Attention-based Motion Prediction: Implementation and Exploration - CS 7643

Gerald Richland
Georgia Institution of Technology
grichland3@gatech.edu

Anish Thomas
Georgia Institution of Technology
athomas382@gatech.edu

Yinyin Zhao
Georgia Institute of Technology
yzhao639@gatech.edu

Abstract

In the field of 3D motion prediction, attention-based deep learning approaches have been gaining popularity recently. One recent advancement is the Spatio-Temporal (ST) transformer model, an architecture that has the ability to incorporate spatial and temporal information using joint representation. The ST transformer model has quickly become a state-of-the-art solution to motion prediction. In this paper, we implement the ST transformer successfully and provide qualitative and quantitative analysis on its performance across different motions, representations, and teacher forcing ratios.

1. Introduction/Background/Motivation

As with many brain functions, humans oftentimes take for granted their ability to subconsciously process spatial information from their environments. For example, when we see a person walking on the sidewalk, we know this person is highly likely to keep walking if they are looking forward. When we see a person walking towards a closed door, we expect the person will turn the doorknob. It is natural for us humans to be able to predict other's movements based on how the other person is currently moving. Unfortunately, models do not possess this skill, so they must learn how to predict motion using complex, dynamic, bio-mechanical representations of motion.

3D Human poses are usually represented as sequences of joint angle positions over time. To predict motion, we are given a set of seed sequences, and the objective is to predict the next sequence of poses. Various model architectures have been implemented to understand the the bio-mechanical dynamics for all the joints at different positions in temporal sequences, with the first attention-based methods being introduced in 2020 [9],[1].Recent innovation in this space have been lead by advancements in 3D pose representation [2] and attention-based deep learning frameworks.

In 2015, Fragkiadaki proposed the Encoder-Recurrent-Decoder (ERD) architecture and the 3 layers of LSTM archetecture [3]. These architectures learn the representation of poses and the temporal dynamics by jointly training the encoder and decoder, yet suffer from the accumulation of errors over time and discontinuity in the first prediction. To improve the long-term results and solve the discontinuity issue, Martinez proposed seq2seq architecture with GRU and adopted residual connection and sampling-based loss in 2017 [10]. In these models, human motion representations are learned through the fully connected layer, and temporal dynamics are learned through the recurrent layers. While recurrent-based models are a natural choice for handling sequences of values, they cannot capture the long term time signal effectively [6]. In 2018, Li proposed the CNN based encoder architecture to better encode the long term input sequence [7]. These models take an auto-regressive approach, and are prone to accumulating errors over time. In 2020, Aksan proposed the Spatio-temporal transformer (ST transformer) [1] and to our knowledge, this architecture is state of the art. The ST transformer utilizes a decoupled temporal and spatial attention to capture the spatio-temporal information explicitly. Just like a vanilla transformer, it extracts information from the temporal sequence but, with the additional capability to extract information from the spatial sequence at each time frame. For each sequence in the batch, a vanilla transformer will attend the input temporal sequence while keeping the spatial information for all the joints inside one embedding dimension [12]. The ST transformer applies attention in a refined way, extracting both temporal and spatial information by introducing another dimension to incorporate each joint's spatial information. This extra joint dimension and spatial attention across joints is expected to enhance the model's motion prediction capabilities because it incorporates spatial and temporal information, instead of

just the temporal information.

Motion prediction is one of the most challenging and vital problems to solve in the field of computer vision and robotics applications. As discussed in [13], it has the potential to enhance human-computer interaction and ultimately move automation forward. Solving motion prediction is especially needed for advancing autonomous vehicles, as unmanned cars, while good at predicting the movements of other cars, still struggle at safely navigating areas with pedestrians present. Of course, it is much more difficult to predict the future movements of a person than a car, due to the wider range of motions that a human can perform. Overall, motion prediction is extremely challenging yet fruitful problem to solve, as the solution will serve as a basis for future technologies.

With the rising popularity of attention-based methods in 3D motion prediction research, we felt it would be appropriate to align our exploration with what has been done in other fields that heavily rely on attention-based techniques. In the field of natural language processing, a great deal of modeling is done using complex transformer architectures that consist of several layers. As is common in deep learning scenarios, when model complexity increases, the overall understanding of how underlying mechanisms interplay with each other decrease. Due to this complexity-interpretability tradeoff, research is constantly performed to better understand these systems. One example is the Bidirectional Encoder Representations from Transformers (BERT) model, a recurrent-based architecture that extracts features from blocks of text. BERT has had several papers written about it, discussing its various abilities and properties and as a result, researchers have found ways to apply BERT to different tasks and contexts[11]. Since the method is so novel, transformer architectures used for 3D motion prediction have not been well-explored. If architectures are not well explored, powerful insights that can be used to make innovations can be missed. Due to this perceived need for exploration, we decided once the ST transformer was successfully implemented, that we would pivot our stretch goals towards exploration. In our exploration, we compare the performance of two transformer architectures across different actions and representations. We are hopeful that our exploration can be used as inspiration for future work within the field of 3D motion prediction.

We used the AMASS DIP data set [8], which is a large database of human motion files from a variety of interesting sources, commonly used for human motion research [2] [5]. The full set of uncompressed AMASS data is around 40G and includes 11 different subsets. To control training time and data file size we are transferring with, we chose the AMASS BioMotion subset to work with. This subset is relatively large with a size of 10G. After preprocessing with the provided split text files, we got 12996 training sequences, 710 validation sequences, and 588 test sequences. From those sequences, we extracted 242 jumping, 228 sitting, 414 walking, and 111 treadmill motions from the Biomotion dataset. We selected these particular motions because we wanted to analyze cyclical, explosive and partial body motions.

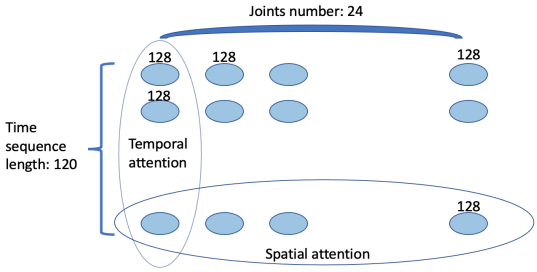

Figure 1. Illustration on spatial and temporal attention

2. Approach

The ST transformer model proposed in [1] is considered a state of the art approach. For this project, we expanded on Facebook's fairmotion starter code [4], implemented the ST transformer model, compared the ST transformer's performance to a baseline vanilla transformer, tuned relevant hyperparameters, and evaluated model performance across different teacher learning rates, motions and representations. Additionally, we generated body motion sequences images using the glut viewer from the OpenGL library, where the bio vision hierarchy (.bvh) files for the reference and the predicted motions are fed to a visualizer, which then uses the motion graphics to extract a sequence of video frames in a png image format.

2.1. Input sequence

Assuming a batch size of 32, the input source sequence fed into the model has a shape of (32 x 120 x 216). The first dimension is the batch size, the second dimension is the number of time frames in 2 seconds of source data, and the third dimension is the flattened result of number of joints multiplied by the dimension per joint in for the representation. UMASS DIP data represents a pose as 24 joints. If use the rotation matrix representation, then dimension for each joint is 9. Thus, our third dimension is 24*9=216.

2.2. Temporal positional encoding

The preprocessed input sequence flattens all joints to its third dimension. To differentiate each joint, we reshape the input's third dimension to (joint x embedding).

In our case, we reshape the third dimension from 216 to (24 x 9). Before sending the sequences to the attention layer, the embedding size of each joint is boosted up to 128. After this linear projection, temporal position encoding is added to the sequences. For temporal position encoding, we convert the dimension from (sequence x embedding) to (batch x sequence x joint x embedding) using two consecutive unsqueeze operations. Once the temporal position is encoded, we branch out the sequences into two branches for temporal and spatial attention separately. Afterwards, we define the attention layer for each attention branch as shown in Figure 2.

2.3. Temporal Attention Layer

In the temporal attention layers, the 24 joints are not attended at the same time. Instead, each joint only attends to its own temporal sequences. There is no temporal attention from one joint to another between different time frames as shown in Figure 1. This indicates the temporal attention is refined to a local joint, and we extract information between different time sequences for one joint and predict its behavior based on its own recent history. Information regarding to how each joint affects its neighbor joints is extracted during the spatial attention step. In each time frame, spatial attention is performed on all the joints. Once completed, temporal and spatial attention are added together, with a residual connection.

For temporal attention, we must reshape the spatial joint dimension and batch dimension into one new 'batch' dimension to properly feed into nn.MultiheadAttention module where the required dimension are (sequence length x batch x embedding). Note that in torch version 1.4.0, there is no batch first option in nn.MultiheadAttention, so we we transpose the first two dimension to meet the order requirement. For temporal attention, we need to pay particular attention to the mask input in the multihead attention module to make sure the time masking is effective during the attention. This mask is not required for the spatial multihead attention input. The output of the multihead attention with dropout was added to the input and normalized. To sum the temporal attention and spatial attention, the output sequence of the temporal attention transformer must be reshaped back to the shape it was when it initially branched out.

2.4. Spatial-Attention Layer

The process for building the Spatio-Attention layers is similar to the one that was performed in the temporal attention layer, except now we group the time sequence dimension with the batch dimension to form a new 'batch' dimension, and we don't feed the mask into the multihead attention module.

Since the output shapes of the temporal attention layer and spatial attention layer match, we can sum them with ease. Once summed, the output is fed to a feedforward layer with two linear layers and ReLU activation. In the first linear layer, the embedding dimension was doubled from 128 to 256 and projected back to 128 in the second linear layer. The output of the feedforward layer with dropout is then normalized with a residual connection. The power of this layer stems from not only its design but, also its ability to have multiple layers.

2.5. Next Motion Prediction

The prediction is done in an auto regressive approach [1]. After the stackable ST transformer layers, the output is projected back to the original embedding dimension (9 for rotation matrix representation) for each joint and reshaped to (batch x time sequence x flattened joints embedding). There is a residual connection between the output and the last source time frame. The output after the residual connection is the joint prediction of the next time frame. The joint prediction is generated one by one. During the training phase, we start from the source frames, and predict the first time frame in the target. Next, we remove the first time frame in the source and replace it with the first time frame in the target. These values are then used to predict the second time frame in the target. We repeat these steps until all time frames in the target are predicted. During the evaluation phase, the predicted output time frame is concatenated with the source sequence instead of the target time frame.

To implement the predictions, we make a data chunk by concatenating the source frames and the target frames. Then, for each iteration, we chop a window slice at the length of the source frames with stride length of one and make this data slice our source frames. During the training phase, the teacher forcing ratio starts from 1 and gradually decreases to 0.

2.6. Challenges and Solutions

As we worked though the proposal phase of our project, we anticipated three main issues. The first issue we expected was difficulty in the implementation of the ST transformer model. The other two issues involved our stretch goals of looking at new loss and evaluation metrics and integrating them into the model. As we worked on the stretch goals, we realized that that integrating the metrics into the model required a large time commitment that was not feasible within the project timeline. As a result, we decided to shift our focus towards providing quantitative and qualitative exploratory analysis.

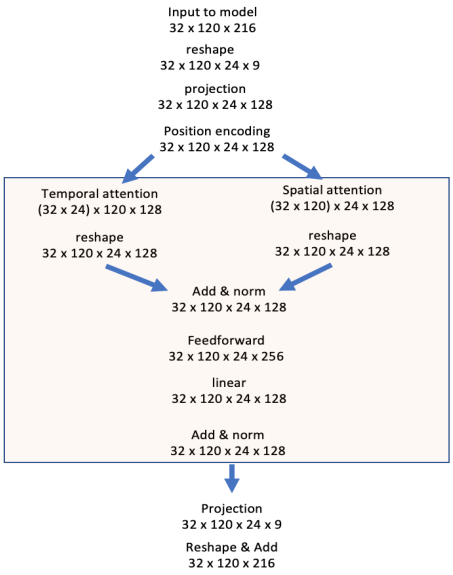

Figure 2. ST model architecture

We used Facebook's fairmotion framework [4] as our starter code and added the ST transformer model to the framework. To add new model files to the framework, we need to install it using setup.py instead of using pip. The PyTorch version installed using pip is 1.4.0, but the PyTorch version installed with setup.py is 1.6.0. It turned out the fairmotion repository was updated to work with PyTorch 1.6.0 but, it was not completely configured, causing issues when running the base transformer. We resolved the issue by manually changing the PyTorch version back to 1.4.0 in setup.py and used that version moving forward.

Once the ST transformer model was successfully implemented, we attempted to run the model using parameters from the original paper, but were immediately faced with out-of-memory errors. In order for the model to run on Google Colab's Tesla P100-PCIE-16GB GPU, we had to lower the hidden dimension, joint embedding size, and batch size to avoid running out of memory. In an effort to train the ST transformer model with parameters that were closer to the ones used in the paper, we obtained temporary access to two 32GB NVIDIA GPUs. We chose the largest parameters that the GPU could handle and trained for approximately 12 hours. Unfortunately, these parameters are still relatively small compared to those in the original paper, but the superior performance of the model is noteworthy enough to proceed forward.

3. Experiments and Results

For the experiment portion of our project, we split the focus of our analysis into 3 areas: representations, motions, and teacher forcing rate. Sections 3.1 and 3.2 focus on the teacher forcing rate analysis, while sections 3.3 and 3.4 focus on the motion and representation analysis.

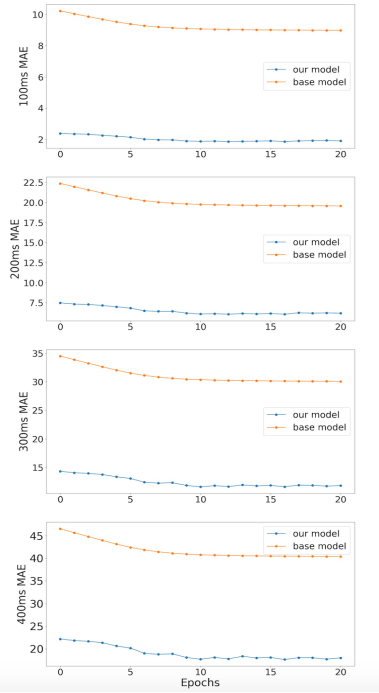

Figure 3. MAE comparison between base model and our model

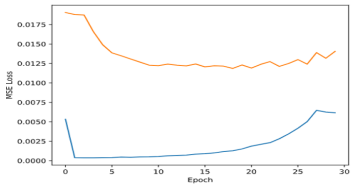

Figure 4. Train loss vs. validation loss for ST transformer

3.1. Comparison with Baseline

We chose the baseline model to be a vanilla transformer model. To compare the performance between the base model and our ST transformer model, we tried to match the original paper's parameter as shown in Table 1, but did not have the computational resources needed for training such a large model. The hidden dimension and joint embedding values in our model are specified for each joint, which is why they are much smaller compared to the base model(vanilla transformer).We used the Adam optimizer since that is what was used in the original paper. The loss function used in the model is a standard mean squared error (MSE). The metric used for evaluation is the Mean Angle Error (MAE) [10]. Figure 3 shows the MAE at different time periods (100ms/ 200ms/ 300ms/ 400ms) over 20 epochs for the baseline model and our model. We

Hyper-parameters	Base model	Our implementation	Original Paper
architecture	vanilla transformer	ST transformer	ST transformer
batch size	32	32	32
hidden dim	256	32	256
learning rate	0.001	0.001	-
joint embedding	256	16	128
number of heads	2	2	8
number of layers	2	2	8
angle representation	rotmat	rotmat	rotmat
window size	120	120	120
optimizer	Adam	Adam	Adam

Table 1. Hyper-parameters for comparison

Euler angle MAE	Base model	Our implementation	Original Paper
at 100ms	8.79	2.01	-
at 200ms	19.15	6.35	-
at 300ms	29.41	11.89	-
at 400ms	39.51	18.07	13.70

Table 2. Euler angle MAE comparison

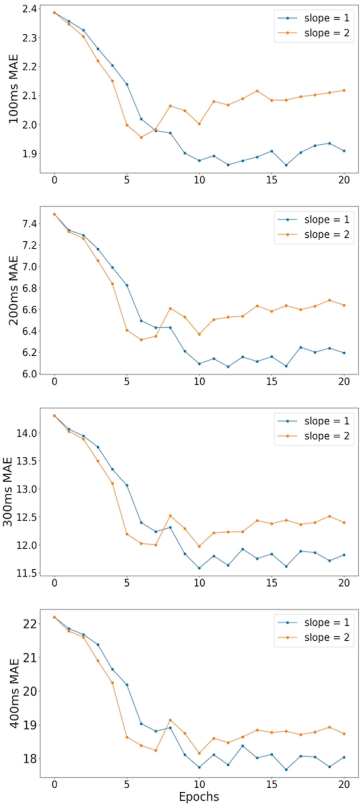

Figure 5. MAE comparison for different teacher enforcing ratio

observe that for all evaluation periods, the ST transformer model outperforms the baseline. This demonstrates the

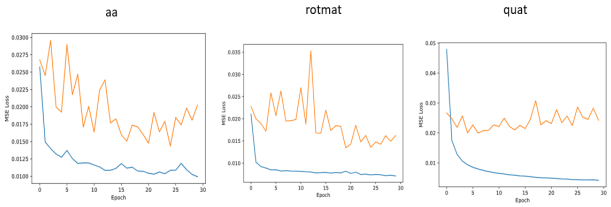

Figure 6. Vanilla Transformer Models used in Section 3.3 & 3.4

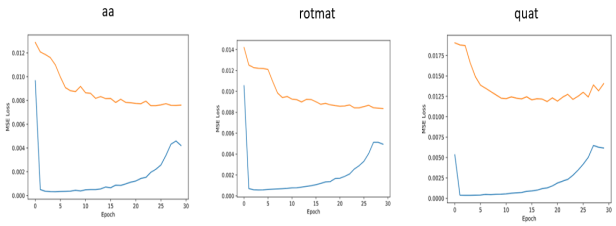

Figure 7. ST Transformer Models used in Section 3.3 & 3.4

superior performance of the ST transformer model over vanilla transformers.

Table 2 lists the Euler Angle MAE at 100ms/ 200ms/ 300ms/ 400ms for base model, our implementation, and model in original paper. Note that for the Euler Angle in original paper, the data point was taken at two attention layers from Fig. 6 in [1] to be consistent with our settings. For the MAE of Euler Angle at 400ms, our implementation was worse than the original paper, but much better than the base model. With the same number of attention layers

Model	Representation	MAE 100ms	MAE 200ms	MAE 300ms	MAE 400ms	Overall MAE
ST Transformer	aa	1.76	5.88	11.51	17.87	9.25
ST Transformer	quat	1.85	6.07	11.78	18.22	9.48
ST Transformer	rotmat	1.74	5.73	11.15	17.33	8.99
Vanilla Transformer	aa	3.33	8.80	15.42	22.81	12.59
Vanilla Transformer	quat	8.28	18.09	27.90	37.65	22.98
Vanilla Transformer	rotmat	3.42	8.59	14.75	21.77	12.13

Table 3. Performance Across Representations

used, the main difference between our implementation of the ST model and the original paper lies within the number of attention heads, hidden dimension, joint embedding dimension, and the size of the training set. We tried to increase the number of heads, but our GPU memory resources cannot support it. Our choices for hidden dimension and joint embedding dimension are limited too due to the same computational resource constraints. In the original paper, the authors used the entire AMASS dataset. As mentioned in the introduction, due to the challenges of working with 40GB of data we chose to only use the Biomotion folder, which still has 10 GB of motion data. In later sections, we explore experiments on teacher enforcing ratio, learning rate, joint angle representation to improve our model performance.

Figure 4 shows the training loss and validation loss of our model. The training loss drops quickly to a very small value at the first 2 epochs, then slowly increasing over the following epochs. The validation loss keeps decreasing even the training loss is rising gradually. The rise of the training loss is rare from our previous experience, and we reason this may be caused by the teacher enforcing ratio is decreasing. As the teacher enforcing ratio decreases, the model is gaining a little more training loss, but the validation loss befits from the decreasing teacher enforcing ratio. The validation loss starts to increases around epochs 20, indicating the model starts to overfit.

3.2. Teacher Enforcing Ratio

We explore two different slopes of teacher enforcing loss decrease in our model and compare their Euler joint MAE in Figure 5. In both cases we decrease the teacher enforcing ratio linearly over epochs at different slopes. For slope 1 we have a linear decrease of teacher enforcing ratio to 0 at the end of training epochs. For slope 2 we doubled the decreasing rate, and the teacher enforcing ratio decreases to 0 at the middle of the training epochs. From Fig 7 we can see that at all time frames, a faster decreasing of teacher enforcing ratio results in higher MAE. This indicates a slowly decreasing teacher enforcing ratio gets the model to a more optimized position, helps the model to learn more from the training, and eventually reach to better performance.

3.3. Performance Across Representations

For the analyses in this section and section 3.4, we investigated how the vanilla transformer and ST transformer perform across different motions and representation formats. To create the dataset, we utilized fairmotion's preprocessing script[4], which is based on methods proposed in [2]. The function converts motion objects into 2000ms sequence windows and 400ms target windows, representing them as matrices, before finally splitting the data into validation, training, and testing sets. This preprocessing step was performed three times for each dataset to extract the axis angle (aa), quaternion (quat), and rotation matrix (rotmat) representations of the data. Once data is gathered, we trained the ST and vanilla transformer on each representation format for 12996 Biomotion sequences, using Google Collab Pro's Tesla P100-PCIE-16GB GPU. The training time was on average 3 hours for the vanilla transformer and 12 hours for the ST transformer. To find the right set of "light" parameters that balanced model performance and computational resources needed for training, there was a great deal of time invested in the tuning process. The train plots for models used in this section and 3.4 are figures 5 and 6. We observe similar training curves for the SPT models, where there is large decrease in training error during the first epoch, followed by gradual increase, while validation error just gradually decreases. This indicated that our model overfit at the beginning and then began correcting itself. The vanilla transformer's training curves were smooth except for rotation matrix, while the validation curves jumped around. We explored different learning rates but, could not find one that improved the validation loss. We believe that if we could train on larger batches, the issues presented could be mitigated because training on larger batch sizes allows the model to learn patterns that are more representative of the overall dataset.

Creating a model for each angle axis, quarternion, and rotation matrix format yields 6 models for this analysis. We evaluated our Biomotion trained transformers on a Biomotion test dataset that consists of 558 sequences.

The evaluation metric used was Mean Angle Error at time steps 100ms, 200ms, 300ms and 400ms. Mean Angle Error(MAE) is defined in the fairmotion repository as the Euclidean distance between predicted and reference Euler angles averaged over all joints[4].The parameters for these models can found in table 3 and results of the analysis can be found in in table 5.

As expected, the ST transformer outperformed the base transformer across all representation formats and evaluation periods. For the base transformer, we noticed that the best performance at 100ms is the angle axis representation while the ST transformer's best performance is the rotation matrix representation. This observation vanishes after 100ms, where the rotation matrix representation begins consistently outperforming quaternion by a large scale and axis angle by a smaller scale. This analysis provides two interesting observations. First, while the rotation matrix representation has a better overall prediction across the 400ms interval, the angle axis representation offers similar performance, while being a lighter computational load, considering the angle axis matrix version is (3x3) and the rotation matrix version is (9x9). Second, across all evaluation periods, the ST transformer average MAE is much lower than the vanilla transformer, implying the ST model is much more adaptive to different representation formats when compared to the vanilla transformer.

3.4. Performance Across Motions

In this analysis, we investigated the performance of the ST transformer and the vanilla transformer across the following motions: jumping, walking, sitting, and treadmill. As mentioned in the introductory section, we selected the motions so that we had a mix of cyclical, inactive, and explosive motions. We used the matrix rotation representation for this analysis because the results of the previous section signaled to us that the rotation matrix representation provides the most stable performance across evaluation periods. The preprocessing and training information for the models used in this section can be found in the first paragraph of the previous section.

Our first hypothesis is that the lowest average MAE across the evaluation periods would be walking and treadmill, since these motions have a cyclical pattern. Our second hypothesis is that the MAE for partial body motions like sitting would vary across models since the ST transformer model works on a per joint basis, while the vanilla transform does not. Our third hypothesis is that the explosiveness of actions like jumping, result in the highest average MAE values across the evaluation period because of speed and variation among explosive actions. Finally, we expect that the ST transformer consistently outperforms the vanilla transformer across all actions. Results of this analysis can be found in Table 4.

As expected, the ST transformer outperformed the vanilla transformer for all motions, with the vanilla transformer having 30-300% higher MAEs than the ST transformer, supporting our final hypothesis. For each model, treadmill was the motion with the lowest MAE overall and at each evaluation point, supporting our first hypothesis. We expected this superior performance to carry over for walking motions, as both motions are cyclical. This expectation held for the vanilla transformer, but for the ST transformer, we observed that the sitting motion had the second lowest MAE overall and at each evaluation point, opposing our second hypothesis. Further, we observe that for the vanilla transformer, MAE values for sitting motions were on average 225% higher than ST transformer. For reference, the different among other actions ranged between 127% and 153%. We attribute this outcome to the fact that if a person is sitting, then the ST model will learn that the leg joints are not moving, so it can have extremely low MAE performance in some joints, reducing the overall MAE to the value that we observed. While not conclusive, the results supports our second hypothesis because the main difference between the two architectures is the spatial component applied to each joint in ST transformer.

Jumping had the highest MAE for all models overall and across all evaluation periods, supporting our third hypothesis. This result is expected because explosive action like jumping vary widely person to person and the pose before jumping can be confused with several other motions like going to sit down. Interestingly, we observe that the vanilla transformer overall MAE is only 124% higher than ST transformer. We interpret this as both models being unable to properly learn the action because explosive motions like jumping are usually strenuous and done with a set motivation. In other words, we think both transformer models perform the worst on explosive actions because neither model can incorporate the contextual information needed to identify cues before an explosive movement is made. Overall, our findings supports 3 of our 4 hypothesizes. The only hypothesis not supported by these results is the first one, where we underestimated the impact of several joints not moving, leading to a lower overall MAE at each evaluation period.

4. Motion Visualization

A visual comparison of the various types of motions namely biomotion, jumping, sitting, treadmill walking and general walking were performed. Here we compare the results of testing the transformer (TR) and spatio-temporal transformer model developed. The three types of angle representations used are axis angles (aa), quaternion (quat) and rotation matrix (rotmat). A total of 143 frames were generated each time using a BVH visualizer and compared across

Model	Motion	100ms	200ms	300ms	400ms	Overall
ST Transformer	Biomotion	1.74	5.73	11.15	17.33	8.99
ST Transformer	Jumping	2.68	8.46	16.85	25.42	13.35
ST Transformer	Sitting	1.17	4.14	8.58	13.98	6.97
ST Transformer	Treadmill	1.06	3.72	7.36	11.32	5.87
ST Transformer	Walking	1.48	4.88	9.56	14.49	7.60
Vanilla Transformer	Biomotion	3.42	8.59	14.75	21.77	12.13
Vanilla Transformer	Jumping	3.94	10.42	19.89	30.41	16.16
Vanilla Transformer	Sitting	3.73	9.37	16.02	23.48	13.15
Vanilla Transformer	Treadmill	1.98	5.22	9.52	14.81	7.88
Vanilla Transformer	Walking	2.72	7.40	13.34	20.01	10.87

Table 4. Mean Angle Error (MAE) Performance Across Motions

Hyper-parameters	Vanilla Transformer	ST Transformer
batch size	16	16
hidden dim	512	16
learning rate	0.001	0.001
joint embedding	-	16
number of heads	1	1
number of layers	1	1
window size	120	120
optimizer	Adam	Adam

Table 5. Transformer Hyper-parameters for Sections 3.3 & 3.4

the two models and the three different angle representations.

Across the five motions, the quat had the least duration of correct motion predications up to around 120 frames and the body is movement is completely frozen in the later frames. The ST model with rotmat has an overall better predictions of motion than transformer model with aa, quat or rotmat angles. Except for the sitting motion, the ST model made the correct predictions throughout the entire frame length. For the sitting motion, the transformer model had the hands either moving to behind the body (for aa) or the hands falling into the legs (for rotmat) causing an overlap. But while there was no overlap of body parts for the ST model, the prediction of body movement was incorrect for the sitting motion especially at the end of the frames generated. As part of this work, the comparison of the motion result is performed via manual visual comparison. Any future work could incorporate a thorough visual analysis using some evaluation tools and advanced techniques. The sample motions frames are provided in the appendix.

5. Conclusion

In this paper, we discussed our approach for implementing the ST transformer model and provided quantitative and qualitative evidence that the ST transformer model , even with light parameters, can outperform a vanilla transformer by a large scale. In terms of achieving our primary goal of implementing the ST transformer, we succeeded. As for our stretch goals of exploring different evaluation methods and loss function, we decided to pivot away from these goals because we were not confident that they could be implemented before this project is due. We pivoted towards exploration work and from our perspective, the results are noteworthy enough to constitute further research into the ST transformer model. If given more time, we would have tried to train the ST transformer with parameters that match the original paper. Additionally, we would have done deeper exploration, analyzing how different pieces of the model learn together. To handle the overfitting we experienced during ST transformer training, we would have attempted to utilize a different loss function in the hopes that it would correct the overfitting. In conclusion, we succeeded at our primary goal and succeeded at providing explorative insight into the performance of the ST transformer model.

6. Work Division

Summary of contributions are provided in Table 6. Our submission folder contains our code, best model, sample visualizations, and model configurations. We would have liked to upload all of our output but, gradescope has a 100MB file limit, so we provided what we felt was the most relevant.

References

[1] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction. *arXiv:2004.08692*, 2021. 1, 2, 3, 5

[2] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured prediction helps 3d human motion modelling. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7144–7153, 2019. 1, 2, 6

[3] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4346–43548, 2015. 1

[4] Deepak Gopinath and Jungdam Won. fairmotion - tools to load, process and visualize motion capture data. Github, 2020. 2, 4, 6, 7

[5] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser learning to reconstruct human pose from sparseinertial measurements in real time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 37(6):185:1–185:15, Nov. 2018. 2

[6] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 156–165, 2017. 1

[7] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5226–5234, 2018. 1

[8] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, Oct. 2019. 2

[9] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. History repeats itself: Human motion prediction via motion attention. *CoRR*, abs/2007.11755, 2020. 1

[10] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683, 2017. 1, 4

[11] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how BERT works. *CoRR*, abs/2002.12327, 2020. 2

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017. 1

[13] Borui Wang, Ehsan Adeli, Hsu kuang Chiu, De-An Huang, and Juan Carlos Niebles. Imitation learning for human pose prediction. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7124–7133, 2019. 2

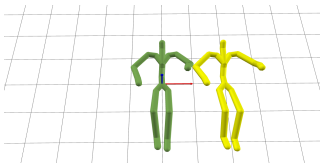

Figure 8. SP - Jumping

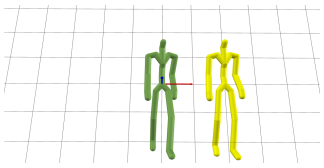

Figure 9. SP - Treadmill

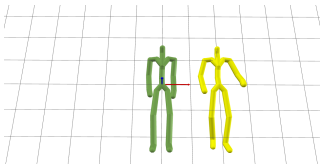

Figure 10. TR - Jumping (aa)

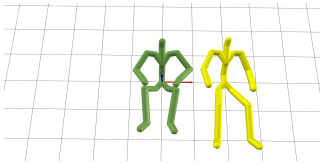

Figure 11. SP - Sitting

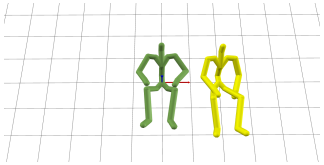

Figure 12. TR - Sitting (aa)

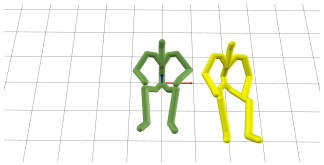

Figure 13. TR - Sitting (rotmat)

7. Appendix

7.1. Visualization of Motion Frames

Above are few of the sample frames from the motion visualizations generated. In each of the images, the first body figure is the reference motion and the second one is the corresponding predicted motion for a single time frame.

Student Name	Contributed Aspects	Details
Yinyin Zhao	Data Creation, Implementation and Analysis	Scraped the dataset for this project, implemented and trained the ST model (section 2.1, 2.2, 2.3, 2.4, 2.5 and transformerSpatialTemporal.py), analyzed the result (3.1, 3.2).
Anish Thomas	Data Analysis, Environment Setup, Testing, Visualization	Researched on human motion, analyzed the data set and prepared the environment, extracted model test results (viz/tests) and motion data, and generated visualizations (Section 4.0 and bvh-visualizer.py).
Gerald Richland	Data Creation, Implementation, Analysis, Editing	Prepared data for experiments (3.3 & 3.4), implemented the testing script for ST transformer, tuned/trained/tested light transformer models for experiments (3.3 & 3.4), revised several portions of the paper, wrote about our challenges (2.6), code can found in the submission folder, at Code\Colab_Notebook\

Table 6. Contributions of team members.