Human Motion Prediction

Maria Rozou
rozoum@student.ethz.ch

Rahel Straessle
strrahel@student.ethz.ch

ABSTRACT

Recurrent Neural Networks are commonly used in natural language translation and also generation. In the recent years, they have shown good performance when applying them to human motion prediction tasks.

0.1 Preprocesssing and representation

1 INTRODUCTION

Motion prediction is one of the tasks that is automatically done 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 it to machines. 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, biomedical engineering applications etc. 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.

Fragkiadaki 2014: introduced an encoder-decoder (ERD) network, which is a type of recurrent neural network (RNN) model, that combines representation learning with learning temporal dynamics [2]. Also there the writers used some basic preprocessing, like standarization of data and adding noise to the input data so that the performance of the RNN base don LSTM technique is better.

Julieta Martinez 2017: Trained encoder and decoder together with shared weights, single GRU. Used velocities through a residual architecture. Error reduction: fed predictions of the net back [3]. This work is similar to the techniques use din NLP, Natural Laguage Processing, as described in [5]

Sutskever 2014 (Sequence to sequence learning with neural networks): Deep LSTM's significantly outperform shallow LSTMs. They used therefore 4 layers. They also reversed the order of word inputs and would then perform better on long sentences. Stochastic gradient, 7.5 epochs, same length of input vector in batches: speed up of training. Best result with ensemble of LSTM that differ in their random initialization and in the random order or minibatches. [5]

Millinges et al:
Olah: [4]

Buetepage: [1]

3 METHODOLOGY

3.1 Preprocesssing and representation

Preprocessing the data before feeding them into the network is understood to be advantageous. 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 0 std 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

3.2.1 Single layer LSTM. In this project, the architecture we used to solve the problem of motion prediction is based on a RNN architecture using LSTM cells. 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.

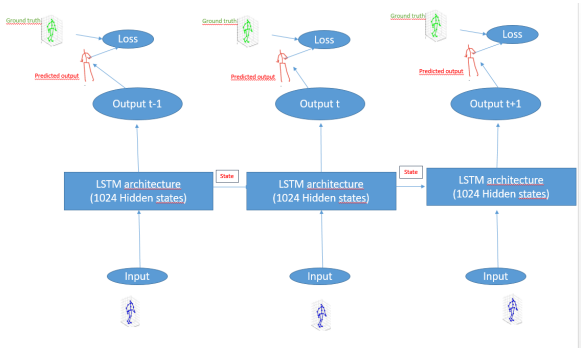

Figure 1: Training architecture

The loss function we used was the mean squared error function (L2 loss) to calculate the prediction error for the training sequence. Other distance functions, like the cosine loss, did not improve the performance of the model on the validation set or even gave worse results.

Table 1: Value of the Parameters

Parameter	Value
Learning rate	0.0015 fixed
Batch Size	10
Optimizer	Adam
Epochs	9
Max Length size	600
Clipping gradient	5

To solve the optimization problem in the training process, we used stochastic gradient descent (SGD) and the ADAM optimizer. Also here, changing to another optimizer or adjusting the settings, such as epsilon, would not improve our model performance. 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.

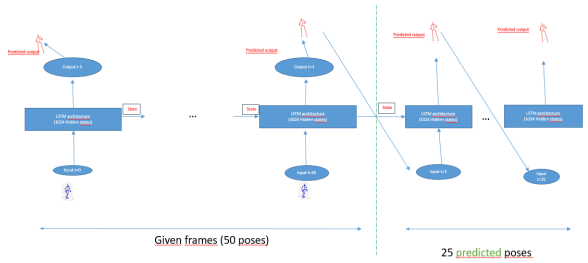

Figure 2: Prediction architecture

In the following table the parameters of the LSTM architecture are summarized. The training progress show us 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.

Figure 1

3.3 Multilayer LSTM with dropout

It is possible to stack layers of LSTM cells on top of each other âĂŞ this increases the model complexity and predictive power but at the expense of training times and difficulties.

3.4 Failed approaches to improve model

Mulitlayer network with and without dropout. Dropout makes models in general more robust but is more useful for deep networks. Which we could experience as dropout would not improve our validation error, even after 12 epochs. We believe, however, that dropout can add robustness to the model if another architecture was chosen.

4 EXPERIMENTS

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 expect from the literature.

The obvious architecture that is outperforming the LSTM network is a sequence to sequence 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 we train the model 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.

To sum up, one improvement that is suggested is to use sequential models for the prediciton of Human Motion. The movements of human body is performing is sequential, meaning in each frame there is dependecy from the previous and also the sequential model will be able to mimic these dynamics of the human motion.Therefore we propose as a next improvement that the state of the art unrolled LSTM is replaced with a seq2seq model.

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 [3].

Human motion is dynamic and not static and human motion can be better expressed in velocities than in positions. Previous work has taken this into account and used a residual connection to have the prediction continuous [3]. 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.

Another interesting improvement could be to use an Adversarial network to produce many more samples.

Include a physical model. As humans we train the internal physical model every day by moving our own body.

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 that allowed us to focus on the more interesting aspects of the task.

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

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

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