Human Motion Prediction

Maria Rozou
rozoum@student.ethz.ch

Rahel Straessle
strrahel@student.ethz.ch

ABSTRACT

Recurrent Neural Networks (RNNs) are commonly used nowadays in machine learning applications, starting from natural language processing as well as image analysis applications, gesture recognition or other deep learning applications. In the recent years, RNNs have shown good performance when applying different RNN architectures also to human motion prediction tasks. In the current work, a state of the art LSTM cell is used as the base of our network to predict human motion poses. We can conclude that a basic LSTM network, with the most optimized parameters performs a very good job, though it cannot compete a state of the art sequence to sequence model when it comes to human motion prediction.

1 INTRODUCTION

Motion prediction is one of the tasks that is automatically done easily by humans but very hard to accomplish by machines. While even insects already have a physical model of the world, either learnt or predefined, we have to teach the machines how to predict motion. There are many applications nowadays that are based on human motion data analysis like human-computer interaction, motion synthesis, motion prediction for virtual and augmented reality, automated driving or biomedical engineering applications. Using a state of the art unrolled LSTM cell we build the most basic RNN network for human motion prediction and took this as our baseline for improvements.

2 RELATED WORK

Recent work focused on RNN based architectures to model human motion, with the goal of learning time-dependent representations that perform tasks such as short term motion prediction.

In 2014, an encoder-decoder (ERD) network based on LSTM was introduced, which is a type of recurrent neural network (RNN) model, that combines representation learning with learning temporal dynamics [2]. Basic preprocessing is done to increase performance and robustness of the network, like standarization of data and adding gaussian noise to the input data. Around the same time, a innovative sequence to sequence architecture for language translation was proposed [6] that would give to the data science field a new direction. More specific, the network consists of an encoder and a decoder and would take as input one whole sentence, so that the network learns the context or the sequence of inputs and not each input independent. An extended sequence to sequence model was recently published [4] for human motion prediction. It is proposed that by sharing weights between encoder and decoder and using Gated Rectified Units (GRU) instead of LSTM as base cell simplifies the model but does not decrease the final performance. A residual connection between input and output allows to use velocities in the network - an advantage add-on of utmost importance as human motion is dynamic and not static.

Additionally, successful long-term results were obtained with a Dropout Autoencoder LSTM (DAE-LSTM) that can extract both structural and temporal dependencies directly from the training data [3]. This network can produce long-term naturally looking poses without converging to mean. The autoencoder is trained to implicitly recover the spatial structure of the human skeleton via randomly removing information about joints during training. A different approach shows that a feed-forward network can also compete with RNNs [1].

3 METHODOLOGY

3.1 Preprocesssing and representation

Preprocessing the data before feeding them into the network is understood to be advantageous [2]. However, for our RNN standardization of the input data would not lead to the best performing model. We standardized the data by subtracting the mean value of each joint and also divided by the standard deviation. One point worth mentioning is that some joints had zero standard deviation, meaning that the training data set were not changing at all for these angles, therefore we decided to discard them and not use them in the analysis. This led to an input with 51 angles instead of 75. If the test set is coming from the same distribution, we could expect that the same angles also in the test set have zero standard deviation so they can be removed from the LSTM analysis and for the test purposes just keep the last frame of these angles as the predicted one. We did not see any improvements of the performance of our LSTM model compared to non-standardized inputs.

3.2 Network structure

In this project, the architecture we used to solve the problem of motion prediction is based on a RNN architecture using LSTM cells [5]. We created an unrolled LSTM network that in each time step is being trained to predict the next pose . Therefore we identified the problem as a regression task, where we construct the predicted poses in the continuous angles-space. This can be seen also in Figure 1, where the red poses are the predicted poses from the LSTM architecture and the green poses are the ground truth, which is in our case the real-true positions from the training data. For the prediction, 50 source frames are feed into the model. The model predicts the first next step using the last feed frame and continues afterwards using the predicted frame as the input for the next prediction until frame 25. The model as well as the training and the prediction process is shown in Figure 2.

Moreover, for language translation applications it was shown that deep LSTMs outperform significantly shallow LSTMs [6]. An optimum was reached with a sequence to sequence architecture with 4 layers. Stacking of LSTM cells increases the model complexity and predictive power but at the expense of training times and difficulties. We combined the LSTM cell with a drop-out and stacked those. Specifically, the dropout technique was added since it makes models in general more robust.

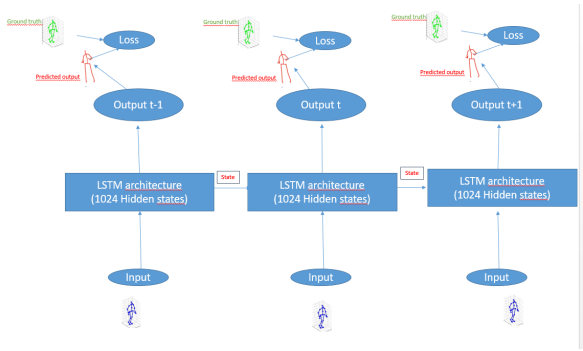

Figure 1: Training architecture

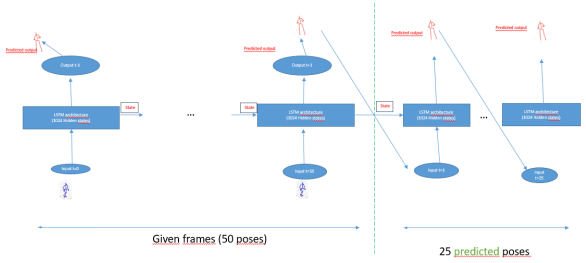

Figure 2: Prediction architecture

3.3 Experiments

Our models are trained on 162 samples with different length. Each sample is a dictionary with an array with 3x25 entries for the joint angles per frame, the action label and the ids. The validation was done on a separate set of 18 samples. For training and validation set together, there were 12 samples of 14 different activities. All the samples are cut at a maximum sequence length and shorter sequences padded with zeros. The padded elements were disregarded from the loss in the training phase, so that they dont influence the training procedure. The test set consists of 516 samples with 50 frames each.

The loss function we used was the mean squared error function (L2 loss) to calculate the prediction error for the training sequence. As comparison, we also tested another distance function, the cosine loss but did not outperform the L2 loss.

To solve the optimization problem in the training process, we used stochastic gradient descent (SGD), and more specifically Adam optimizer. For the optimizer, we looked at different choices: the ADAM optimizer, the Gradient Descent Optimizer and the RMSPropOptimizer. Moreover, next to the learning rate more parameters of the optimizer can be tuned as well like clipping the gradients.

4 RESULTS AND DISCUSSION

The best performing model was the most basic LSTM model with one layer, trained with the L2 loss and optimized with the Adam Optimizer. We clipped the gradient norm to a factor of 5, so that we limit the magnitude of the gradients. With such a configuration, the Adam optimizer descends indeed to a minimum. In the following table the parameters of the LSTM architecture are summarized. The training progress shows that after 10 epochs there was no substantial decrease in validation error. To avoid overfitting, our final submission is with 6 epochs. A small batch size can help to step away from local minima. In our best run, we had a batch size of 10.

Table 1: Parameters for our best performing model

Parameter	Value
Learning rate	0.0015 fixed
Batch Size	10
Optimizer	Adam
Epochs	6
Max Length size	600
Clipping gradient	5
Hidden states	1024
Layer	1

Adding more layers would not yield better results compared to a single layer. It is well possible that we couldn't find the convergence point for this model. Neither did it help to add a drop-out layer. Dropout is more useful for deep networks, which can be the reason that dropout would not improve our validation error, even after 12 epochs, for our rather shallow networks. The predictions were very human like, also for the longer term prediction, but they were flickering or noisy. Even though in this model we could not see an improvement, we believe that both stacking of more layers and dropout could increase model performance and add robustness to the model if another architecture was chosen.

Other distance functions, like the cosine loss, did not improve the performance of the model on the validation set and instead the results were poorer. As well for the optimizer, changing to another optimizer or adjusting the settings, such using not fixed learning rate, would not improve our model performance.

5 CONCLUSION AND OUTLOOK

Our network performed reasonably well for its simplicity. Tuning the hyper-parameters, optimizer or loss function would not have a significant influence on the performance. Adding more layers and dropout could not improve the model. This is in line what we could expect when consulting the literature.

Last but not least, the movements of human body are sequential, meaning in each frame there is a dependency from the previous. Therefore, the obvious architecture that is outperforming the LSTM network is a sequence to sequence (seq2seq) architecture, where two networks are trained: an encoder generates an internal representation of the input, which is fed to the decoder. The decoder produces a maximum likelihood estimation for the prediction. One of the obvious advantages is that the model is trained for exactly the task that it should carry out in the test. As the seq2seq architecture is heavily used in machine translation, there are several options to improve the architecture also for motion prediction, such as attention or bidirectional architectures.

Furthermore, another preprocessing step would be to add Gaussian noise to the angles as has been done in previous work [2]. This

makes the LSTM more robust and can help to avoid overfitting. However, the robustness comes with the price of having another hyperparameter to tune: the standard deviation of the Gaussian noise. Another solution to increase robustness of the model is to feed into the decoder its own produced samples. This comes without any parameter to tune and yields good results [4].

Additionally, human motion is dynamic and not static and human motion can be better expressed in velocities than in angles-joints positions. Previous work has taken this into account and used a residual connection to have the prediction continuous [4]. Many architectures suffer from a unreasonable first step when the prediction starts which can be avoided with the residual connection. A add-on to this model could include a further connection from the last input to the output that would teach the model about acceleration too.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Dr. Otmar Hilliges for providing the theoretical background needed for this work and his assistants for the skeleton code. The work was carried out in the frame of the lecture "Machine Perception" at ETH Zurich in the spring semester 2018.

REFERENCES

[1] Judith Bütepage, Michael J. Black, Danica Kragic, and Hedvig Kjellström. 2017. Deep representation learning for human motion prediction and classification. *CoRR* abs/1702.07486 (2017). arXiv:1702.07486 http://arxiv.org/abs/1702.07486

[2] Katerina Fragkiadaki, Sergey Levine, and Jitendra Malik. 2015. Recurrent Network Models for Kinematic Tracking. *CoRR* abs/1508.00271 (2015). http://arxiv.org/abs/1508.00271

[3] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. 2017. Learning Human Motion Models for Long-term Predictions. *CoRR* abs/1704.02827 (2017). arXiv:1704.02827 http://arxiv.org/abs/1704.02827

[4] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *CVPR*.

[5] Christopher Olah. [n. d.]. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] I. Sutskever, O. Vinyals, and Q. V. Le. [n. d.]. Sequence to Sequence Learning with Neural Networks. *ArXiv e-prints* ([n. d.]). arXiv:1409