Recurrent Neural Networks for Modeling Motion Capture Data

Mir Khan, Heikki Huttunen, Olli Suominen and Atanas Gotchev
Laboratory of Signal Processing
Tampere University of Technology
Tampere, Finland
Email: mirabdul@tut.fi, heikki.huttunen@tut.fi, olli.j.suominen@tut.fi, atanas.gotchev@tut.fi

***Abstract*—Recurrent Neural Networks have recently received attention for human animation applications including motion synthesis; however, previous works did not provide any quantitative approaches for evaluating the quality of the motion generated by these models. In this paper, we use three different recurrent neural network architectures for synthesizing human motion for a detailed skeleton with 64 joints. We introduce a novel motion quality metric for quantitatively evaluating the realism of the synthesized motion. We use this metric, among others, to compare the motion generated by the three network architectures and empirically study the impact of the network's complexity on the quality of the motion.**

Keywords: motion capture, recurrent neural network, generative model, long short-term memory

1. Introduction

Producing natural animations is important for many entertainment industries. Motion capture is one commonly used method for reproducing convincing animations by recording motion played by human actors. Motion capture data is used to animate 3D characters by mapping the same motion onto the virtual character. While motion capture offers many advantages, it suffers from several drawbacks. This includes the difficulty of reusing motion data for different scenarios. Therefore, it would be of great interest to the industry to generate animations through alternative approaches that do not rely heavily on human actors and manual processing.

One alternative approach to motion capture and manual animation is simulation-based methods for generating natural motion, which is an area closely related to robotics. The principle in these methods is to develop control strategies for humanoids and imaginary creatures in a simulated environment through the use of reinforcement learning and optimization methods such as genetic algorithms. The control strategies are then optimized with respect to a criteria (often called a *fitness function*) such as total distance traveled. An early example of these methods was introduced in [24], where virtual creatures with random morphologies develop relatively optimal control strategies such as swimming, running, jumping, and crawling depending on their environment, their own physical characteristics, and the defined fitness function. Similar techniques have been used to produce more controlled behavior such as bipedal gait animations in a simulated environment [22]. Using bio-mechanical constraints have been shown to produce even more convincing and natural-looking animations for bipedal virtual creatures [10]. The main drawback of these techniques is that the generated motion is not often exactly what the animator may desire.

Techniques based on neural networks with multiple layers are known as deep learning. Neural networks, which are crude models of the information-processing mechanism of the biological brain, are extremely versatile machine learning tools with an ever-growing range of applications. Neural Networks techniques previously have been successfully applied for tasks such as image classification [17][1], face-recognition [18][21], audio classification [16], and speech recognition [9].

Deep learning techniques for synthesizing 3D human motion has drawn attention recently. An approach has been proposed in [14][15] to learn a space of valid human poses (called the motion manifold) using a convolutional-autoencoder network architecture. The framework can be used for novel motion synthesis, motion interpolation, and error-correction. A method that uses conditional Restricted Boltzmann Machines (cRBM) have been used to synthesize human gait animations [25].

Recurrent Neural Networks (RNNs) are a variation of standard neural networks with feedback loops where the inputs of previous time steps determine the output at future and present time steps. This temporal attribute of RNNs makes them well-suited for analyzing time-series data. RNNs have bee successfully used for automatic music composition and analysis [3][20], image synthesis [11], and handwriting generation [8].

A variant of RNNs, called Long Short-Term Memory (LSTM), has been proposed for solving several problems standard RNNs suffer from, among these are learning long-term dependencies which LSTM networks are capable of for up to 1000 discrete time steps into the future [12]. More notably, LSTM networks solve the phenomena of vanishing gradient [13], which is a problem that arises in very deep neural networks, RNNs in particular. The main difference between a standard RNN and an LSTM network is the inclusion of memory states and gates. LSTM networks have

been previously used successfully for human motion synthesis by training the network on a large dataset of human motion capture data with a variety of motion categories [4]. A similar approach was presented in [2] that uses a data set which consists of recordings of a dancer, which allows the network to generate dancing animations in a similar style.

Our work is most closely related to [2] and [4], but our focus is on motion synthesis for a detailed skeleton with 64 joints. More importantly, we study and compare the quality of the synthesized motion and present a quantitative evaluation of the motion generated by three different LSTM architectures. The purpose of this study is to determine the impact of the complexity of the model in terms of layers on the quality of the generated motion, and therefore provide a rigorous and quantitative justification for selecting a model for applications of motion synthesis.

2. Methods

2.1. Data preparation

Our data set consists of 5 hours of motion capture data recorded at 120 frames per second. Various motion categories are present in the data set such as walking, running, and dancing. The skeleton model of the motion capture data set is a 64-joint human skeleton model, which includes details such as hand fingers. The motion capture data is stored as a time-series of joint rotation angles in the files, but we transform this data to a time series of joint position coordinates. Thus, each time-step of the sequence is represented by $3 \times 64 = 192$-dimensional vector consisting of the concatenation of the x, y and z position coordinates of each joint. We construct the training set by extracting a fixed-length sequence using a 200-frame temporal window sliding at a temporal step size of 100 frames, resulting in 50% overlap between consecutive training samples. Each input sample $\mathbf{X} \in \mathbb{R}^{192\times 200}$ is a motion sequence given as

$$\mathbf{X} = \begin{bmatrix} x_{hips}^{(1)} & x_{hips}^{(2)} & x_{hips}^{(3)} & \dots & x_{hips}^{(200)} \\ y_{hips}^{(1)} & y_{hips}^{(2)} & y_{hips}^{(3)} & \dots & y_{hips}^{(200)} \\ z_{hips}^{(1)} & z_{hips}^{(2)} & z_{hips}^{(3)} & \dots & z_{hips}^{(200)} \\ x_{spine}^{(1)} & x_{spine}^{(2)} & x_{spine}^{(3)} & \dots & x_{spine}^{(200)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{foot}^{(1)} & x_{foot}^{(2)} & x_{foot}^{(3)} & \dots & x_{foot}^{(200)} \\ y_{foot}^{(1)} & y_{foot}^{(2)} & y_{foot}^{(3)} & \dots & y_{foot}^{(200)} \\ z_{foot}^{(1)} & z_{foot}^{(2)} & z_{foot}^{(3)} & \dots & z_{foot}^{(200)} \end{bmatrix}, \quad (1)$$

where the horizontal dimension corresponds to the number of frames in the sequence, and the vertical dimension corresponds to the degrees of freedom of all 64 joints, i.e., the joint position coordinates. The subscript denotes the joint name according to the hierarchy defined in the motion data files. In our case, the first joint from which all other joints extend is the hips joint. The superscript denotes the frame number and it corresponds to the horizontal axis of the matrix.

Each target output sample $\mathbf{y} \in \mathbb{R}^{192\times 1}$ in the training set is a vector, representing a single frame of motion, and it is the frame that follows the input sample $\mathbf{X}$ extracted from the same original motion file. Thus, a single training sample is given as the pair $(\mathbf{X}, \mathbf{y})$. In essence, the network is trained to complete the motion sequence it is given one frame at a time. These data preparation methods result in a final data set of nearly $15,000$ samples, of which 500 are reserved for validation and another 500 for testing and motion synthesis

To generate a motion sequence, first, the input $\mathbf{X}$ is simply fed to the network and the output is computed. Then, this output is concatenated with the previous input sequence $\mathbf{X}$ along the temporal axis, and it is shifted one frame forward, such that $\mathbf{X}$ will then contain the previous output $\mathbf{y}$ as its last frame while maintaining its length of 200 frames. This process is repeated for as many time steps as desired by feeding the newly constructed input sequence $\mathbf{X}$ to the network again to generate motion sequences of arbitrary length.

2.2. Network architecture

We use three network architectures which we will refer to as LSTM1, LSTM2, and LSTM3, where the post-fix denotes the number of layers in the network, each with 1000 LSTM units. Previous approaches in applying LSTM for motion capture used slightly more complex network architectures such as an encoder-decoder architecture in [4] and a Mixture-Density Network layer in [2]. For the purpose of analysis and comparison, we decided to keep the models simple so that our analysis captures the essential properties of these networks.

The weights for all layers are initialized according to [23] by generating an orthogonal matrix with a gain factor of 1.0. The weights matrix for the recurrent kernel is initialized by sampling a truncated normal distribution; this is known as a *glorot normal initialization* [6]. The bias values are all initialized to zeros. A linear activation function is used at the output layer of the network. All hidden layers use the hyperbolic tangent as their activation functions, and a hard sigmoid as the activation function for the recurrent step. The Mean-Squared Error (MSE) was used as the loss function and RMSprop [26] was chosen as an optimizer with the initial learning rate of 0.001. The network was trained with approximately $14,000$ samples using a mini-batch size of 32 samples. Each network was trained until the performance of the network (in the MSE sense) stopped showing any improvement.

3. Results

All three trained networks can generate novel motion sequences that complete the given input sequence, while

maintaining inter-joint relationships to varying degrees. Examples of the generated motions are included in the supplementary materials. From visual observation alone, the LSTM network with 3 layers maintains inter-joint relationships for the longest number of time steps and shows better motion variety. However, in order to make this analysis more rigorous, we perform quantitative evaluation of the quality of the motion.

Quantifying the quality of the generated motions is difficult due to the strongly qualitative nature of human motion. One way to measure the *correctness* of motion is by evaluating the network's understanding of inter-joint relationships. This is done by computing the distance of each joint from its parent and taking the difference from the original distance of this link. We can further average this result over all the joints in the skeleton and take the absolute value to obtain an average of this measurement over all joints. We will denote this metric by the name Inter-Joint Variation (IJV). For a single frame, the IJV averaged over all joints is given by the expression

$$\text{IJV} = \frac{1}{K}\sum_{i=0}^{K-1}\left|\|\mathbf{s}_i - \mathbf{s}_{p(i)}\|_2 - \|\mathbf{v}_i - \mathbf{v}_{p(i)}\|_2\right|, \qquad (2)$$

where the vectors $\mathbf{s}_i, \mathbf{s}_{p(i)} \in \mathbb{R}^3$ are the position coordinates of the joint i and joint i's parent respectively. The total number of joints is denoted by K (this is 64 in our case). The function $p(i)$ is a discrete-valued function that returns the identifying number for the parent of joint i. Similarly, $\mathbf{v}_i$ and $\mathbf{v}_{p(i)}$ are arbitrary position coordinates for the same joints and for the same skeleton (typically from the original recorded motion file) which can serve as the ground truth. It should be noted that the actual values of $\mathbf{v}_i$ and $\mathbf{v}_{p(i)}$ don't matter and that it is the length of the link connecting these joints is what is important for this measurement. In our case, as ground truth, we simply use the inter-joint distances in the first frame in the sequence of interest. Figure 1 provides a visualization of this measurement for each model at each frame, averaged over all samples. It can be seen that the IJV values for LSTM3 grows slowest in comparison with the other two models.

Measuring the joint relationships on their own may not always be an accurate quantification of the quality of motion, since it is possible that the network outputs sequences with little to no movement, and yet small IJV values. Therefore, we use an additional metric that measures motion *energy*, which for a sample $\mathbf{F} \in \mathbb{R}^{m\times n}$ can be computed as follows:

$$E = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=2}^{n}(F_{i,j} - F_{i,j-1})^2 \qquad (3)$$

Here, $F_{i,j}$ is the element at row i and column j, m is the number of degrees-of-freedom of all joints (192 in our case), and n is the number of frames in the sequence. Consider the two graphs shown in Figure 3 illustrating the performance of each network (in the IJV sense) as we restrict the samples to a subset with mean energy exceeding the threshold on the horizontal axis (top). The graph at the bottom shows the

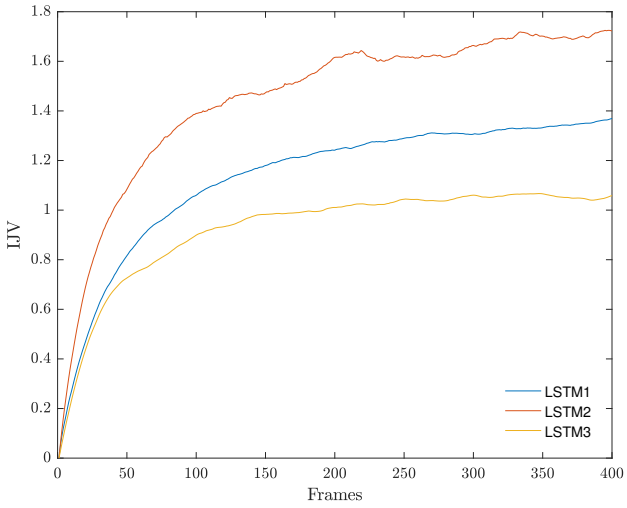


Figure 1. IJV measurements for each model, shown at each frame, averaged over all samples.

TABLE 1. ANALYSIS RESULTS OF THE MOTION SEQUENCES GENERATED BY THE THREE MODELS, AVERAGED OVER ALL 500 SAMPLES AND ALL 400 FRAMES PER SAMPLE.

	Avg. Measurements Over All Samples			
Joint	LSTM1	LSTM2	LSTM3	Ground Truth
IJV_{avg}	1.150 cm	1.472 cm	**0.939 cm**	0.0 cm
MID_{avg}	**6.675 cm**	13.643 cm	6.975 cm	5.608 cm
Energy_{avg}	1.128	6.306	1.028	0.409
Energy_{std}	1.247	15.156	1.387	1.319

distribution of these energies, visualized in the same manner by representing each point in the graph as the measurements on the restricted subset of samples exceeding a mean energy threshold. The purpose of this analysis is to study how the network's understanding of joint relationships changes in relation with the energy level. Additionally, it illustrates the proportion of the samples for which IJV measurements are made in the top graph. One can also think of the energy measurement as a measure of the average *amount* of motion in a sequence, such that motion sequences with faster movements will have more energy than motions with slower movements.

A straight-forward metric is the euclidean distance between consecutive frames, which can be a reasonable approach to quantify the motion similarity between consecutive frames. We compute this result using the equation

$$MID = \frac{1}{n}\sum_{j=2}^{n}\left\|(\mathbf{f}_j - \mathbf{f}_{j-1})\right\|_2, \qquad (4)$$

where we denote by $\mathbf{f}_j \in \mathbb{R}^{192\times 1}$ the jth frame in the sequence. This result can then be averaged over all samples. Table 1 shows these measurements for all samples for each model, in order to provide a general comparison of the

quality of the motion generated by each network. We convert the IJV results to the physical unit of centimeters in order to provide an intuitive sense of the errors. IJV and MID are calculated as shown before and averaged over all samples. On the third row, Energy$_{avg}$ shows the average energy as calculated by equation 3. The last row, Energy$_{std}$, shows the standard deviation of the energy of all samples. The fourth column shows these measurements for the 500 samples from the data set reserved for motion synthesis. It can be argued that LSTM3 shows the best capacity for novel and realistic motion synthesis and maintaining inter-joint relationships.

Figure 2 shows the IJV values for each joint averaged across all models and all samples. This illustration aims to highlights the joints which seem to be most problematic for our models to learn. One possible explanation for the severity of the errors at the fingers is that finger motions can be very complex, while leg joints, for example, remain mostly similar over the data set.

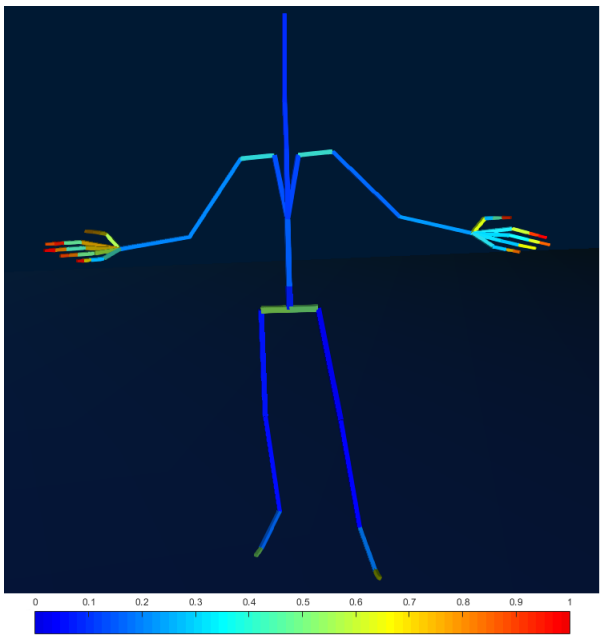
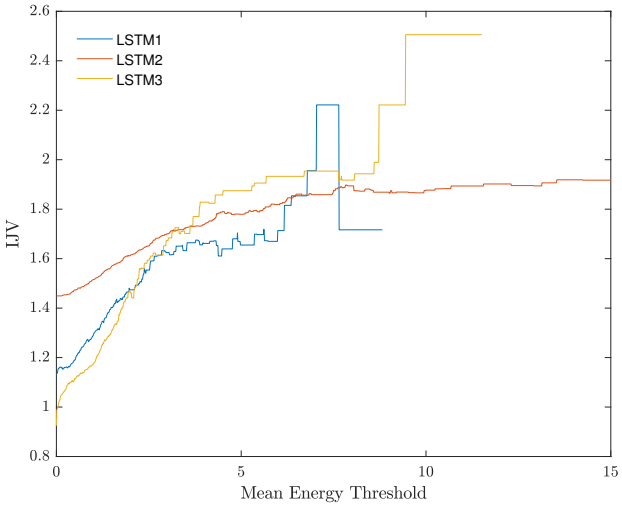


Figure 2. IJV values averaged across all models for each joint.



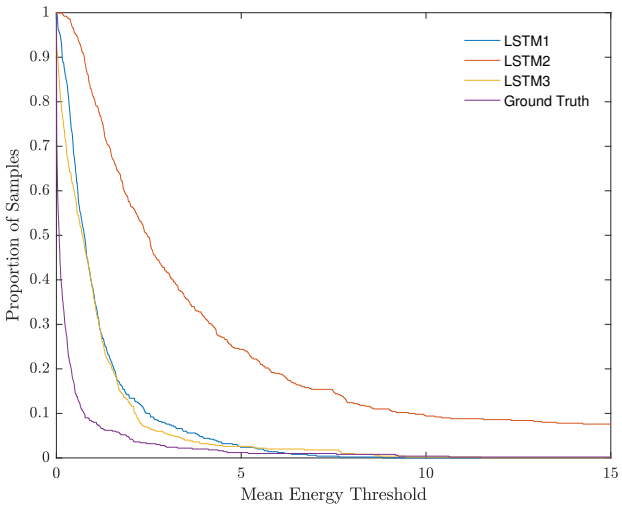


Figure 3. IJV at each frame averaged over all samples for each model (top) and its relationship with the energy measurements shown in the bottom graph.

4. Conclusion

We have studied three different LSTM models for human motion synthesis for a detailed skeleton with 64 joints. Analysis shows that the three-layered LSTM architecture, with 1000 nodes in each layer, produces motions that are most realistic when compared to the single-layer and the 2-layer LSTM networks. The LSTM3 network can maintain good inter-joint relationships which extends up to 400 frames. Analysis has also shown that, overall, these models can reasonably accurately maintain joint relations for the spine, neck and legs, but they faces some difficulty in maintaining these relations for the more detailed segments of the skeleton such as hand fingers.

We can also observe that the two-layered LSTM architecture is an exceptionally bad model for human motion synthesis, even more so than the single-layered model. This may be due to the fact that the two-layer model is not robust against over-learning (like LSTM1), while it still does not possess the expressive power that the LSTM3 has. However, more research is needed in order to confirm this observation and to consider even deeper models than the three-layer model.

In the future, we aim to make use of a larger dataset and of specific motion types. Such a data set may allow more control over the motion categories generated. More specifically, we wish to build on the work in [2], which uses a data set of recordings of a dancer, and extend it to other

motion categories such as walking, sprinting, crawling, or even different dancing styles.

Additionally, we wish to examine the effect of larger models in terms of layers and number of neurons in each layer on maintaining joint relations. We expect that deeper and more complex models will show better capacity for maintaining detailed joint relations such as hand fingers and feet.

Finally, we plan to investigate the impact of semantically-rooted regularization techniques. The intuitive motivation behind this is to allow incorrect outputs (in the MSE sense) of the network that honor inter-joint relations to have less of an impact on the direction of the gradient while training. This could however, in theory, result in a worst case scenario where the network always outputs the same pose, but further studies are needed.

5. Supplementary Material

Video links showing the motion sequences generated by each model. Sequences with the blue skeleton are the input sequences feed to the network. Motion sequences in green are generated by the network.

- LSTM1
- LSTM2
- LSTM3

Acknowledgments

This work was supported by the research infrastructure of Center for Immersive Visual Technologies CIVIT, Tampere University of Technology. The training and testing data was collected and provided by Keho Interactive Oy.

References

[1] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on 2012.

[2] Crnkovic-Friis L, Crnkovic-Friis L. Generative Choreography using Deep Learning. arXiv preprint arXiv:1605.06921. 2016.

[3] Eck D, Schmidhuber J. A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale. 2002.

[4] Fragkiadaki K, Levine S, Felsen P, Malik J. Recurrent network models for human dynamics. In Proceedings of the IEEE International Conference on Computer Vision 2015.

[5] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural computation. 2000.

[6] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In Aistats 2010 (Vol. 9, pp. 249-256).

[7] Glorot X, Bordes A, Bengio Y. Deep Sparse Rectifier Neural Networks. In Aistats 2011 (Vol. 15, No. 106, p. 275).

[8] Graves A. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850. 2013.

[9] Graves A, Jaitly N. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In ICML 2014 (Vol. 14, pp. 1764-1772).

[10] Geijtenbeek T, van de Panne M, van der Stappen AF. Flexible muscle-based locomotion for bipedal creatures. ACM Transactions on Graphics (TOG). 2013;32(6):206.

[11] Gregor K, Danihelka I, Graves A, Rezende DJ, Wierstra D. DRAW: A recurrent neural network for image generation. arXiv preprint arXiv:1502.04623. 2015.

[12] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997 Nov 15;9(8):1735-80.

[13] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1998;6(02):107-16.

[14] Holden D, Saito J, Komura T, Joyce T. Learning motion manifolds with convolutional autoencoders. In SIGGRAPH Asia 2015 Technical Briefs 2015 (p. 18).

[15] Holden D, Saito J, Komura T. A deep learning framework for character motion synthesis and editing. ACM Transactions on Graphics (TOG). 2016;35(4):138.

[16] Kanda N, Takeda R, Obuchi Y. Elastic spectral distortion for low resource speech recognition with deep neural networks. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on 2013 (pp. 309-314).

[17] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems 2012 (pp. 1097-1105).

[18] Lawrence S, Giles CL, Tsoi AC, Back AD. Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks. 1997;8(1):98-113.

[19] Le QV, Jaitly N, Hinton GE. A simple way to initialize recurrent networks of rectified linear units. arXiv preprint arXiv:1504.00941. 2015.

[20] Nayebi A, Vitelli M. GRUV: Algorithmic Music Generation using Recurrent Neural Networks. 2015.

[21] Parkhi OM, Vedaldi A, Zisserman A. Deep Face Recognition. In BMVC 2015 (Vol. 1, No. 3, p. 6).

[22] Reil T, Husbands P. Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Transactions on Evolutionary Computation. 2002;6(2):159-68.

[23] Saxe AM, McClelland JL, Ganguli S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120. 2013.

[24] Sims K. Evolving virtual creatures. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques 1994 (pp. 15-22).

[25] Taylor GW, Hinton GE. Factored conditional restricted Boltzmann machines for modeling motion style. In Proceedings of the 26th annual international conference on machine learning 2009 (pp. 1025-1032).

[26] Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning. 2012;4(2).