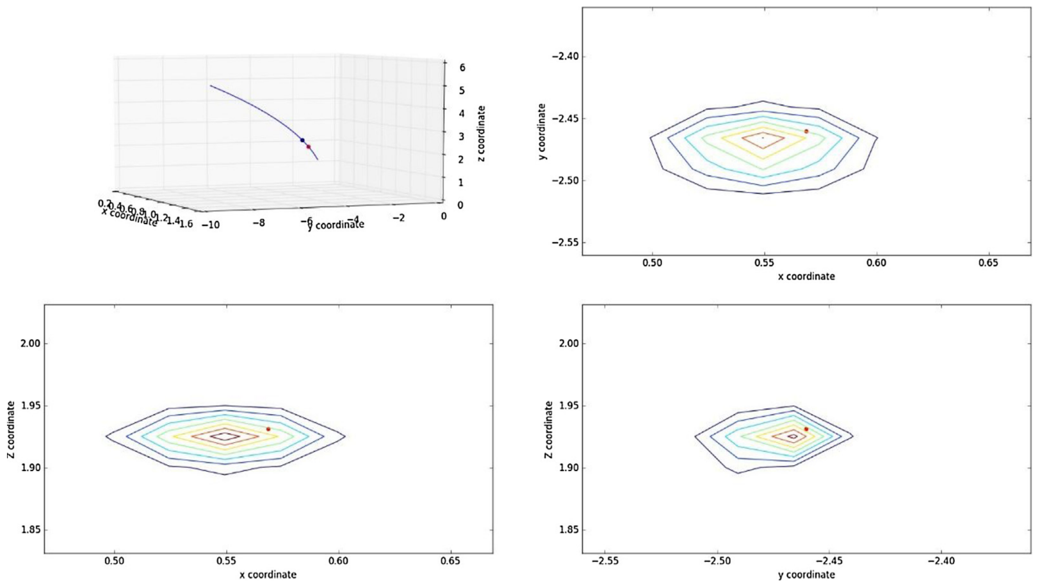

Fig. 3. Distribution probability of the 10th point given the 9th point.

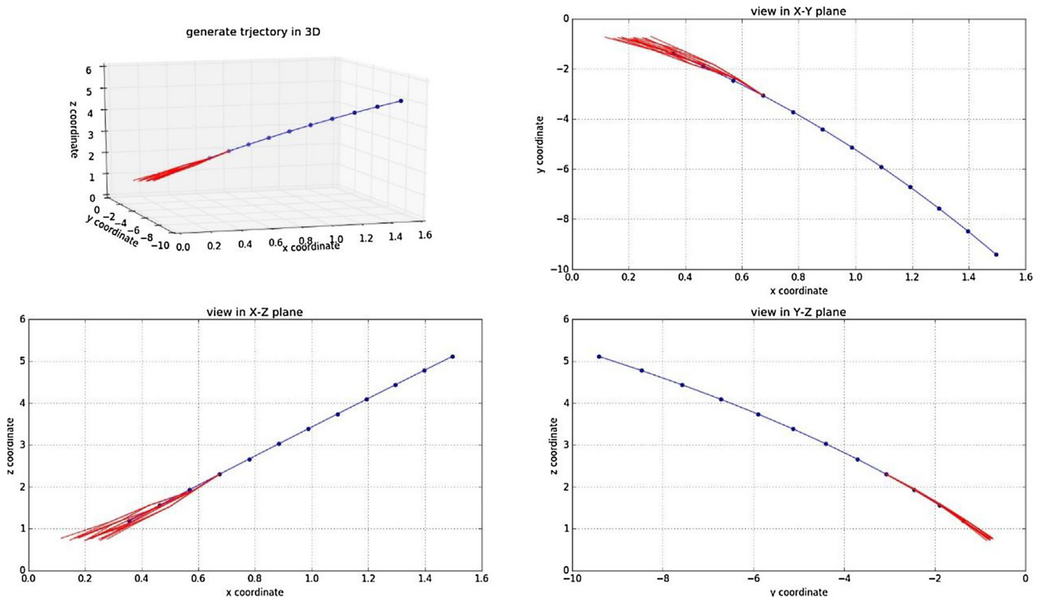

Fig. 4. Generated trajectories with given time stamps.

3.3. *Trajectory generation*

Using BLSTM-DMN, we generated the trajectory of a flying basketball. The model has the same hyper-parameters as were used for Section 3.2. Thanks to MDN, it can obtain the distribution probability of each point. Therefore, it is easy to know the next point using a roulette algorithm.

The top left subplot in Fig. 3 shows a real 3D trajectory, which is composed of 12 discrete points. The red point and blue point denote the 9th and 10th points respectively. The other subplots are 10th point in the horizontal plane, vertical plane and width plane respectively. The circles around mean the probabilistic contour lines of the 10th point. And probability increase along with the color ranging from blue to red. It is obviously that the real position (red point) is almost locates in the center of each circle, which means BLSTM-MDN gets very high accuracy prediction.

In Fig. 4, the blue lines are real trajectories, and the red ones are generated. Again, we picked the 9th point as the time stamp. Then, for the subsequent positions (the 10th, 11th, and 12th), the model generated two predicted points. In sum, the model had eight (2^3) trajectories. It can be seen that the generated trajectories are very similar to the real one.

4. Conclusion

We introduced a BLSTM-MDN model for three-point shot prediction. During model training, the Python library Hyperopt was applied for hyper-parameter self-optimization. Compared to the previous models, such as the conventional CNN and BLSTM models, the proposed one performed better in both convergence rate and prediction accuracy.

Even though the basketball trajectory prediction is a toy problem, we suggest that BLSTM-MDN can produce correct results in many other areas, such as UAV route planning, human activity recognition, and stock market prediction. There are also many factors that need to be further considered. For example, the model should be able to take into account player cooperation and defense when predicting NBA player positions. Time series prediction is a complicated but meaningful research topic. We aim to improve our model by taking into account more features in the future.

References

[1] S. Ji, M. Yang, K. Yu, W. Xu, 3D convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 221–231.
[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social LSTM: human trajectory prediction in crowded spaces, IEEE, 2016 IEEE Conference on Computer Vision and Pattern Recognition (2016) 961–971.
[3] K. Ohno, T. Nomura, S. Tadokoro, Real-time robot trajectory estimation and 3D map construction using 3D camera, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2006), 2006, pp. 5279–5285.
[4] W. Chen, T. Qu, Y. Zhou, K. Weng, Door recognition and deep learning algorithm for visual based robot navigation, IEEE, IEEE International Conference on Robotics and Biomimetics (2014) 1793–1798.
[5] P. Chandler, S. Rasmussen, M. Pachter, UAV cooperative path planning, (August), AIAA-2000-4370, AIAA Conference on Guidance, Navigation and Control (2000).
[6] J. Tisdale, Z.K.Z. Kim, J. Hedrick, Autonomous UAV path planning and estimation, IEEE Robot. Autom. Mag. 16 (2) (2009) 35–42.
[7] G.R. Hamilton, C. Reinschmidt, Optimal trajectory for the basketball free throw, J. Sport Sci. 15 (5) (1997) 491–504.
[8] M. Beuoy. Introducing ShArc: Shot arc analysis. In http://www.inpredictable.com/2015/05/introducing- sharc-shot-arc-analysis.html.
[9] C.M. Tran, L.M. Silverberg, Optimal release conditions for the free throw in men's basketball, J. Sport Sci. 26 (11) (2008) 1147–1155.
[10] K. Wang, R. Zemel, Classifying NBA offensive plays using neural networks, in: MIT Sloan Sports Analytics Conference, 2016, pp. 1–9.
[11] R.C. Shah, R. Romijnders, Applying depp learning to basketball trajectories, in: KDD '16, 2016, pp. 4.
[12] S. Zheng, Y. Yue, J. Hobbs, Generating long-term trajectories using deep hierarchical networks, Nips (Nips) (2016) 1543–1551.
[13] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Netw. 18 (5–6) (2005) 602.
[14] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, B. Schuller, Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks, IEEE 219 (6) (2014) 2164–2168.
[15] A. Graves, N. Jaitly, A.R. Mohamed, Hybrid speech recognition with deep bidirectional LSTM, in: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings, 2013, pp. 273–278.
[16] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, G. Rigoll, Bidirectional LSTM networks for context-sensitive keyword detection in a cognitive virtual agent framework, Cognit. Comput. 2 (3) (2010) 180–190.
[17] Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, in: Computer Science, 2015.
[18] N. Sankaran, C.V. Jawahar, Recognition of printed devanagari text using BLSTM neural network, in: Proceedings - International Conference on Pattern Recognition, 2012, pp. 322–325.
[19] M. Wöllmer, F. Eyben, B. Schuller, Y. Sun, T. Moosmayr, N. Nguyen-Thien, Robust in-car spelling recognition - A tandem BLSTM-HMM approach, in: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2009, pp. 2507–2510.
[20] C.M. Bishop, Mixture density networks, IEEE Computer Society, Neural Networks, IEEE - INNS - ENNS International Joint Conference (1994) 4455.
[21] A. Graves, Generating sequences with recurrent neural networks, Tech. Rep. (2013) 1–43.
[22] H. Zen, A. Senior, Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis, Institute of Electrical and Electronics Engineers Inc, ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (2014) 3844–3848.
[23] L. Bazzani, H. Larochelle, L. Torresani, Recurrent mixture density network for spatiotemporal visual attention, Arxiv (2016) 1–17.
[24] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.
[25] Ian Goodfellow, A.C. Yoshua Bengio, Deep Learning. Deep Learning, 2016.
[26] S.M. LaValle, M.S. Branicky, On the relationship between classical grid search and probabilistic roadmaps, Springer Tracts in Advanced Robotics 7 (2004) 59–75.
[27] J. Bergstra, D. Yamins, D.D. Cox, Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms, in: 12th Python in Science Conf. (Scipy 2013), (Scipy), 2013, pp. 13–20.