人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-046-08 (11/09)

Sequential Deep Learning for Dancing Motion Generation

Nelson Yalta[*1], **Tetsuya Ogata**[*1], **Kazuhiro Nakadai**[*2]
Waseda University[*1], Honda Research Institute Japan Co., Ltd.[*2]
{nelson.yalta@ruri., ogata@}waseda.jp, nakadai@jp.honda-ri.com

Abstract

In recently years, robots have invaded our life in many activities such as manufacture process or social activities as dance. In dancing, robots have shown a good performance following the rhythm of a music using beat-tracking algorithm. In this work, we show an implementation of a deep learning based on sequential learning model for dancing robots. The model is trained without middle states mixing audio information and captured motion from a person, and it can substitute a beat-tracking algorithm. The model generates quasi-realistic dance pattern motion without constraints from the music information and its start position, and following the rhythm beat from a multiple sound source music.

1 Introduction

During last decades, robots have been involved and become part of humans everyday life, possible to perform dangerous tasks instead of humans, or appearing in the media. As robots becomes part of humans everyday life, the should interact constantly with humans. Thus in order to improve this interaction, dance presents as a bound between humans and robots [Oliveira 2015], and for dance performing they should be able to listen with its own ears live sounds. While robots perform a dance, they follow the rhythm on a real-environment robustly responding to music, and generating continuous movements. Some techniques which allows robots to dance, are based on beat-tracking implemented for real time execution. Beat tracking systems are implemented to track the beat timing of music pieces and synchronize movements of a dancing robot with it. Real-time application [Oliveira 2015], allows to track the beat timing, but it also overcome with noise problems and obtain the desired audio information from multiple sound sources, implementing a robust general framework. Beat tracking systems allow robots to perform dance, but also to play instruments [Itohara 2012]. Through a multimodal system, which uses audio and video information, allows robots to play an instrument e.g. a guitar. Also, a method which can generate automatic choreographies by music content analysis was introduced at [Fukayama 2015]. They proposed a probabilistic framework which generate automatic choreography without constraints, and which can satisfy both music content and motion connectivity. These implementations synchronize with the beat timing, a motion that is previously generated and stacked in a database. In this paper, we implemented a system based on deep learning to generate motion positions for dancing robots, directly from an audio information. Deep learning (DL) based models called deep neural networks (DNN) and deep convolutional neural networks (DCNN) have shown successful performance on many signal processing fields such as computer vision [Krizhevsky 2012, Clevert 2015], speech recognition [Sainath 2015a], and natural language processing [Sutskever 2014, Venugopalan 2015]. DL based models implemented for natural language processing task has shown a remarkable performance, demonstrating that can translate correctly long sentences [Sutskever 2014] on translation task, using deep Long-short term memory (LSTM) models. Or also being able to extract information converting video to text [Venugopalan 2015], mixing a DCNN with a deep LSTM (DLSTM) for the task. Generating smooth realistic dance motion shows as a constraint for implementations which only use the audio information. It should be able to track the beat timing, and at a same time separate the desired sound information from a multiple audio input. Thus, we focus on implementing a DL model taking advantage of the sequence-to-sequence model introduced at [Venugopalan 2015]. Implementing a model for video-to-text requires a previous pre-trained DCNN model, which increase the training time i.e. learning cost. For implementing, we remark that is possible to replace a beat tracking system with an end-to-end trained DL model, which uses the audio information as input and mixing with the current position, generates the next position.

The DL model generates a quasi-realistic motion pattern without constraints, separating from a multiple sound source audio, the beat timing performed by drums.

2 Proposed Method

DL models have shown an outstanding achievement in different tasks which have been involved, and sound signal processing task such as speech recognition, they showed outstanding performance[Abdel-hamid 2012, Sainath 2015b]. DL models trained without middle states i.e. end-to-end training, has become the state-of-art in multiple signal processing areas. They can process information from the same field, but also in multimodal fields such as video-to-text task. Thus, we replaced a beat tracking system that synchronize the beat timing and a stacked motion pattern, with a DL model trained end-to-end. The motion pattern of the task is a combined vector of the normalized spatial position of a dancer, and the joint rotations expressed in quaternions obtained from a kinect device. A mixed model which a deep LSTM network is connected after the plain DCNN but not connecting it in a sequential mode, generated random movements from the audio information. However, the motion patterns are not linked with each other and the generated motions are rough. Then, we replaced it with a similar structure implemented at [Venugopalan 2015], and modified the activation functions. Finally, we trained it end-to-end using the audio as input and a sequential motion pattern as the target.

2.1 Deep Learning

Recently, factors such as the development of improved optimization algorithms, the availability of open-source DL libraries and so on, have made the employment of DL possible for classification or recognition problems. DL based models as DNN and DCNN approaches, has shown that can surpass even humans performance. Optimization algorithm which become part of the model such as batch normalization, or training algorithm such as ADAM, or adding noise to the gradients has allowed to improve the performance of DL task on different signal processing fields. Batch normalization introduced at [Ioffe 2015] is a mechanism that allows training with higher learning rates without fast overfitting risk and also can accelerate the training as makes possible to train deeper models. Adaptive moment estimation i.e. ADAM optimization was introduced at [Kingma 2014]. This optimization method allows to implement a robust optimization with a little memory requirement. As ADAM is an algorithm for first-order gradient-based optimization, this method is straightforward to implement and can be used on machine learning problems where it is used large datasets or with high-dimensional parameters models. Adding gradient noise in very deep models [Neelakantan 2016] improves its learning, but it helps generalization and training on complicated neural networks, as it also improves the learning from a poor initialization with almost a zero computational cost. Open-source DL frameworks has allowed researchers to use DL on different tasks. DL frameworks such as Chainer [Tokui 2015], allows to implement flexible DCNN and trained it with multiple optimization algorithm such as Batch Normalization, ADAM optimization or adding noise to the gradient during the training.

2.2 Nonlinearities Activation Functions

Recently researches has shown that the performance of a DCNN also depends on the type of nonlinearity function used in the model. Novel nonlinear activation functions (Figure 1) has been introduced [Clevert 2015, He 2015], improving the accuracy and performance of DCNN. Rectifier neurons e.g. Rectified Linear Unit (ReLU) nonlinearity have shown successfully performance on vision computer [Krizhevsky 2012], but also for sound tasks. The ReLU activation function is defined by:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}, \tag{1}$$

ReLU activation outputs are non-negative, thus the mean activation is larger than zero, and because of not be zero-centered it can speed down the learning. Then, a nonlinearity called Parametric Rectified Linear Unit (PReLU) was introduced at [He 2015], which thus the implementation of a negative part activation it can speed up the learning. Later, Exponential Linear Unit (ELU) introduced at [Clevert 2015], was showed as a non-linearity that can improve learning characteristics compared to other linear activation functions. ELU defined as:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ a(exp(x) - 1) & \text{if } x < 0 \end{cases}, \tag{2}$$

Where a is a fixed value, and controls the saturation for negative inputs, and *exp* is the exponential function of e. It was showed that networks implemented with ELU activations can speed up learning as bring robust generalization performance compared to ReLUs and PReLUs.

2.3 Sequential Learning

Sequential Learning models i.e. sequence-to-sequence (SEQ2SEQ) model (Figure 2) has been applied for

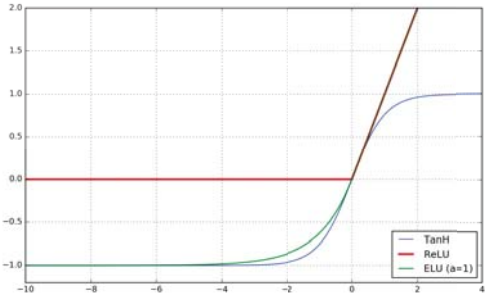

Figure 1: Nonlinearities Activation Functions

natural language processing task such as foreign language translation task [Sutskever 2014] or video-to-text task [Venugopalan 2015], and speech recognition [Graves 2014]. DL models has shown remarkable performance in different task such as speech recognition, where a DNN with Convolutional Neural Networks (CNN), LSTM and fully connected layers has better performance than LSTM models to perform the task [Sainath 2015a]. However, sequential problems such as speech recognition or machine translation are best expressed on sequence with unknown length, and DNN could present limitations due to the fixed dimensionality of its input and targets outputs. SEQ2SEQ models based on LSTM, was introduced as an architecture that can solve general sequential problems. SEQ2SEQ introduced at [Cho 2014] was implemented with Recurrent Neural Networks (RNN) to solve these problems. Despite of RNN based models theoretically can be applied for sequential problems, their performance gets reduced due to the long-term dependency on larger sequences. However, SEQ2SEQ implemented with LSTM models [Graves 2014] have shown better performance in many sequential tasks. LSTM cell unit (Figure 3) is defined as:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (4)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (6)$$

$$h_t = o_t \otimes tanh(c_t) \quad (7)$$

At a time t, the LSTM computes a hidden state h_t and a memory cell state c_t, for an input x_t. σ is the sigmoid function, and i, f and o are respectively the *input*, *forget* and *output* gate vectors. c is the *cell activator* vector and W is the weight matrix which corresponds to each vector. In applications, such as machine translation, two LSTM layers are used. One LSTM layer is connected to the input and obtains a fixed representation on a time step. Then, the second LSTM layer, which is conditioned to the output of the first, is a recurrent neural network language model. At the same time, it is shown that deep LSTM has significantly performance than shallow LSTM [Sutskever 2014], where each one LSTM layer is replaced by a four LSTM layers. Sequence-to-sequence models has been also tested on information extraction task, they can generate captions from videos [Venugopalan 2015]. Regarding to use only a DLSTM for the task, a DCNN model previously trained, sequentially obtains the features from the frame image of a video, and then the DLSTM generates sequential words. The model learns to associate a sequence of words with the temporal structure of the frames, allowing to handle variable-length inputs and outputs.

2.4 Model Architecture

Beat tracking system can be replaced by DL models for movement generation and synchronization. However, a DL model based on a sequential learning can improve

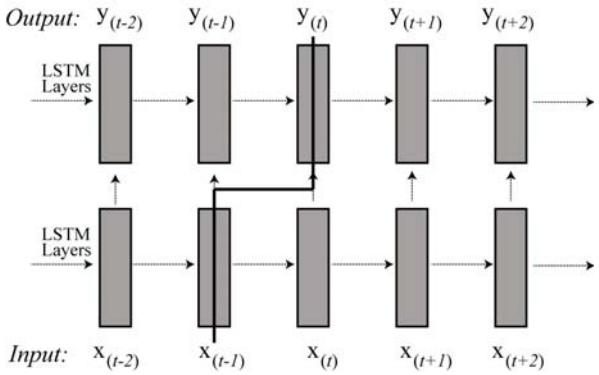

Figure 2: Sequential Learning. The thick line shows the information flow.

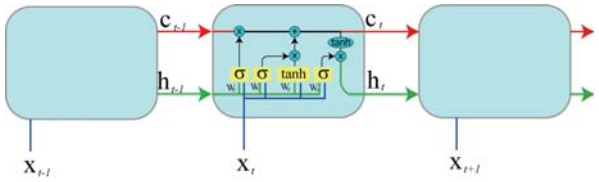

Figure 3: LSTM Cell

the performance and generate smooth motion pattern while it separates the beat information from a multiple sound source audio piece. So we implemented a DCNN +DLSTM model based on [Venugopalan 2015]. The audio has only one input channel. After the input, we connected four Convolutional Layers each one with 33×2 dimension of kernel size. The size of the kernel is empirically set up. After each convolutional layer, a batch normalization and the activation function is implemented. ELU, TanH, and ReLU has been evaluated as activation function of the DCNN part. After the fourth layer, the output has a dimension of 65×1, and this become the input for the DLSTM section. Each block of the sequential section has three LSTM layers with 500 units. LSTM layers with 250 and 1000 units has been also tested. After each LSTM block, a fully connected layer is stacked. Different activation functions have been also tested on the fully connected layer. Each block is connected as showed in Figure 4, where the output of the current last fully connected layer is concatenated with the output of the next first block and it becomes the input of the next second block.

3 Experiments

3.1 Data Preparation

For training and evaluating, we used different latin music genre such as salsa, cumbia, etc. For training, the music tracks are combined with only drum, instrumental and with singer tracks. For visualizing the generated motion pattern, we used a 3D model implemented on a simulator. For preparing the training dataset, we obtained the position information using a Kinect from a person, and synchronized with the music. The position is sampled

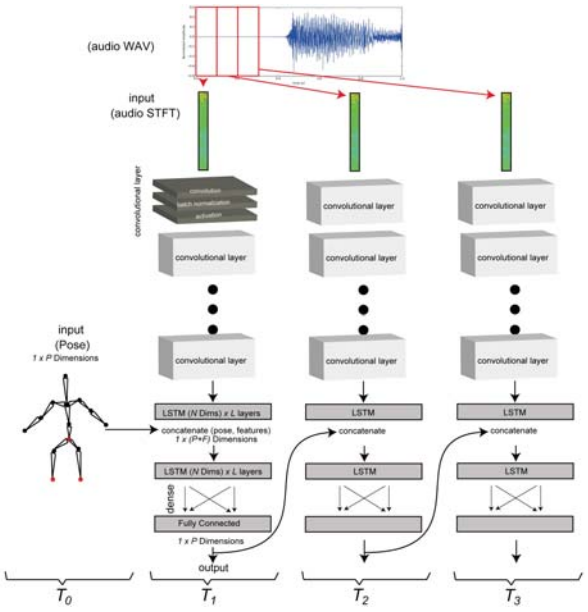

Figure 4: Model Architecture

with a frequency of 30 frames per second (FPS) and the music has a sampling frequency of 16KHz. To synchronize the music with the position, we used a slice of the audio corresponding to the same time of the position. We prepared the input data of the network using shot-time fourier transform (STFT) on the audio slice and the normalized position was used as target. As for the input, we pre-processed according the following steps:

- To synchronize the audio with the positions, we extracted a slice with length of 534 samples (33ms), which corresponds to its position. This extracted slice is converted on H STFT frames of 160 (10ms) samples with a hop of 80 samples (5ms).
- From the STFT, we used the power information, which is normalized between 0.1 and 0.9 on the W frequency bin axis.
- We stacked the H frames, thus the input of the network becomes a $1 \times W \times H$ dimensional input file.

As for the output, we prepared the position vector target as following:

- Using a Kinect device, we capture the D positions angles of a person. From the position angles, we obtain the spatial information of 3 body parts in a vector *(x, y, z)* on meters. And 14 rotations vector in quaternions with denotation (q_x, q_y, q_z, q_w).
- For each vector component, we normalized it using the maximum value of each component, between a range of -0.9 and 0.9.
- Then, the target becomes a vector of 65 dimensions.

3.2 Training Process

For our experiments, we prepared multiple training files using the STFT from around 15 music pieces as input, and a corresponding sequence motion as target. For training, each sequence step has an input of 1 channel audio $\times$ 129 frequency bins $\times$ 5 frames, and to predict we set the next corresponding vector position as target. We trained different end-to-end networks (Table 1) using ADAM solver [Kingma 2014], and Mean Squared Error is set as loss function. No fine-tune or previous training was used for the experiments. We tested different parameters during the training such as the activation function, the training sequence, the units of each LSTM layer, and the number of frames using on the input. ELU, ReLU and TanH nonlinearities are evaluated as activation functions. A sequence of 150 steps is set for the trainings. However, we also evaluated 50 and 100 steps configuration. We used 500 units for each LSTM layer, but we also tested with 250 and 1000 units. As for the input, we used a model to predict the next position using only the previous audio slice. The initial alpha for the solver was set to 10^{-4}, and the target vector was set 45 dimensions. We added noise to the gradient. We compared the training of the models (Table 1) mixing different genres and one same genre on the training dataset. We trained the models for 10 epochs, using a sequential minibatch with a size of 50 files. Each training took approximately 20 hours using a GPU NVIDIA Tesla K80.

Table 1: Models architecture configurations

layer name	output size	12 layers		
input	129×5			
conv1	97×4	33, 16 channels (TanH, ReLU, ELU)		
conv2	65×3	33, 32 channels (TanH, ReLU, ELU)		
conv3	33×2	33, 64 channels (TanH, ReLU, ELU)		
conv4	1×1	33, 65 channels (TanH, ReLU, ELU)		
LSTM1	1×1	250 dims	500 dims	1000 dims
LSTM2				
LSTM3				
fc1	1×1	65 dims fc (TanH, ReLU, ELU)		
concatenate	1×1	65 dims fc1 & 65 dims from previous step		
LSTM4	1×1	250 dims	500 dims	1000 dims
LSTM5				
LSTM6				
fc2	1×1	33, 64 channels (TanH, ReLU, ELU)		

4 Results

We analyzed the performance of the model for motion generation. Each model is trained with a shallow dataset, but the models can generalize the information for the audio input and can keep the motion pattern for trained information.

4.1 Training Process

After training for five epochs, we evaluated the use of different activation function and the performance on the training process. We used the structure described at part 3 Setting a minibatch to 150 sequences and the LSTM

layers to 500 units, we trained models with TanH, ReLU and ELU activation functions at the convolutional section. Figure 5 shows the training loss of each model. We see that despite of training a model from the scratch, the models have a fast convergence on the first iterations. The optimization process can backpropagate the error to the deepest layer; making possible a training without middle states. Regardless the activation function, the training process has the almost same convergence behavior. However, in Table 2 we show that, at the same number of iterations, ELU activation has a lightly lower learning cost compared to ReLU, TanH activations.

4.2 Motion Pattern Generation

For evaluating the performance of each model trained for five epochs, we calculate the Symmetric Mean Absolute Percentage Error (SMAPE) of the dancer's motion pattern, and the motion pattern generated by the models. The SMAPE is defined as:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{|A_t| + |F_t|} \tag{8}$$

where F_t is the output pattern from the models, and A_t is the pattern obtain from a real world dancer. Here, we compared that the model can keep and generate the motion pattern using the audio frame information and the previous calculated position. Music pieces used for training have multiple instruments, some pieces contain voices from the singer or the crowd. The dance motion generated has the shape of a wave, thus the movements keeps the beat timing of the music piece and are synchronized with the drum of each music piece. Figure 6 show pattern generated by the models and the comparison with the pattern from a real dancer. Table 3, 4 and 5 show the performance of using different parameters on the training for the motion pattern generation. We compared the motion generated resetting the LSTM each sequence period and following the pattern from the beginning. Also, we compared the evaluation of the processing time for the data forwarding using a GPU GTX Titan X. The default structure was set to ELU activations, with a LSTM of 500 units, and sequence length of 150 steps. In Table 3, we compare the generation error using same training parameters but with different activation functions. We see that ELU activation has a lower error compared to the other functions. However, its processing time is quite higher. Table 4 shows the comparison of using different unit length at the LSTMs layers. The models were trained using ELU activation and the same sequence training number. We see that, using a larger unit length can improve the performance for the motion generation. However, both the time processing and its learning cost are quite higher. Also, using few LSTM units can increase the error of the motion generation. However, the learning cost is lower as its processing time.

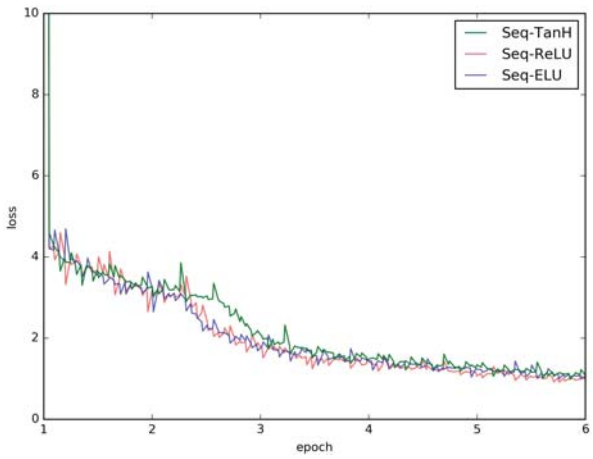

Figure 5: Training graph

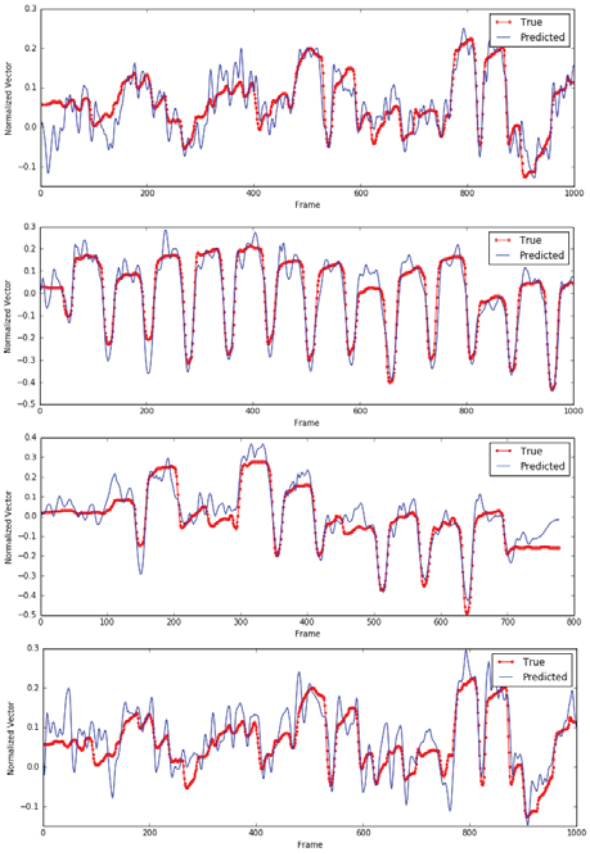

Figure 6: Dance Generated Pattern. Dotted Red: Real World Dancer, Blue; DL Models

Table 5 shows the comparison of using different length sequence on the training. Using the same model, we trained it changing the length sequence used on the training process. We see that the sequence affects the learning cost and the pattern generation error. Training a model with larger sequence improve the performance of the network. Due to the length of the sequence, the learning cost is also affected, increasing the training time. However, due to the models have the same parameters of dimensions and activation function, the processing time for real world application is not affected. Having a larger dataset for the training can lower the

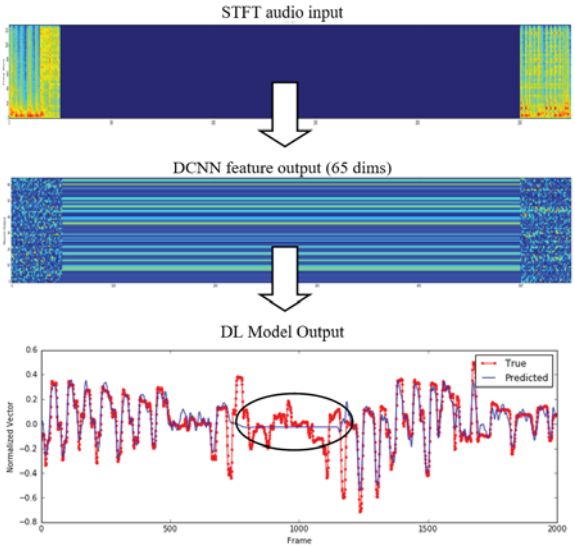

Figure 7: Generated dance pattern at silence input

Table 2: Comparison of Learning cost of activation functions

activation	Training Time (minutes)
TanH	1182.79
ReLU	1152.02
ELU	1139.81

performance of the models. For some tasks it is required larger dataset. However, we see that since the generated pattern is related with the music, a larger dataset with different motion pattern can affect these motion generator models. This becomes a noisy output for the motion (Figure 6).

4.3 Sound dependency

The models generate motion patterns from the previous motion step, but also it is required the sound information. Figure 7 shows the motion generated when the sound is stopped and became silence. We stopped the audio input at the middle of the song, thus the STFT input became blanked. At silence input, the DCNN feature outputs have a same pattern for all the silence frame. This features have been self-learned due to end-to-end training. Therefore, the motion output is stopped. However, after this silence space the sound is restored. The model restarted its pattern generation and generated the pattern from the corresponding audio input frame matching the generated motion with the trained one. Thus, the model can generate pattern from the audio input, but also can continue the generation on non-continues audio inputs.

Table 3: Evaluation of activation functions

Activation	SMAPE (%) without reset	SMAPE (%) with reset	Training Time (mins)	Average Frame Forwarding Time (ms)
TanH	19.73	20.34	1182.79	9.16
ReLU	19.67	20.27	1152.02	8.54
ELU	18.57	19.2	1139.81	10.94

Table 4: LSTM units evaluation

LSTM units	SMAPE (%) without reset	SMAPE (%) with reset	Training Time (mins)	Average Frame Forwarding Time (ms)
250	19.76	20.28	663.99	10.16
500	18.57	19.2	1139.81	10.94
1000	17.81	18.53	1305.85	14.97

5 Conclusion

We proposed a method for motion generation using Deep Sequential Learning which can be trained end-to-end. We showed that the models can generate correlated motion pattern and due to the low forwarding time, they could be used for real robot tasks. The models have a low learning cost and also can be trained from the scratch, avoiding the use of other process such as auto encoding. The proposed model shows reliable performance for motion generation. However, the motion pattern is also affected by the diversity of the training patterns.

References

[Oliveira 2015] J. L. Oliveira, G. Ince, K. Nakamura, K. Nakadai, H. G. Okuno, F. Gouyon, and L. P. Reis, Beat Tracking for Interactive Dancing Robots, Int. J. Humanoid Robot., vol. 12, no. 04, p. 1550023, 2015.

[Itohara 2012] T. Itohara, T. Otsuka, T. Mizumoto, A. Lim, T. Ogata, and H. G. Okuno, A multimodal tempo and beat-tracking system based on audio-visual information from live guitar performances, EURASIP J. Audio, Speech, Music Process., vol. 2012, no. 1, pp. 117, 2012.

[Fukayama 2015] S. Fukayama and M. Goto, Music Content Driven Automated Choreography with Beat-wise Motion Connectivity Constraints, in Proceedings of SMC, 2015.

[Krizhevsky 2012] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., ImageNet Classification with Deep Convolutional Neural Networks, Adv. Neural Inf. Process. Syst. 25, pp. 19, 2012.

Table 5: Sequence length evaluation

Training sequence	SMAPE (%) without reset	SMAPE (%) with reset	Training Time (mins)	Average Frame Forwarding Time (ms)
50	22.99	23.73	247.19	10.94
100	19.16	20.01	774.75	10.94
150	18.57	19.2	1139.81	10.94

[Clevert 2015] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), Under Rev. ICLR2016, no. 1997, pp. 113, 2015.

[Sainath 2015a] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks, ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., vol. 2015-Augus, pp. 45804584, 2015.

[Sutskever 2014] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to Sequence Learning with Neural Networks, Nips, p. 9, 2014.

[Venugopalan 2015] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, Sequence to Sequence - Video to Text, Proc. IEEE Int. Conf. Comput. Vis., pp. 45344542, 2015.

[Ioffe 2015] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv, 2015.

[Kingma 2014] D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, arXiv1412.6980 [cs], pp. 115, 2014.

[Neelakantan 2016] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, Adding Gradient Noise Improves Learning for Very Deep Networks, Iclr, pp. 111, 2016.

[Tokui 2015] S. Tokui, Introduction to Chainer: A Flexible Framework for Deep Learning. 2015.

[He 2015] K. He, X. Zhang, S. Ren, and J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, arXiv Prepr., pp. 111, 2015.

[Graves 2014] A. Graves and N. Jaitly, Towards End-To-End Speech Recognition with Recurrent Neural Networks, JMLR Workshop Conf. Proc., vol. 32, no. 1, pp. 17641772, 2014.

[Cho 2014] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Proc. 2014 Conf. Empir. Methods Nat. Lang. Process., pp. 17241734, 2014.

[Sainath 2015b] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A. rahman Mohamed, G. Dahl, and B. Ramabhadran, Deep Convolutional Neural Networks for Large-scale Speech Tasks, Neural Networks, vol. 64, pp. 3948, 2015.

[Abdel-hamid 2012] O. Abdel-hamid, H. Jiang, and G. Penn, Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition, Acoust. Speech Signal Process. (ICASSP), 2012 IEEE Int. Conf., pp. 42774280, 2012.