Human Motion Prediction

Jovan Andonov
andonovj@student.ethz.ch

Ilija Stanojković
ilijas@student.ethz.ch

ABSTRACT

Modeling, understanding and predicting human motion has many applications in industries such as game development, video animation, 3D modeling, medicine, biomechatronics etc.

As human motion is a sequential problem, it seems reasonable that machine learning techniques already proven to work in natural language processing and understanding can also be used to tackle it. In this work, we focus specifically on the problem of human motion prediction. We start by implementing a simple RNN baseline and we further improve our performance by using sequence-2-sequence models and state of the art mechanisms such as attention.

1 INTRODUCTION

Humans are active creatures. As such, all types of motion are part of their daily activities. Doctors, biologists, software, mechanical and electrical engineers, as well as animators and designers are all working on modeling and understanding human motion. People are researching motion everyday, hoping they will come up with better understanding, that can later be used to improve many industries.

For example, Hugh Herr's Bionics Lab [8] at the Massachusetts Institute of Technology is researching human motion to improve their bionics. Prof. Yong-Lae Park's Soft Robotics Lab [14] at Carnegie Mellon is researching how muscles move and develops soft robotics inspired by them. Furthermore, many people in the movie and video animation industries are constantly trying to improve 3D human models and motions to make them look as realistic as possible. Predicting human motion has incredible applications in all of the aforementioned topics. By being able to get a short glimpse into the future, we can optimize for the present. Therefore, the problem of human motion prediction is an incredibly interesting and challenging one.

Humans predict the future based on explicit or implicit probability distributions for a certain domain. More precisely, given their knowledge of the domain and a certain amount of information regarding an event of interest, humans can approximate what is the most likely outcome for the future. Until recently, for a machine to predict complex things such as human motion, was thought to be impossible. However, with the latest breakthroughs in machine learning, it is actually the machines who have the upper hand, when it comes to modeling probability distributions and predicting the future. If we look at the human motion prediction problem, we can model it as a conditional probability distribution:

$$P(x_t \dots x_{t+\Delta} | x_0 \dots x_{t-1})$$

Let $x_0 \dots x_{t-1}$ be an input sequence of motion. Let each $x_i \in R^D$ be a vector representing a complete human posture at a certain time. We can then interpret this as: *"What is the most likely $x_t \dots x_{t+\Delta}$ sequence of human postures that can happen, given that we have observed the sequence $x_0 \dots x_{t-1}$"*. As the domain is not categorical, but rather in R^D, we are left with solving a multivariate regression problem.

In this work, we focus on predicting human motion by using state of the art deep learning techniques. We start by implementing a simple RNN that serves as a baseline and allows us to get acquainted with the skeleton code. We then reproduce the state of the art model. Finally, we extend that model to make use of the attention mechanism, add regularization in the form of dropout and add several layers of preprocessing.

2 RELATED WORK

For a couple of decades now, human motion has been researched extensively with several surveys done along the way [1][17]. Over the years, numerous solutions and improvements have been proposed, using various methods and techniques, ranging from both traditional machine learning approaches to currently wide-spread deep learning ones.

Some of the prior methods utilized for generation of human motion were based on Hidden Markov Models [11]. These models were easy to optimize, but due to the simplicity of the model, could not capture the entire distribution and generalize well. Improvements were, among others, proposed by Wang et al. [16], who used Gaussian Processes and Bayesian approach to learn nonlinear dynamic systems.

Since both natural language processing and human motion prediction can be expressed as sequence to sequence modeling, methods used to tackle both could be interchanged. Therefore, RNNs and Encoder-Decoder Networks are some of the models which seem promising for the task of motion generation. Fragkiadaki et al. [6] proposed Encoder-Recurrent-Decoder networks for both body pose labeling and body pose forecasting. The model described in their paper takes each frame of the motion sequence and feeds it to the encoder, which effectively embeds it into latent space. The encoded frame is then passed to a LSTM [9] cell, which accumulates the knowledge about human movement along time and predicts the next frame in the latent space. Afterwards, the cell output is decoded into the original space.

Martinez et al. [12] proposed a classical seq2seq model, inspired directly from machine translation, with two additional changes. The first change they added was a sampling-based loss. Instead of taking the ground truth decoder inputs, they used the decoder output in the previous step as input in the next one. This enforced the network to learn from its own mistakes. The second change was a residual connection between the input and the output of each RNN cell, which is nothing more than a simple element-wise summation of the input and the output. The idea was that by modeling first-order motion derivatives, the network would be able to generate short term predictions better.

Ghosh et al. [7] researched a different RNN model, and proposed stacked architecture of a LSTM network and a de-noising autoencoder. Their reasoning behind running the predicted frame through an autoencoder, after being processed by the LSTM cell, was that

this would potentially prevent the network to accumulate error noise and motion drift over time.

3 DATA

In this work we use a subset of the Human3.6M dataset [10] [3], which consists of 3D human poses and corresponding images. It is composed of 15 action class labels, ranging from jumping and walking, to eating and discussing. This subset contains 202821 train, 21319 validation and 25800 test frames. Each frame describes a human posture, previously denoted as $x_i \in R^D$. There are several ways to represent human postures, and probably the first that comes to mind is keeping 3D Cartesian coordinates of the joints. This representation, although suitable for visualization, does not fit our task of motion prediction well, since it is not invariant to scale and body proportions. Therefore, joint angles w.r.t. its parent are used for parametrization.

By default, data fed to our model is first split into disjoint sequences to match a predefined maximum sequence length. One data augmentation method we propose is to split data into overlapping sequences. This is controlled by two hyperparameters: *maximum_sequence_length* and *stride_value* – in our case, set to 75 and 25 respectively. The *stride_value* parameter defines the difference between the beginning of two neighboring sequences, meaning if *maximum_sequence_length* and *stride_value* are set to be the the same, we would obtain the default disjoint splitting behavior. Consequently, by using this striding sequence extraction, we end up with more training samples. However, reasonably choosing the *stride_value*, is of great importance. Choosing, a *stride_value* of 1 results in the highest amount of training samples, however these samples are highly correlated and therefore of poor quality. We believe that, a value of 25 is a good trade off between the amount of training samples and their quality. We base our belief on the fact that this way, the model sees each frame 3 times within a single epoch - once in the beginning of the sequence, once in the middle and once as part of the prediction sequence.

4 MODELS

We propose several different models and results for over 90 different hyperparameter configurations. In this section we are going to explain the models, and in the following one we are going to both qualitatively and quantitatively describe how different hyperparameters influence training and performance. All models use the same input, as described in the previous section, however some of them use different pre/post processing techniques.

4.1 Baseline

As a baseline we use a vanilla GRU [4] based RNN. GRU stands for Gated Recurrent Unit and together with LSTM, they are the two most popular RNN cell architectures. We decided to use GRU instead of LSTM, because there are studies showing that GRU tends to work better for small datasets [5]. The current state of the art is also using GRUs, and it turns out that it outperforms other LSTM models even on the entire Human3.6M Dataset [10].

The RNN is being used in a language model [13] fashion, hoping to obtain a human motion model. During training we feed sequences of human postures of a maximum length of 75. This seems like a logical choice, if we consider the fact that during inference we always receive sequences of that same length. During inference, we feed the first 50 sequences and we let the human motion model generate the last 25.

4.2 State of the art

The state of the art is achieved by the previously mentioned Martinez et al. [12] model. Using this model, we passed the hard baseline.

4.3 Extensions

Furthermore, we extend the state-of-the-art model in multiple ways. We start by adding a few preprocessing options, among which, the first one is data normalization. We use min-max normalization, ending up with features $x_i \in [0, 1]^D$. Analyzing feature variations, we noticed that some features, barely change throughout the entire dataset, with some of them simply being 0. Therefore, we also add a filtering step, that acts as a feature selection mechanism. We then highly modularize the state of the art model, so that we are able to include or exclude individual parts of the computational graph, by simply setting a binary value within a config file. Additionally, we add an attention mechanism [2] that represents the current state of the art addon, when it comes to seq2seq models. Finally, we add regularization in the form of a dropout layer [15] on top of the encoder outputs.

The following model configurations are available within a config file:

- ***normalize***: boolean specifying whether min-max normalization should be used.
- ***train/valid_stride***: boolean specifying whether striding should be used during training/validation.
- ***stride_value***: integer denoting the striding value.
- ***concat_labels***: boolean specifying whether one hot encoded action labels should be concatenated to the joint angles.
- ***hidden_state_size***: integer specifying the amount of output units within a single GRU cell.
- ***num_layers***: integer specifying the number of layers
- ***model_velocities***: boolean specifying whether a residual wrapper for the GRU cell should be used.
- ***share_weights***: boolean specifying whether the encoder and the decoder should share weights.
- ***attention***: boolean specifying whether attention should be used.
- ***dropout***: boolean specifying whether a dropout of 0.5 should be used.
- ***learning_rate_type***: string specifying the type of learning rate

Figure 1 shows an abstract high level representation of our model's computational graph.

5 RESULTS & DISCUSSION

Our baseline model reached a mean squared error of only 0.026174. Having this in mind, throughout this sectionm we will neglect it.

Once we passed the hard baseline and obtained the final model, described in section 4.3, we decided to focus on hyperparameter tuning. For this cause, we created a bash script that calls our train.py

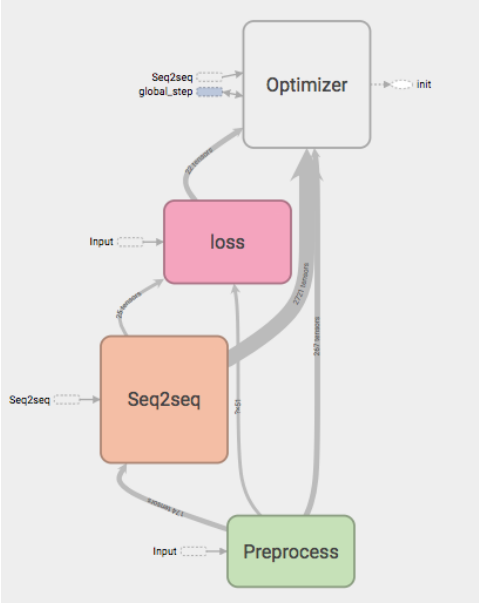

Figure 1: Tensorflow Computational Graph

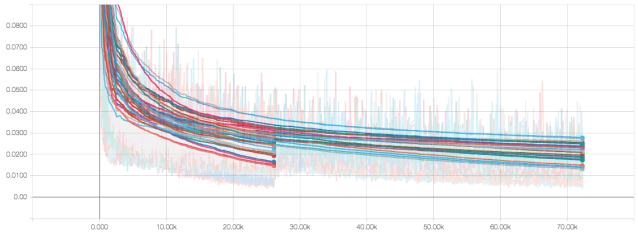

Figure 2: Training losses

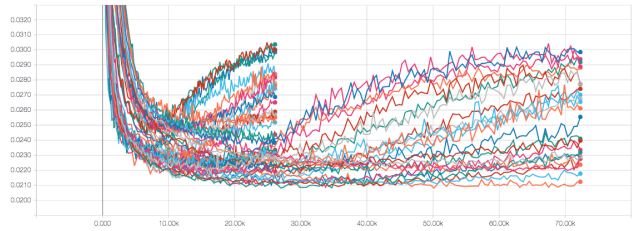

Figure 3: Validation losses

file with different command line arguments, respectively overwriting the static config values. As the time complexity for evaluating all possibles models scales exponentially in the amount of hyperparameters, we decided to do an initial grid search followed by a complete DFS search.

Normalization and filtering did not improve the results, so we simply fixed those parameters to false. On the other hand, the residual wrapper resulted in a constant performance boost which is why we fixed the *model_velocities* to True. Dropout, helped attention models generalize better, so we always used it in attention models. We also noticed that simple models tend to work better on the dataset we used, hence we fixed *num_layers* to 1.

The parameters over which we did a bruteforce search, were: *train_stride, concat_labels, hidden_state_size, attention, share_weights* and *learning_rate_type*. Except for the *hidden_state_size*, where we considered the values 128, 256 and 512, all other hyperparameters were binary variables. Regarding the learning rate we used two configurations: 0.005 with *exponential* decay and a *fixed* learning rate of 0.0001. We ended up with 96 models in total. Unfortunately, all models that used a learning rate of 0.005 and an *exponential* decay started diverging after approximately 30 epochs, so we only analyzed the remaining 48 models. Detailed tensorboard information about all of them can be found at: http://data-cruncher.westeurope.cloudapp.azure.com:8080.

By looking, analyzing and comparing the training (fig. 2) and validation (fig. 3) curves of different models we came to multiple conclusions regarding how our hyperparameters influence training and performance.

- ***stride*** - models that do not use striding, train on less data and are therefore more prone to overfitting. This can be confirmed by looking at the curves. The very same models start overfitting later when striding is used.
- ***hidden_state_size*** - increasing this value results in obtaining a more powerful model. However, as mentioned previously simple models worked better. Our best model used a *hidden_state_size* of 128.
- ***share_weights*** - having separate specialized encoder and decoder also turned out to be an overkill for this task. Sharing weights corresponds to using a simpler, less powerful model.
- ***concat_labels*** - contrary on what Martinez et. al [12] say, concatenating one hot encoded action labels did not have an impact on our performance.
- ***attention*** - adding attention resulted in the most significant power boost. Unfortunately, even though attention models were able to learn the training set remarkably well, they generalized poorly in comparison to our best model.
- ***learning_rate*** - we found out that using a learning rate larger than 0.001 will eventually cause our model to diverge.

All this being said our best model used: striding, a *hidden_state_size* of 128 and was sharing weights between the encoder and the decoder. The loss achieved on the training set was 0.02065, on the validation was 0.02081 and on the test set was 0.01707.

6 FUTURE WORK

Our investigation hints that using more data, or augmenting data in different ways would most likely benefit the existing models and improve predictions. To be more precise, all of our complex models were struggling with overfitting and as more data is always the cure for that, we are pretty sure that the attention model will outperform the current best model.

Additionally, we are curious to see how CNN models would perform. Again, inspired by language processing, one could use CNNs in the same manner as CNN language models are used.

REFERENCES

[1] Jake K Aggarwal and Quin Cai. 1999. Human motion analysis: A review. *Computer vision and image understanding* 73, 3 (1999), 428–440.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Cristian Sminchisescu Catalin Ionescu, Fuxin Li. 2011. Latent Structured Models for Human Pose Estimation. In *International Conference on Computer Vision*.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[6] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 4346–4354.

[7] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. 2017. Learning Human Motion Models for Long-term Predictions. *arXiv preprint arXiv:1704.02827* (2017).

[8] Hugh Herr. [n. d.]. Biomechatronics Lab - MIT. https://biomech.media.mit.edu/.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (jul 2014), 1325–1339.

[11] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. 2014. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1314–1321.

[12] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. *CoRR* abs/1705.02445 (2017). arXiv:1705.02445 http://arxiv.org/abs/1705.02445

[13] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[14] Yong-Lae Park. [n. d.]. Soft Robotics Lab - CMU. http://softrobotics.cs.cmu.edu/.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* (2014).

[16] Jack M Wang, David J Fleet, and Aaron Hertzmann. 2008. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence* 30, 2 (2008), 283–298.

[17] Liang Wang, Weiming Hu, and Tieniu Tan. 2003. Recent developments in human motion analysis. *Pattern recognition* 36, 3 (2003), 585–601.