Multi Head Attention for Motion Attention

Guy Lüthy
guluethi@student.ethz.ch

Dominik Alberto
doalbert@student.ethz.ch

Gabriel Gavrilas
ggabriel@student.ethz.ch

ABSTRACT

Human motion prediction is the task of predicting future human poses given past motion. Recent approaches have applied attention mechanisms to achieve state of the art performance. Inspired by these approaches, we set out to integrate more concepts from the original "Attention Is All You Need" paper to further improve an existing model. We propose a model based on motion attention using multi-head attention. Additionally, we discuss the applicability of Transformer models to this problem.

1 INTRODUCTION

Human motion prediction is the task of predicting future human poses given past human motion. Traditionally this task was solved using CNNs and RNNs[6, 15]. Following the good results of models using attention and specifically Transformers [19] in NLP, recent approaches have tried to use attention and Transformers for machine perception and specifically also for human motion prediction.

Our work is mainly based on Mao et al.[13] and Mao et al.[14]. Mao et al.[14] proposes a two-fold encoding. They use a DCT[1]-based temporal encoding to represent human motion in the trajectory space. To encode the spatial structure, they use a GCN, which efficiently models the spatial dependencies between joints. Mao et al.[13] extends these concepts by introducing the concept of motion attention, where the input is split into sub-sequences, for which the model then computes attention weights, which it then passes to the GCN together with the input.

Inspired by the idea to use attention for human motion prediction, we set out to implement more concepts from the original "Attention Is All You Need"[19] paper. Specifically we focused on applying multi-head attention and the Transformer model. The model we propose here introduces multi-head attention to the model defined in Mao et al.[13]. While this model has given us the best performance, we have investigated many alternative approaches. The most interesting of these was our efforts to use the Transformer for human motion prediction. Most attempts did not give good results and our current model consistently outperformed the Transformer models. In Section 5 we discuss possible reasons for the bad performance of these models.

We have implemented our models using PyTorch[17]. Wherever possible, we tried to use already implemented models such as `torch.nn.MultiheadAttention` and `torch.nn.Transformer`.

As provided in the project, we use a subset of the AMASS dataset which consists of frames defined by joint angle matrices. As given in the project description the data is already split into training, validation and test sets. The training data is further split into sequences of size 144 to match the length of validation sequences. The test set consists of sequences of length 120 and the goal is to predict the following 24 frames.

2 RELATED WORK

In this section we briefly want to mention some additional papers related to our work.

The state of the art for sequence-to-sequence models are currently recurrent neural networks (RNNs). Therefore, RNNs have been widely used for human motion predictions as well, such as by Fragkiadaki et al.[6].

Attention for human motion modeling has been explored in Tang et al.[18], but in contrast to our approach in a frame-wise manner.

Aksan et al.[2], to our knowledge, are the only ones to apply the idea of Transformers to human motion prediction. They propose a new model based on the Transformer with decoupled temporal and spatial self-attention mechanisms.

The following papers have used Transformers for time series forecasting. Li et al.[12] show that Transformers lack locality and the memory bottleneck of Transformers. Huang et al.[8] apply Transformers to music generation. Child et al.[4] propose a sparse Transformer architecture to predict longer sequences. Zimbres[22] modifies the Transformer for time series prediction and proposes to interpret the output of the Transformer as probabilities.

Finally Kazemi et al.[9] propose a model-agnostic vector representation for time that can be used as a positional encoding for Transformer models.

3 METHOD

3.1 Base Implementation

We first implemented the mechanics described in the History Repeats Itself paper. We used some code that was publicly available on GitHub [21]. The base model uses attention modeling, which needs three parameters as input, namely the Keys, a Query and the associated values. We select two parameters, the Kernel size K and the Direct Cosine Transform (DCT) depth D. As Key, we can then use the full 120 Frames of input data, and process it through a 1D-Convolutional Neural Network (1D-CNN). As Query, we use the last K frames, processed through a 1D-CNN as well. Finally, for the value vector, the input is first split and duplicated into $N_{IN} - K - N_{OUT} + 1$ sections of length $K + N_{OUT}$ each. For example, the first section contains the frames [0..33], the second section contains the frames [1..34] and so on. Each of these value sections is then transformed using the Direct Cosine Transform, to encode temporal dependencies in the data, by representing each sub-sequence in the trajectory space. We found the optimal values for the parameters K and D by analyzing a set of different ones and choosing the ones with optimal performance.

To now predict the future motion, we use a a Graph Convolutional Network (GCN). This approach allows the model to exploit spatial locality within the calculated attention. The GCN then directly predicts the next N_{OUT} frames. As input to said GCN we use the calculated motion attention, as well as the last ten frames from the input sequence. To pad the input size to the needed length, the last frame is repeated until the full sequence reaches a target lenth of D.

3.2 Novel Approach

For our novel approach, we included the principle of multi-head attention into this model. Instead of having a single attention vector, we calculate multiple attention 'heads', and feed everything into the GCN again. This allows the GCN to work on more information, than a single attention head would provide, and therefore make more accurate predictions. Figure 1 displays how the model works in a block diagram. The drawbacks of this approach is, that there is a lot more to train. The number of attention heads linearly increases the number of 1D-CNNs, as well as the number of input features of the GCN. This leads to longer running times, however with modern Hardware the model is still somewhat performant.

Compared to the method used by Vaswani et al.[19] we use a concatenation of our attention heads, instead of combining them with some metric. We tried to lose as little information as possible, and the approach of simply concatenating the output is the most straightforward way of doing so. The only downside is that the GCN is slightly larger, but this does not lead to a big performance impact.

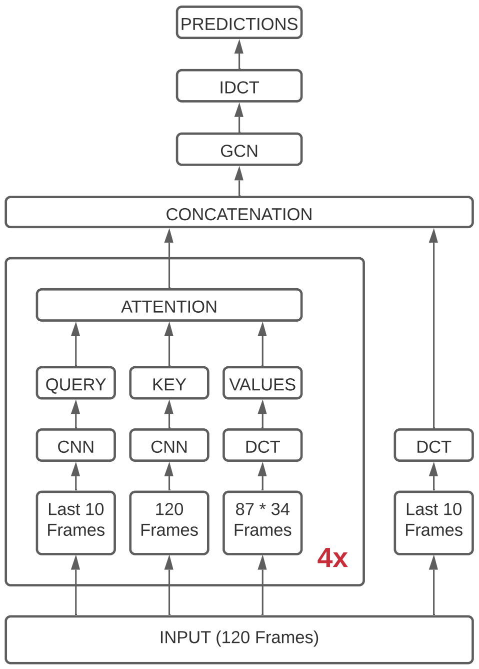

Figure 1: The block diagram of our prediction model. Note that the attention block is repeated four times, to implement multi-head attention

3.3 Implementation Details

Our final model consists of four attention heads, each having its own input set of Keys, Queries and Values. Each attention head first processes the input data using the process described above, and together with the random initialization of the CNNs this leads to for four different attention vectors. These attention values are then merged to one, large attention vector that is passed as input to the graph convolutional network. The GCN consists of a single linear layer, followed by four GCN layers, and a final single linear layer in the end. Each GCN layer employs dropout to prevent overfitting. For the process of applying DCT we use the PyTorch builtin matrices, that allow to apply the DCT with a simple matrix multiplication.

Table 1: Comparison Joint Angle Scores after 20'000 epochs

	Validation Set	Public Test Set
Single-Head Attention	val;	1.94438929312
Multi-Head Attention	2.352	1.89348918873

3.4 Training

We use the Adam[10] optimizer, and a simple mean squared error metric between the predictions and the ground truth. We trained the model for a total of 20'000 epochs, using a batch size of 256. Using this batch size the model needs to train for around 60 hours on the supercomputer.

4 EVALUATION

After training with 20'000 epochs, we got very promising results as shown in Table 1. The joint angle metric used to grade the model was lower, when compared to the single attention head described in [13]. Furthermore, the achieved public score on the online grading platform was better as well.

4.1 Performance Analysis

We think that the gained improvements are due to the wider information available for the GCN. Since our actual predictions are done by said network, having more, and a wider variety of information available than with a single attention head, leads to better predictions. However the optimal number of attention heads is still unknown. There is research done, especially in the field of natural language processing, as done by Voita et al.[20], however in motion prediction this is largely unexplored. Nevertheless, more attention heads seem to have an impact on the prediction quality, so we remain confident that multiple heads work for motion prediction as well. As mentioned above, this comes with a cost. The model grows quite large in size, in all aspects. The additions imposed by the CNN parts of the model can grow quickly, and therefore the number of heads should remain somewhat lower, to not impact performance to a point of where the model is simply to large.

4.2 Comparison to Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are state of the art for motion prediction right now. As Mao et al.[13] mention, they achieve good results and are highly successful on sequence-to-sequence tasks. However, the use of RNNs usually leads to differences between the last input frame and the first prediction frame. Gui et al.[7] tried to overcome this downside by using adversarial training, however these models are notoriously hard to train [3].

Our use of motion attention does not lead to such jumps. Since we model our motion in trajectory space using DCT, we get motions that are a lot smoother overall. This is further amplified by using a GCN that is able to exploit spatial coherence between the last frames and the predictions. Overall, the cut between input and predictions is barely noticable, and the generated sequences look smooth to the eye.

5 DISCUSSION

Apart form the model we just described, a lot of our efforts went into using Transformers for human motion prediction. Transformers have recently given very good results in NLP [5] and we wanted to investigate how they can be applied to our task without drastically changing the architecture of the Transformer. We omit a description of the Transformer here and refer the reader to Vaswani et al.[19]. The main challenge to adapting the Transformer to our task is the vastly different problem setting. Where in NLP Transformers are mainly used as Sequence-to-Sequence models, we want to apply it to time series prediction. As we will discuss later, that has large implications as to how input and output to the Transformer should look like.

We explored many different approaches. We started with simple frame-wise prediction approaches using the input as given. We then tried combining the ideas from Mao et al.[13] with the Transformer, such as splitting the input and using DCT for temporal encoding. Additionally, inspired by [11, 16] we used teacher forcing, scheduled sampling and positional encoding.

In general, no Transformer model was able to achieve results similar to our multi-head attention model. In the following we want to discuss some common problems we found.

5.1 Problems of the Transformer

Models that predicted single frames, without any temporal addition often converged to a constant pose after some frames during prediction, similar to what Aksan et al.[2] reported. Mao et al.[13] further give some intuition to this behaviour as the attention mechanism fails to model motion direction accurately for frame-wise prediction, and similar poses can occur in different scenarios.

When we tried to implement positional encoding or apply concepts from Mao et al.[13], such as DCT, we encountered a different problem. The model was able to model the motion accurately, but there was always a mismatch between the last given frame and the first predicted frame. Mao et al.[13] explicitly implement a spatial and a temporal encoding, arguing the temporal encoding alone is not able to capture spatial dependencies between different joint coordinates or angles. Li et al.[12] argue Transformers are locality-agnostic, making them insensitive to local context, which can make the model prone to anomalies in time series. Based on these two observations we believe our models were unable to model the local dependency between the joints of the last frame and the predicted frames.

Another issue we encountered was the large size of the Transformer. In our experiments, the Transformers were relatively large in parameter size as well. This then impacted training times, and as a consequence the training takes a lot more time than the multi-head attention approach. Furthermore, the transformers are quite memory intensive, and the batch size has to be reduced, to not get memory- overflow errors. This only furthers the impact on training time. Because of this, we had to resort to data compression methods to be able to train the models in reasonable time on our machines. But in doing so, we might lose valuable data the transformer needs, which in turn leads to inaccuracies in the predictions.

The last problem we want to mention here is defining how to perform inference given a Transformer. It is not well defined how the output of a Transformer is to be interpreted for time series. In NLP the output of the Transformer is interpreted as a distribution over the embedding space. For time series the output could be interpreted either as predictions already in output space, or, alternatively, as a distribution, as proposed by Zimbres[22]. We have implemented several approaches, where the distribution option has given better results, but we were not able to define the best approach.

5.2 Comparison to our Model

Finally it is interesting to discuss why our simple multi-head attention model with a GCN performs so good compared to the Transformer models we implemented. The following reasons are just based on our intuition. First, our model makes a clear distinction between temporal and spatial encoding. As we said, that enables it to keep the spatial coherence while also being able to attend to positions in the past and model motion accurately using DCT. The Transformer model makes no such distinction. Further our model is comparatively simple and clear in design and thus understandable by humans, which from our experience is preferable to large models, that are hard to understand. Nevertheless we don't claim to have found the best Transformer model for our problem, such that there might be a simple variation to the Transformer that yields good results. Here again we want to mention Aksan et al.[2] that have used Transformers efficiently for long-term prediction of human motion.

6 CONCLUSION

We have shown that adding multiple heads to a motion prediction model using motion attention can lead to improvements. We suspect that this is due to the graph convolutional network having more information available to make its predictions. Additionally, compared to the commonly used recurrent neural networks, motion attention based modelling does not lead to a 'jump cut' between the last input frame and the first predicted frame. This is due to the fact that our approach encodes the spatial localities in the joints.

Furthermore, we found that implementing a Transformer for motion prediction is a very hard task, and although it could be possible to gain good results, the amount of work needed would surpass the scope of this course. This is down to the many open questions revolving around the transformer itself. The most glaring ones are how to define the inference inside the transformer, as well as adapting the transformer itself to a sequence-to-sequence task. Finally, the large model sizes when using a transformer model impact the training time to such a high degree, that training them becomes almost infeasible.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K.R. Rao. 1974. Discrete Cosine Transform. *IEEE Trans. Comput.* C-23, 1 (1974), 90–93. https://doi.org/10.1109/T-C.1974.223784

[2] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. 2020. A Spatio-temporal Transformer for 3D Human Motion Prediction. arXiv:cs.CV/2004.08692

[3] Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods for Training Generative Adversarial Networks. arXiv:stat.ML/1701.04862

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. arXiv:cs.LG/1904.10509

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:cs.CL/1810.04805

[6] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[7] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and Jose M. F. Moura. 2018. Adversarial Geometry-Aware Human Motion Prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[8] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer. arXiv:cs.LG/1809.04281

[9] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2Vec: Learning a Vector Representation of Time. arXiv:cs.LG/1907.05321

[10] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:cs.LG/1412.6980

[11] Natasha Klingenbrunn. 2021. Transformers for Time-series Forecasting. https://medium.com/mlearning-ai/transformer-implementation-for-time-series-forecasting-a9db2db5c820

[12] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2020. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. arXiv:1907.00235

[13] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. 2020. History Repeats Itself: Human Motion Prediction via Motion Attention. *CoRR* abs/2007.11755 (2020). arXiv:2007.11755 https://arxiv.org/abs/2007.11755

[14] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. 2020. Learning Trajectory Dependencies for Human Motion Prediction. arXiv:cs.CV/1908.05436

[15] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. arXiv:cs.CV/1705.02445

[16] Theodoros Ntakouris. 2021. The Time Series Transformer. https://towardsdatascience.com/the-time-series-transformer-2a521a0efad3

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[18] Yongyi Tang, Lin Ma, Wei Liu, and Wei-Shi Zheng. 2018. Long-Term Human Motion Prediction by Modeling Motion Context and Enhancing Motion Dynamics. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 935–941. https://doi.org/10.24963/ijcai.2018/130

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[20] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5797–5808. https://doi.org/10.18653/v1/P19-1580

[21] Mathieu Salzmann Wei Mao, Miaomiao Liu. 2021. History Repeats Itself: Human Motion Prediction via Motion Attention (GitHub Page). https://github.com/wei-mao-2019/HisRepItself/

[22] Rubens Zimbres. 2021. How I turned a NLP Transformer into a Time Series Predictor (PyTorch). https://www.linkedin.com/pulse/how-i-turned-nlp-transformer-time-series-predictor-zimbres-phd