Motion Prediction
Next Move: CS 7643

Eren Jaeger *, Armin Arlert*, Mikasa Ackerman*,
Georgia Institute of Technology
{REDACTED}@gatech.edu

Abstract

The ability to forecast human motion is useful for a myriad of applications including robotics, self driving cars, and animation. Typically we consider this a generative modeling task, where given a seed motion sequence, a network learns to generate/synthesize a sequence of plausible human poses. This task has seen much progress for shorter horizon forecasting through traditional sequence modeling techniques; however longer horizons suffer from pose collapse or converge to static pose. This project aims to explore recent approaches that can better capture long term dependencies. To this end, we implemented the state of the art paper, A Spatio-temporal Transformer for 3D Human Motion Prediction [2], which has no known public implementation and extended it. In this report, we document our experiences and challenges, describe our experimental setup, report our results on the implementation and extensions and compare them to other baseline methods such as RNN and seq2seq methods, and interpret our findings.

1. Introduction/Background/Motivation

Forecasting Human motion is useful in settings from safety to content creation. Autonomous cars may predict whether a pedestrian will run across the road, or a designer may speed up their creative process animating their characters. This problem also offers a rich history of potential solutions making it an interesting domain to explore. Essentially, given the seed sequence of human motion, we want to predict their future motion. Commonly, the seed sequence may be 120 frames (2 seconds) and we would like to predict the next 24 frames (0.4 seconds). Our primary goal was to implement the state of the art paper: *A Spatio-temporal Transformer for 3D Human Motion Prediction* [2] which uses both spatial and temporal attention mechanism [16] to achieve SOTA performance. There is no public implementation of this paper and our aspirations were to implement every piece of the ST Transformer ourselves, endowing us with deeper of understanding of how self-attention and transformers can be applied to solve problems with inherent hierarchy where where both spatial and temporal dependencies exist. Further, this work could be relevant to any domain in which there are variables with temporal covariance or structured dependencies between outputs. Our ambitions included a detailed comparison with baseline approaches, extensions to increase parameter sharing or lower the temporal footprint, and thorough analysis of the st-attention mechanism from a structured prediction and computational perspective (e.g. affect of the number of spatio-temporal attention layers). We included experience report of ideas that did not pan out, either due to time or resources in the final section.

The research into human motion modeling and prediction has a long history outside of deep learning. Human motion was considered so diverse that even with a large amount of data, adding prior knowledge to the model was a given [15]. Human pose estimation and motion has been directly modeled Hidden Markov Models (HMM) [12] and Gaussian Processes (GP) [17].

In 2015, Fragkiadaki et al. [5], introduced an Encoder-Recurrent-Decoder (ERD) network uses an encoder to encodes poses as input to an LSTM and uses a decoder to regenerate the poses from the output of the recurrent network. They also evaluated evaluated a network with 3LSTM layers (LSTM-3LR). They noticed a trade-off between the two networks. ERD lacks smoothness and LSTM-3LR does well on short term prediction but converges to mean pose. In [11], a method to transform spatio-temporal graphs (st-graphs), which can be used to model interactions with the environment, into RNNs in an architecture called structural-RNN (S-RNN). This method presupposes such a graph exists for the interaction. GRUs with residual connections [14] have been added to standard RNNs and shown to outperform baseline.

The authors of our target paper, have also done previous research [3] that showed that in addition to the previous

*Random order. All authors made equal contribution.

mentioned issues, previous methods do not do well on larger AMASS dataset which is 14 times larger than often used at the time H3.6M [10] dataset. Their method introduces a structured prediction layer (SPL) that provides hierarchical structure corresponding to the joints to generate the output of the motion. All the previous methods struggle on is prediction spanning longer time horizons, which their current work (which we discussed in various parts of this report), aims to alleviate. Other areas that require further research to make progress, is how to evaluate the results of predictions. Surely, even a human watching motion can imagine many future prediction that could be valid. Previous research have addressed this by performing user studies. Other areas that would move the state of the art is in a better metric, rather than say MSE between predicted joints and their true value. We have considered adding loss to joint lengths and other constraints on relative joint structures.

AMASS (Archive of Mocap as Surface Shapes) [13] seeks to unify existing human motion capture datasets into a single repository. Each joint in the Motion capture (Mocap) data can be represented using axis angle (aa), a rotation matrix (rotmat), or quaternion (quat) and the fairmotion library [6] was used to preprocess raw motion sequences into source and target for training, validation and test sets. For all datasets, we introduce extra pre-processing that removes joints which are unused from the data rather than the default behavior of fairmotion which leaves unused joints as identity matrix. We used AMASS-DIP [8], which was created and hosted by the Max Planck Institute for Intelligent Systems as the final source of data for our results and experiments. Like [1], we used rotation matrix representation resulting in the joint representation $M \in \mathbb{R}^9$ for each of the 15 joints. This is an older variant of AMASS totaling over 9 million motion frames (see Table 4 in appendix). The large volume of this dataset presented a challenge for our approach, therefore we carved out three smaller datasets from AMASS-DIP, which are denoted Small, Medium and Large. Small corresponds to the ACCAD dataset accounting for 2% of AMASS-DIP. Medium corresponds to ACCAD plus subsampled BioMotion corresponding to 5%. Finally, Large is ACCAD plus all of BioMotion, making up 25% of AMASS-DIP. We focused on ACCAD and BioMotion, because of the clear identification of each motion type in motion file names; such as walking, running, kicking, etc. used for further analysis.

Initial work was performed on the Small dataset to optimize and interpret after which we moved onto the medium and large datasets. The size of the dataset alongside the computational complexity of the ST Transformer constrained us to only one run on our "large" dataset which still took over 100 hours. It should be noted that in [1] Aksan et. al. express concerns on model performance when using datasets smaller than AMASS-DIP (e.g. H3.6M [10]).

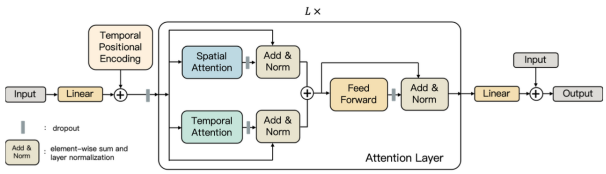


Figure 1: Architecture of patio-temporal transformer [1]

2. Approach

The Tensorflow code for [1] has not been made available yet, and to our knowledge ours is the first available authentic replication of this model. The architecture consists of an embedding layer with positional encoding followed by L spatio-temporal layers and a final linear layer that re-projects the embedding back to the joint representation space.

2.1. Embedding Layer + Positional Encoding

The input $x \in \mathbb{R}^{T \times N \times M}$ is passed through an embedding layer for each joint $n \in N$ to output an embedding $E \in \mathbb{R}^{T \times N \times D}$. A positional encoding (cosine and sine) is added to this embedding and it is passed to the attention block feeding both spatial and temporal attention sublayers. The purpose of the positional encoding is give information to the later parts of the network about the relative position of the poses to each other. This would allow the attention mechanism to determine the significance of the relative ordering. For example, the attention mechanism may deem looking only a certain number of frames useful for determining what happens in the next time step.

2.2. Spatio-temporal Attention Layer

2.2.1 Temporal Attention Layer

the embedding E for each joint is passed into the temporal attention layer where attention is applied similar to the self-attention formulation described by [16] where Queries, Keys, and Values are extracted from the embedding, attention weights are computed and applied to the Values. In language tasks we have a single sequence on which we apply attention to select which words from the vocabulary to attend to, however an interesting aspect to the motion prediction problem is that we have N (15 for our case) streams of sequences for which we must apply multi-head attention to independently. For a given motion frame, we take the embeddings E for each joint and pass them through a temporal attention layer independently, creating an output $\bar{E} \in \mathbb{R}^{T \times N \times D}$. Note that since we are using a generative prediction approach we must apply masking (a triangular causal mask was used) in order to prevent the model from looking at future values for its prediction. A visualization of the attention weights without masking 5 is discussed further in the appendix C.

2.2.2 Spatial Attention Layer

In parallel to the temporal attention layer, the embedding E for each joint is passed into a spatial attention layer where attention is applied spatially across a frame for all joints rather than temporally. A modified version of attention is applied to leverage parameter sharing, concretely keys and values are shared between joints while each element of the query is extracting independently for each joint. This spatial attention is applied at each timestep independently thus the output embedding $\tilde{E} \in \mathbb{R}^{T \times N \times D}$.

2.2.3 Feed forward layer

The output embeddings $\bar{E}$ and $\tilde{E}$ are summed and fed into a 2 layer feed-forward network. The FFN does not change the of the input-output pair however projects the summed embeddings into a larger representation space in the hidden layer and applies a nonlinearity (ReLU). The output E_l represents the output of the spatio-temporal attention layer and L layers can be stacked like decoder blocks to progressively refine the embedding.

2.3. Output Projection Layer

After refining the embedding through L spatio-temporal attention blocks, a linear layers takes the final embedding and reprojects it back into joint space for each joint. ie. we have N output projection layers that take the corresponding joint embedding and project it back. The output of these parallel embeddings represents the prediction

2.4. Residual Connections and LayerNorm

Note that residual connections are applied for both the spatial and temporal attention sub-layers. Moreover, residual connections are employed in the output layer between the raw input and the output.

In addition LayerNorm [4] is applied for all layer normalizations. Activation statistics are useful to improve the learning dynamics of neural networks and Transformers are no different. Recall that in BatchNorm [9] is commonly applied in ConvNet's where statistics used for normalization are computed across the batch and spatial dims making it sensitive to the batch size. LayerNorm however is independent since it is computed across channels and spatial dimensions (in our case both temporal and spatial).

2.5. Teacher/Student & Curriculum Learning

For sequence prediction tasks for generative models, we are factorizing the probability distribution over motions as a product of conditionals over motion frames and training through maximum likelihood. As such the sequence has to be predicted one at a time. Usually the prediction p_{t-1} passed in as part of the input X_t to predict p_t (Student Forcing). Teacher forcing is an approach where during training the ground truth is fed with input rather than the prediction from the previous timestep [18]. This helps convergence as predictions early on in the network are not very good and we do not want to compound this error on subsequent predictions. Consequently ground truth predictions are not available during inference creating "exposure bias" [7]; an unbalance between how the model is provided data at training versus test time.

3. Reducing Computational Load

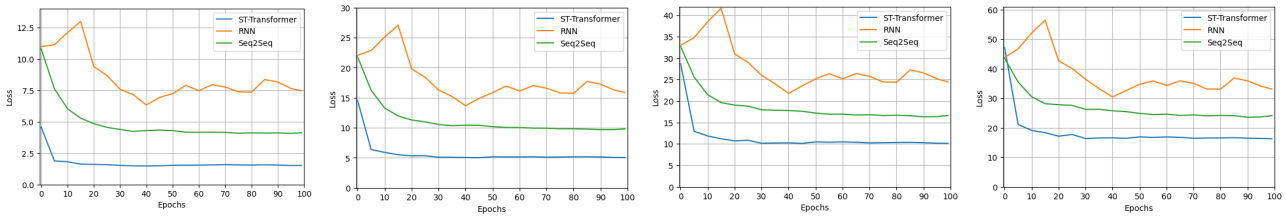
Given that GPU memory was one of the largest challenges faced in implementing this new attention architecture, we explored two techniques to reduce memory load.

3.1. Joint Mirroring

The first technique that we considered was the sharing of parameters between joints that have an opposite counterpart (e.g. left and right shoulder) on the embedding, temporal, and output linear. We name this technique joint mirroring and apply it to the following joints: hip, knee, collar, shoulder, and elbow. Theoretically it makes sense that the weights for these joints should match as any difference, bar slight human tendencies towards right handed gestures, is likely the result of training sampling. Not only do these shared weights reduce the number of trainable parameters but they provide the added benefit of partially training in both directions with each training motion as each joint also learns important feature representations from its counterpart. Further work could be done to mirror the spatial attention layer, but computational trade-offs would have to be made as weights would have to be truly mirrored (e.g. same weights between left shoulder and left elbow as between right shoulder and right elbow) as opposed to shared. Results indicate a slight improvement over baseline at all validation frames.

3.2. Temporal Downsampling

We also implemented temporal down-sampling via convolutional layers with one biasless, non-shared kernel per joint that is strided across each joint embedding to downsample time-steps. This could be replaced with a pooling layer if one were seeking to minimize trainable parameters, but implementation with a convolutional layer allows for the model to learn a slight temporal weighting of time-steps and was implemented to allow for further study of the possible computational benefits of skipping some input entirely through dilation. An inverse convolutional layer works to upsample the stacked output from the temporal layer back to the necessary dimensions to match the skip connection and spatial input. We name this technique temporal sampling in the chart. This technique reduced the length of the time steps to 25% of its original size, from 130 to 40, which



(a) Validation MAE@6 (b) Validation MAE@12 (c) Validation MAE@18 (d) Validation MAE@24

Figure 2: Validation MAE Comparison for St-Transformer, RNN and Seq2Seq

lead to a 65% decrease in the memory requirement on the GPU while validation error approximately 3%.

4. Experimental Setup and Results

4.1. Loss function and Training Scheme

The loss function applied was the per joint $\mathcal{L}2$ distances on the rotation matrices per [1], defined as $\mathcal{L}(P, tgt) = \sum_{t=2}^{T+1} \sum_{n=1}^{N} ||j_t^n - \hat{j}_t^{(n)}||_2$

Aksan et al. offer some insight regarding training but it can be interpreted several ways. Especially as some of these interpretations create out-of-memory (OOM) issues due to the size of the computational graph increasing with timesteps. They state that their approach is not seq2seq and use the entire sequence for training. In addition they use an auto-regressive approach by predicting the next pose given the frames observed so far. An approach commonly applied in generative language models which is clearly factorizing the probability of a motion into conditional probabilities over frames (individual poses), however OOM and correspondence with Facebook suggests a hybrid training approach. More detail can be found in the appendix regarding optimizer D and training schemes E.

Self-Attention is a mechanism that's been postulated to approximate convolution in the limit [16], meaning that it is a more general method of structured prediction that does not have the locality architectural bias. Our results show that in the teacher forcing case, the attention weights 4a amount to the model learning to do some type of local convolution which is an interesting result. In addition, when posed with the task of correcting for error in its prediction to reduce exposure bias, the model learns to look further back [4b 4c] as perhaps the recent prediction may be suspect so previous predictions may hold more information.

4.2. Spatio-Temporal Transformer vs. Baselines

We compare ST Transformer to RNN and Seq2Seq which were adapted from [6] as baseline models for measuring improvement with our approach. Both RNN and Seq2Seq have a single layer and a hidden context vector $h \in \mathbb{R}^{256}$. This is lower than $\mathbb{R}^{1024}$ used in [6]. Our reasoning is that we are using 5% of the data used in [1] and therefore keeping the models simpler. Full details of the models are given in 2. Figure 3 shows the validation MAE comparison at 6, 12, 18 and 24 epochs predictions. It can be seen that ST-Transformer outperforms both RNN and Seq2Seq in all four metrics. This finding is inline with experiment results from [1]. ST Transformer's Temporal attention per joint combined with spatial attention improve model capacity and prediction power. On the one hand, for each joint the model computes a temporal attention looking back at previous frames and given a specific frame, spatial attention is computed for each joint on other joints. All of these extra features improve model capacity.

4.3. Teacher-Forcing Effect: Loss

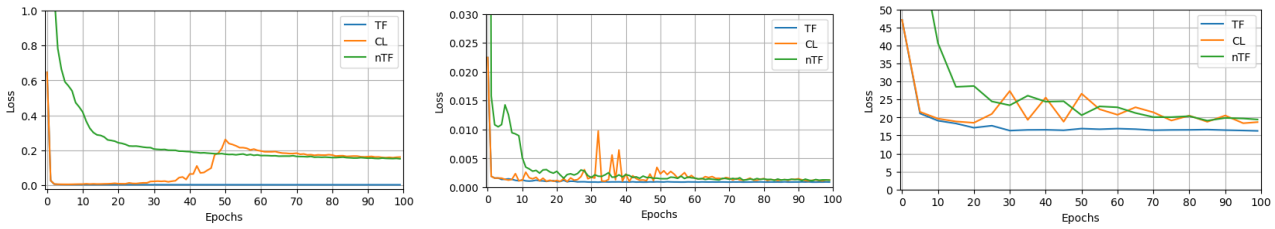
Figure 3 illustrates the effect of teacher-forcing on training and validation losses. When teacher-forcing is used, the input to the model is the original input sequence. On the other hand, when no teacher-forcing is used, the model takes the predicted sequence in the previous iteration as input. We have compared three different teacher-forcing configurations. 1) Teacher-Forcing (TF) applied throughout training. 2) Teacher-Forcing (TF) applied throughout training. 3) Student-Teacher Curriculum-Learning (CL). Teacher-forcing weight starts off at 1 and gradually drops to 0. This weight determines the probability of applying teacher-forcing. The weight decay is scheduled to hit 0 at epoch 50 and remain at 0 until the end of training.

For both of TF and SF configurations, training loss gradually decreases until the end of training. The difference is that TF maintains a lower loss level compared to SF. For CL, the behaviour of the loss curve is between the other two configurations. At earlier epochs, where the teacher-forcing weight is high, the loss is similar to that of TF. Then, the loss diverges from TF loss, and steadily increases as the weight reduces eventually to 0 at epoch 50. Between epoch 50 and 100, the loss curve behaves similar to that of SF configuration. This is expected, as both configurations are applying no teacher-forcing in this epoch range.

Overall, this behavior makes sense. The model suffers lower loss, when it inputs ground truth into the model and takes on higher loss when the model input comes from previous predictions. CL loss acts as a bridge between these two behaviors as the weight of teacher-forcing changes.

Method	GPU Memory	Params	Tr. Loss	Val. Loss	MAE@6	MAE@12	MAE@18	MAE@24
Baseline (ST-TF)	16.1GB	297,127	0.003	0.001	1.480	5.014	10.123	16.281
Joint Mirroring	16.1GB	209,562	0.003	0.001	1.446	4.831	9.686	15.659
Temporal Sampling	6.7GB	297,607	0.003	0.001	1.516	5.109	10.238	16.557

Table 1: All runs done with with same configuration as Model ST-TF in Table 2 using an Nvidia V100.



(a) Training Loss (b) Validation Loss (c) Validation MAE@24

Figure 3: Loss for various teacher-forcing scenarios.

4.4. Teacher-Forcing Effect: Temporal Attention

Teacher-forcing exposes another interesting behavior, this time with temporal attention. For TF, the model's attention does not look too far back into the frames (Fig. 4a). The non-zero weights form a 6-8 frame block before the current frame for all joints. As time progresses, so the block maintains its width as well as its shape and moves the same distance. It is as if temporal attention performs a 1-D convolution (per joint) on previous frames. This behaviour could be intrinsic to TF: the ground truth is fed into the model as input and the model decides that it only needs a few recent frames to predict next new sequence. In SF scenario, attention reaches out more into the past; for some joints it looks back by 70 frames, as shown in Figure 4c. The non-zero weights again for a block; albeit a much wider one. This behavior does not change, as training advances through 100 epochs. This different attention behavior from TF could be due to model input coming from previous predictions. Somehow the model finds the need to reach out to earlier frames, as a few recent frames do not help reduce the loss. Again, CL provides a bridge between TF and SF (Fig. 4b). The model still reaches out to earlier frames and does so more than TF and less than SF. Furthermore, in earlier epochs, where teacher-forcing weight is high, the non-zero attention weights form a narrow block (the model does not look back much). As the epochs advance and the weight drops, the further back frames receive more attention.

Our exploration with curriculum learning as a means to reduce exposure bias suggests teacher forcing does not induce any performance loss and in-fact helps generalization. This is inline with results from Generative language models [7] which suggest mismatches between the model distribution and the data distribution leads to little or no performance gains.

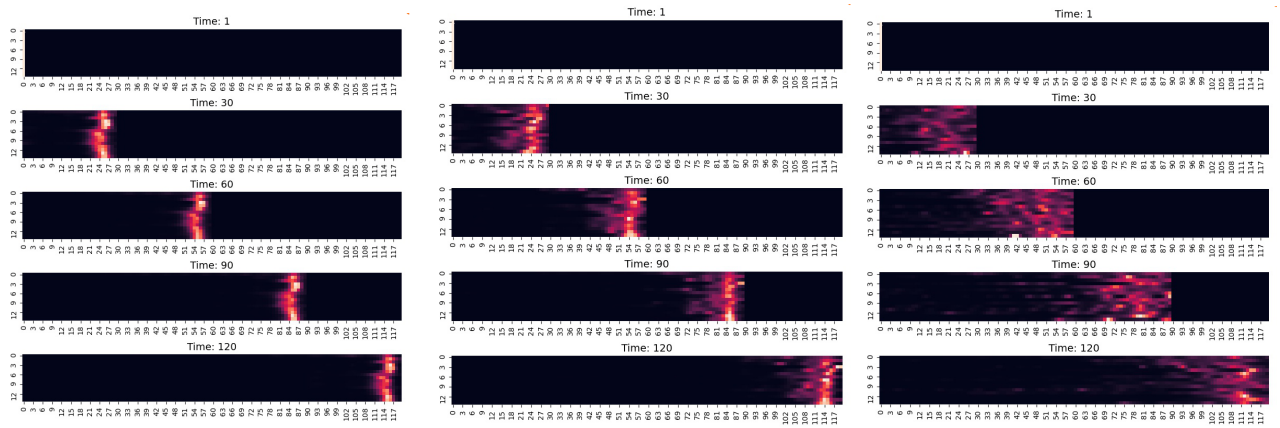
4.5. Embedding Visualization

As the stacked spatio-temporal attention layers are applied to the input embedding, we observed a natural alignment/clustering of motions in the activation space, reference 7 in appendix. Walking and running activations are initially close to one another due to the natural movement dynamics of an individual. Over subsequent attention layers we observe a natural clustering in representation space between the two motions. Intuitively the generative transformer model seems to learn a rich representation of what walking and what running are, that they are 2 different concepts however they overlap (perhaps transition dynamics). Future work could include a more rigorous understanding of this across other motions and further leveraging motion prediction as a surrogate task for downstream tasks such as activity recognition.

5. Experience

5.1. Challenges

We anticipated several challenges: We expected it to take some time to manage and work with the large dataset. We also expected some difficulty in understanding the format of the dataset and what it represents. Dealing with both AMASS and AMASS-DIP was more challenging than we had originally expected. Neither dataset is accompanied with the clear, concise pose data explanation accompanying other datasets such as Human 3.6M [10]. Due to the fact that both of these datasets are too large for our needs, we had hoped to use H3.6M instead, which would have provided the added benefit of allowing for baseline results on a common dataset used in other papers. Unfortunately we hit another unanticipated problem when the Human3.6M Team refused to provide us with the data because we are not affiliated with an official research group. Thus, we instead solved our dataset problem with sampling (E.2).



(a) Teacher-Forcing (b) Curriculum Learning (c) Student-Forcing

Figure 4: Temporal attention for various teacher-forcing scenarios.

Dataset	Model	Val. / Test	B	Epochs	Optimizer	Emb-Dim	Layers	Heads	FFN-Dim	Tr. Loss	Val. Loss	MAE@6	MAE@12	MAE@18	MAE@24
Small	ST	Small	32	90	Adam	64	4	4	128	number	number	number	number	number	number
Small	ST	Small	32	150	Sgd	64	4	2	128	number	number	number	number	number	number
Small	ST	Small	32	150	Sgd	64	4	4	128	number	number	number	number	number	number
Small	ST	Small	32	150	Sgd	64	4	8	128	number	number	number	number	number	number
Small	ST	Small	32	100	Adam	32	4	4	128	number	number	number	number	number	number
Small	ST	Small	32	100	Adam	64	number	number	number	number	number	number	number	number	number
Medium	**ST-TF**	**Medium**	**176**	**100**	**Noamopt**	**32**	**4**	**2**	**64**	**number**	**number**	**number**	**number**	**number**	**number**
Medium	ST	Medium	176	100	Noamopt	32	4	2	64	number	number	number	number	number	number
Medium	ST	Medium	176	100	Noamopt	32	4	2	64	number	number	number	number	number	number
Medium	ST	Medium	176	100	Noamopt	32	4	2	64	number	number	number	number	number	number
Medium	ST	Medium	176	100	Noamopt	32	4	2	64	number	number	number	number	number	number
Medium	ST	Medium	176	100	Noamopt	32	4	2	64	number	number	number	number	number	number
Large	**ST**	**Large**	**176**	**100**	**Noamopt**	**64**	**number**	**number**	**number**	**number**	**0.001**	**number**	**number**	**number**	**number**

Table 2: Run results

Student	Contributed Aspects	Details
All (see Table 5)	Pre-processing, implementation, Analysis	Includes: Multiple meetings per week. Live coding. Live editing. Code reviews. Analysis. Paper reviews. etc.

Table 3: Contributions of team members.

On a lighter note, even with a cloud based DL project, security is important. Account was temporarily suspended because a hacker was able to gain access and use our resources for cryptocurrency mining.

5.2. Changes in Approach

Initial testing indicated the spatio-temporal transformer would require larger hardware and extremely long training times. As such trade-off's in our implementation were made to facilitate training. The most effective change was moving to PyTorch's built-in multi-head attention module, which uses low-level routines that reduced run-time by 72%. However this decision led us to diverge from the SOTA paper as we now refocused the spatial attention to only the latest time step using a query representation that is shared across all joints in a fashion similar to the key and value projections in the original paper. This trade-off allowed us to scale the implementation to size in which we could run different experiments to examine the impact of different design and hyper-parameter decisions.

5.3. Project Success

We had originally planned to measure success through comparison of our results with SOTA paper's results. In this regard, we failed due to computational limitations. However, we succeeded in implementing a motion prediction model that outperforms common baselines 2. We also understood the inner workings of this model well enough to make improvements to the model 1. In this regard, we feel we succeeded.

6. Work Division

Summary of contributions are provided in Table 3. More details can be found in Table 5.

References

[1] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction, 2020. 2, 4, 9

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

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

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

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

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

[7] Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James R. Glass. Quantifying exposure bias for neural language generation. *CoRR*, abs/1905.10617, 2019. 3, 5

[8] 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

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[10] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 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):1325–1339, 2013. 2, 5

[11] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016. 1

[12] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321, 2014. 1

[13] 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

[14] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[15] Raquel Urtasun, David J Fleet, Andreas Geiger, Jovan Popović, Trevor J Darrell, and Neil D Lawrence. Topologically-constrained latent variable models. In *Proceedings of the 25th international conference on Machine learning*, pages 1080–1087, 2008. 1

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017. 1, 2, 4, 8

[17] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007. 1

[18] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. 3

A. Project Code Repository

We used the Facebook Research Fairmotion library as an initial start for training purposes and baseline. We've made the following changes to the code base:

- Implemented Spatio-Temporal (ST) Transformer from scratch
- Implemented an optimized version of ST Transformer along with several other variants
- Implemented Temporal Transformer (which utilizes only temporal attention) from scratch
- Heavily modified training code to suit our understanding of how ST Transformer trains.
- Modified PyTorch's MultiHeadAttention class to expose multi-head attention weights.
- Added pre-processing of AMASS data (fairmotion codebase only handled AMASS DIP data)

The link Facebook Research Fairmotion Github repository is at: https://github.com/facebookresearch/fairmotion
The Github repository for our final project is at: redacted

B. Data breakdown

Motion Set	#Frames	Weight
AMASS_Eyes	3,471,173	38%
CMU	2,237,617	number%
AMASS_BioMotion	2,106,567	number%
AMASS_HDM05	709,598	number%
AMASS_CMU_Kitchen	186,937	number%
AMASS_ACCAD	137,109	number%
JointLimit	88,067	number%
AMASS_Transition	78,592	number%
HEva	38,714	number%
AMASS_MIXAMO	34,406	number%
AMASS_SSM	8,852	number%

Table 4: AMASS DIP Dataset breakdown by motion set.

C. Masking

As the approach is generative. Masking is required to prevent the attention layers from attending to timesteps in the future. This would pose a problem during inference when future data is not available. An image of the attention weights absent masking shown in 5

C.1. Motion Spatial Attention

As shown in figure 6. Each motion elicits a different attention map in our trained model.

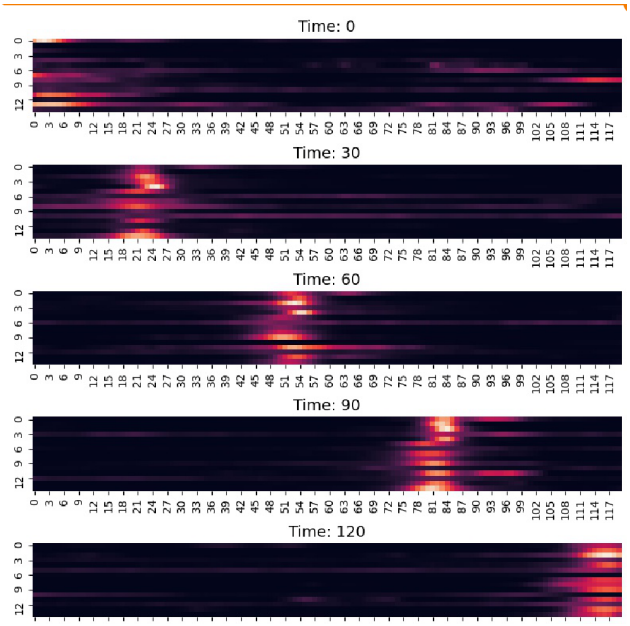


Figure 5: Attention weights with no masking applied. This approach would not work during inference as the network is cheating by looking into the future.

Figure 6: Spatial attention maps for different motions

D. Optimizer

The noamopt optimizer was used for training. This is essentially the ADAM optimizer wrapped in a custom scheduler which accounts for warmup as well as learning rate decay in a closed form equation on each timestep. This is specifically used for training transformers as they are sensitive to learning rate [16]. As such it was used in our experiments.

$$\text{learning rate} = D^{-0.5}\dot{\min}(\text{step}^{-0.5}, \text{step} \times \text{warmup}^{-1.5})$$

E. Training Scheme

More detail on the training schemes are shown as follows, first some notation

$t \in T$. timestep t in full sequence length T

Given src and tgt sequences,

T_{src} = 120, T_{tgt} = 24

X = concat(src, tgt) $T_X = 144$

E.1. Training Scheme 1

We want to output prediction $P \in \mathbb{R}^{143}$. This is obtained by taking the full sequence and feeding it into the model one at a time, taking the predictions from each autoregressive step and filling our predictions vector. As such we can measure the loss for the sequence per [1]. Moreover for simplicity lets assume teacher forcing = 1, ie. we feed ground truth on each subsequent prediction p.

model$(x_1) \rightarrow \bar{x}_2 \rightarrow p_1$

model$([x_1, x_2]) \rightarrow [\bar{x}_2, \bar{x}_3] \rightarrow p_2$

model$([x_1, x_2, x_3]) \rightarrow [\bar{x}_2, \bar{x}_3, \bar{x}_4] \rightarrow p_3$

...

model$([x_1, x_2, \ldots, x_{143}]) \rightarrow [\bar{x}_2, \ldots, \bar{x}_{144}] \rightarrow p_{143}$

Observe that $p_t \in \mathbb{R}^1$ here. We just take the last element of the output vector which denotes the next prediction. We can then measure loss between P and $[x_2, x_3, \ldots, x_{144}]$ however the computational graph grows linearly with T and goes OOM quite fast. This can be mitigated by performing backpropagation on each prediction $\mathcal{L}(p_t, x_{t+1})$ however then this loss diverges from [1].

E.2. Training Scheme 2

We want is an output prediction $P \in \mathbb{R}^{24}$ which we compare to tgt. we feed the source sequence into the model in a sliding manner with window size 120 and predict the 121st. Like above, assume teacher forcing = 1 for simplicity.

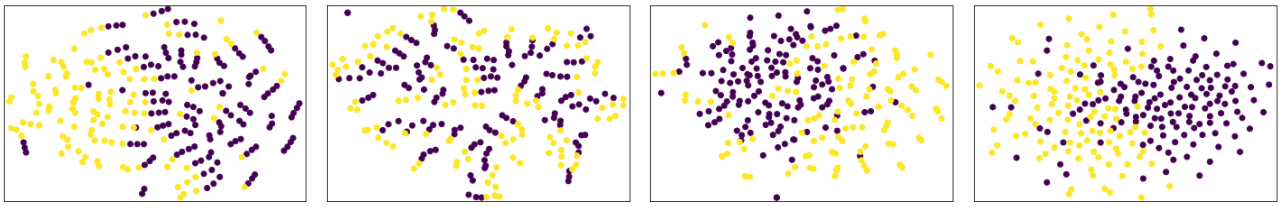
model$(x_0 \ldots x_{119}) \rightarrow [\bar{x}_1, \ldots, \bar{x}_{120}] \rightarrow p_1$

model$(x_1 \ldots x_{120}) \rightarrow [\bar{x}_2, \ldots, \bar{x}_{121}] \rightarrow p_2$

...

model$(x_{24} \ldots x_{143}) \rightarrow [\bar{x}_{25}, \ldots, \bar{x}_{144}] \rightarrow p_{24}$

With this approach we can measure loss between predictions P and tgt and each element of P, $p_t \in \mathbb{R}^1$. This presents the same computational graph growth and OOM issues as E.1. An alternative approach is to measure loss between the full predicted output sequence and shifted input sequence and backpropagate on each prediction. This keeps a similar interpretation as the loss presented by Aksan. We're always predicting the input shifted over 1 and can measure loss on each sequence. As a result $p_t \in \mathbb{R}^{120}$, we are always predicting the shifted sequence and measure loss as follows, $\mathcal{L}(p_t, [x_{t-1}, \ldots, x_{t+120}])$. This Training 2 alternative scheme was applied throughout our experiments as it mitigated OOM issues and also measured loss in a similar fashion.



(a) layer 1 activation (b) layer 2 activation (c) layer 3 activation (d) layer 4 activation

Figure 7: t-SNE for walking (yellow) and running (purple) activations over increasing spatio-temporal attention blocks.

Student	Contributed Aspects	Details
Redacted	What I did	my details
Redacted	What he did	his details
Redacted	what she did	her details
Redacted	what they did	their details

Table 5: Contributions of team members.