CS 7643 Project Report: Human Motion Prediction

Christopher Forgach
chris.forgach@gatech.edu

Wei En Chen
wchen658@gatech.edu

Austin Patchin
apatchin3@gatech.edu

Abstract

We explore modern Transformer and Generative Adversarial Network architectures for 3D human motion prediction on the Deep Inertial Poser data set. Spatio-Temporal Transformers have shown success over longer time horizons than previous approaches through the use of two independent self-attention blocks, and yield our closest results to the state of the art due to the relative simplicity and different levels with which we leverage respective libraries. Although we do not beat the state of the art, we examine several paths in depth upon which future work could be built. Our code can be found at https://github.gatech.edu/sp22-cs7643-keep-it-on-the-dl.

1. Introduction

We performed three groups of experiments to implement, improve upon, and understand state-of-the-art (SOTA) models for predicting human body motion. Given a series of video frames, can we predict what a person will do next? Such predictions are useful anywhere machines navigate the same spaces as people, such as for cars to avoid pedestrians, for surveillance to follow occluded targets, for assisting motion capture to produce movies, or for virtual reality game development. It can also serve as genre-specific foundational knowledge for related applications such as automatic video captioning, or video generation where the main subjects are people.

The recent work done in [1] with Spatio-Temporal Transformers (STTs) represents the latest advancement in generative human modeling. STTs utilize a dual self-attention mechanism that learns temporal and spatial high-dimensional embeddings independently, allowing the model to use both current and past data simultaneously. It achieves mean Euler distances of 87% on our data set, which is the baseline against which we compare our other experiments. Nonetheless, there is still room for improvement such as with its auto-regressive nature that compounds errors quickly as predictions stretch farther out in to the future. Generative Adversarial Networks (GAN) also address this issue by abstracting the loss functions further still, but there is less existing research applying them to produce sequential outputs.

Format	Points per Joint	Size (GB)
Axis angle	3	5.2
Quaternion	4	6.9
Rotation matrix	9	15.5

Table 1. The DIP skeletons can be preprocessed to three mathematically equivalent but computationally distinct formats. Axis angle representation is the smallest and fastest to use because the most samples fit within memory.

Human motion prediction is of commercial interest to Meta Platforms in particular for its recent development of a virtual reality online multiplayer game, in order to reduce the perception of lag and to extrapolate fake body parts for which there are no sensors. The SOTA performance is suitable for artistic purposes, but it is still grossly inadequate for safety-critical applications like robotics. If we are successful in expanding the SOTA, it could further improve industry's ability to rely on cheap-to-develop machine learning models for such risk-averse fields. Note that since our work is a subset of sequence-to-sequence [12] modeling, it could also be adapted to other sequence-based data sets like video, text, audio, or some combination thereof. However, our realistic aim is to build a foundation for future researchers, given our time and compute constraints.

We use the Deep Inertial Poser (DIP) [8] data set, specifically the DIP synthetic data set generated from archival motion capture data and not the DIP-IMU set intended for fine tuning. It was developed for academic-only use by researchers from the Advanced Interactive Technologies Lab (ETH Zurich), and the Max Planck Institutes for Intelligent Systems (Tubingen) and Informatics (Saarbrucke). Note that DIP and AMASS are continually evolving, and our version was retrieved April 2022. Figure 1 shows one frame.

DIP is composed as a subset of the Archive of Motion Capture As Surface Shapes (AMASS) [9]. It contains 8720 sequence files of varying lengths. The filenames are not of uniform format because the data set is a compilation, but most contain short descriptions of the sequence in either English or German. It is available in both skeleton and

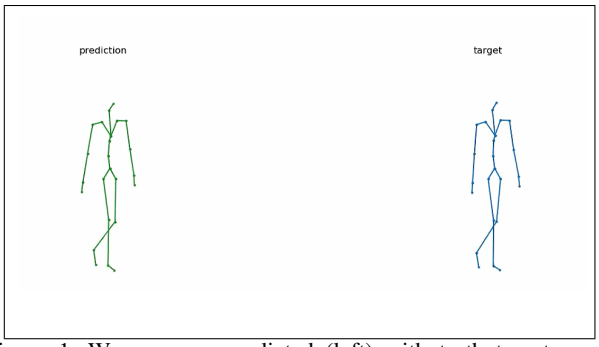

Figure 1. We compare predicted (left) with truth target poses (right). The data set has labels for up to 24 frames, so we can asses longer predictions only qualitatively. The body is made from 24 joints, including the terminal joints of head, hands, and feet which do not contribute to poses in practice.

2D video formats, but we use only skeleton form to minimize computational and compute complexity and to remove male-female distinctions. The body in each frame is made of 24 joints and the lines between them. Note this includes end nubs such as the hands, feet, and head whose data are unused in practice except for visualizations, because they lack the third points with which to form angles.

The skeleton format itself is preprocessed in to three representations as outlined in Table 1, where each sample is a time slice of 120 input frames with the immediate next 24 frames as labels. We use the Fairmotion splits [7] of 60678, 3007, and 3304 samples in the training, validation, and testing sets, which allows us to compare against other baselines and is appropriate because the validation and testing splits are of equal size with the training split 20 times larger. Our experiments use the axis angle format unless specified otherwise because it is the smallest and fastest format computationally, since it allows the most samples to fit in memory at once.

2. Approach

2.1. Motion Taxonomy

Motion can be viewed as a hierarchical structure, as seen in Figure 1 from [3]. Different types of mechanical motion can be clustered together (i.e. rotating, shaking, or a static lack of motion) to provide embedded information for the overall trajectory of motion. We wanted to see the effect that motion taxonomy, classifying the type of motion, has on a Transformer architecture for motion prediction when applied on the input. The architecture for the input prior is shown in Figure 2. The input is fed into a linear layer of size 18, to represent the 18 motion codes detailed in [3], before passing into a sigmoid function. Then, the bit-wise result is fed into a linear layer of the original input size and multiplied by the original input to focus the input on the important frames of motion. The augmented input is then fed into the existing Transformer Fairmotion [7] framework.

2.2. Generative Adversarial Network (GAN)

We also implement a GAN-based model as described in [11] and outlined in Algorithm 1 using the Fairmotion [7] and PyTorch frameworks to try to improve the quality of predictions if not the accuracy. The version from [11] seeks to overcome limitations of the original GAN propsal from [6] including *mode collapse*, *vanishing gradients*, and *non-convergence*. Note that our data preprocessing makes no attempt to group whole actions in to each sample, but the 24-frame outputs are unlikely to contain multiple actions because the inputs are downsampled from 60 to 10 frames per second.

The inputs and outputs are both sequences, so we use transformer-based models with hidden Long-Term-Short-Term (LSTM) units of size 512 for both the generator and the critic. The generator takes 120 frames and outputs the next 24 frames. The critic takes 24 frames and outputs a single frame of probabilities that each joint is real throughout the sequence. Future work may study partial inputs where some joints are fake and some real within the same sequence, but we did not study it in these experiments. We train the generator to maximize the critic's output probabilities, and the critic to distinguish between real and generated sequences.

Algorithm 1 Wasserstein GAN Implementation. C = critic, G = generator, c = clipping range

function WASSERSTEIN LEARN ON BATCH
  Load size m minibatch of inputs X and truths y
  for $t \leftarrow 1..n_{\text{critic}}$ **do**
    $T \leftarrow \text{concat}(X, y)$
    $U \leftarrow \text{concat}(X, G(X))$
    $\Delta C_\theta \leftarrow \Delta\text{EMD}(C(T), C(U))$
    Clip ΔC_Θ within $\pm c$
    $C_\theta \leftarrow C_\theta + \Delta C_\theta$
  $V \leftarrow \text{concat}(X, G(X))$
  $\Delta G_\theta \leftarrow \frac{1}{m} \sum_{i=1}^{m} (T - V)^2$
  $G_\theta \leftarrow G_\theta - \Delta G_\theta$
  return G_θ

The experiments in [11] stabilize GAN training by replacing the critic's brittle Kullback-Leibler (KL) or Jensen-Shannon losses with the Earth Mover Distance (EMD) as described in [6] and shown in Equation 1. In short, EMD is the mean squared error between the cumulative density functions of two distributions. We maximize the critic's output EMD for real versus generated inputs, rather than naively minimizing its error between output versus true probabilities. We implement the Wasserstein algorithm faithfully except for the underlying transformers, which we

do not expect to make much difference versus RNNs/LSTMs except for sample complexity. We also use new data. We find the generator needs additional stabilization over those used in [11]. Its outputs are seeded with truth probabilities during training to tie high values to real and low values to falsified, but with uniform random noise during validation to maintain the integrity of the results.

$$\text{EMD} = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{i=1}^{k} C(T_k)) - \sum_{i=1}^{k} C\big(U_k\big)^2\right) \quad (1)$$

There is another approach by [2] which adds two extra terms to the generator loss L_g as shown in Equation 2.

$$\begin{aligned} L_g &= L_{adv} + \alpha L_{pg} + \beta L_b \\ L_{adv} &= -D(x||G(x,z)) \\ L_{pg} &= \left[\sum_t |y_t - y_{t-1}|^p\right]^{1/p} \\ L_b &= \sum_t \left[\sum_i |b_t^i - b_{gt}^i|^2\right]^{1/2} \end{aligned} \quad (2)$$

L_{adv} is the generator loss we implement, L_{pg} is the pose gradient to smooth out the motions, and L_b is the change in bone lengths which is not applicable to our fixed-bone data. Considering Figure 4 yields the worst training and validation results of our three experiments, this may be a better approach since the Wassersetin GAN is too abstract to train well reliably.

2.3. Spatio-Temporal Transformers (STTs)

In this section, we dig deeper into the architecture of spatio-temporal transformers [1] by Aksan et al, who use the DIP data set and serve as our direct baseline. We experiment with extending our predictions from 24 to 1000 frames, look briefly at embeddings, and then perform 14 studies of various ablations and augmentations in an attempt to understand the performance. We implement these experiments in Tensorflow to build from the code base provided by [1]. Each study is enumerated in Figure 6, and their details are discussed alongside the results.

3. Experiments and Results

3.1. Motion Taxonomy

For this experiment, although we cannot use the pretrained motion taxonomy embeddings from [3], we are able to emulate the motion codes for the 18 categories of motion, as seen in Figure 2. We implement this addition to the existing Transformer model in the Fairmotion [7] framework using Pytorch 1.11.

The resulting training curve is seen in Figure 3. The training curves are visibly identical, both increasing as the validation curve undergoes noise and decreasing steadily afterwards. The motion taxonomy validation curve is boosted with a lower starting MSE value and experiences a lower maximum amplitude of noise. However, the motion taxonomy validation curve contains a higher overall area of noise, which is probably caused by the added complexity of the motion taxonomy model. The noise visible between 25 and 45 epochs is most likely an artifact of information loss that existed when training the original Transformer model.

The multiplication of the motion codes filters the input tensor on the areas of motion. The goal it to reduce the overall information loss by focusing on the pertinent data for a particular type of motion. Augmenting the input data through motion taxonomy as a prior does allow a better validation curve before 25 epochs; however, after the added information the motion taxonomy transformer performs at most equal to the original Fairmotion Transformer. Future

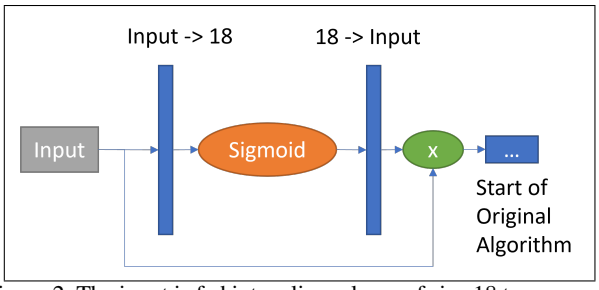

Figure 2. The input is fed into a linear layer of size 18 to represent the 18 motion codes. The sigmoid function is applied to emulate the bit-wise hierarchical structure of motion from [3]. A linear layer is applied to return the new bit-wise structure to the input size. Finally, before proceeding with the STT algorithm, the input is multiplied by the motion codes to realize the added motion taxonomy.

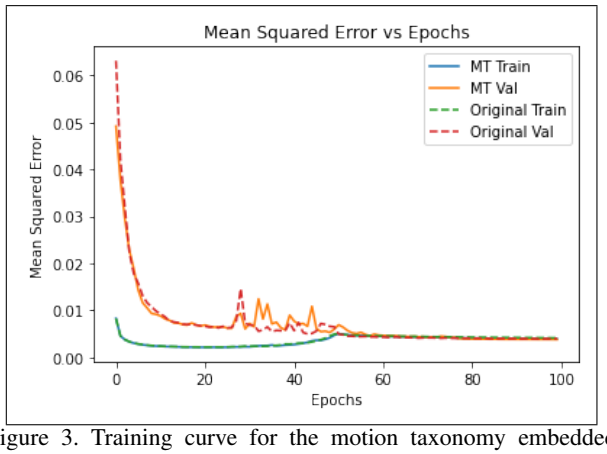

Figure 3. Training curve for the motion taxonomy embedded Transformer. MT Train/Val represent the motion taxonomy Transformer, and Original Train/Val represent the original Fairmotion Transformer.

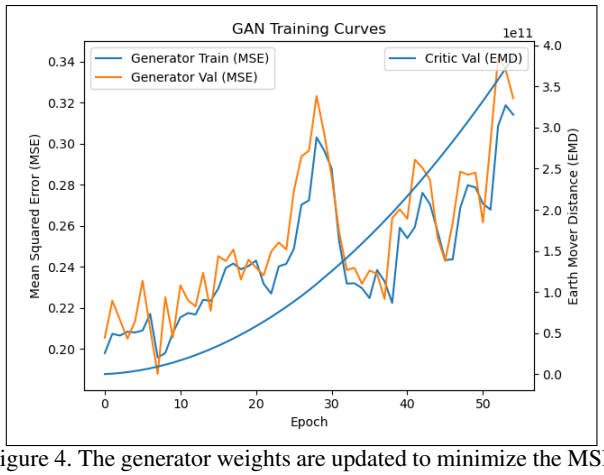

Figure 4. The generator weights are updated to minimize the MSE (jagged lines) between the critic's output and a prediction that the generated sequence is 100% real, but that is a moving target so the values shown are the MSE between the generated and true sequences. The critic weights are updated to maximize the EMD (smooth line) between predictions on real and generated sequences.

work includes focusing on reducing the information loss evident in the original transformer architecture.

3.2. Generative Adversarial Network (GAN)

We implement the GAN in a Python 3.7 environment by adding a new task and model to the Fairmotion [7] framework. Figure 4 shows the training curves. The smoothness of the critic's EMD curve suggests that the Wasserstein loss function and the weight clipping are effective. The generator performs with 21% error early on, but degrades throughout training, likely due to poor tuning. We used the same hyperparameters as in [11], which do not translate well to the new data set. It may be helpful to add gradient clipping to the generator to smooth out its progress similar to the critic, in addition to the other changes in the HP-GAN loss. Future work should consider a generator loss function composed of MSE from the training labels to tie its training to reality more tightly, the critic output as a mere quality penalty, and the position gradients as a smoothness penalty.

Overall its best MSE of 15% is worse than those of our other experiments. The sudden improvement between epoch 26-35 followed by continued degradation suggests an irregular gradient space, which makes sense given the the loose connection between the generator and reality through the critic. Perhaps a critic is most necessary when the output is long compared to the input, unlike our data which has a five to one input/output ratio. Note this range aligns with the sudden irregularities in Figure 3, suggesting it is a precipice in the gradient space is large enough to appear even in moderately different models.

3.3. Spatio-Temporal Transformers (STTs)

3.3.1 STT Long Term Prediction

In the first STT experiment We increase the prediction duration from 0.4s (24 frames) to 16.7s (1000 frames) and use full resolution data of 60 frames per second to see what happens at this time scale. Several motions are selected such as walk, run, jump, etc. The videos are uploaded to here, with target (left right) and prediction (left) motions. Figure 1 shows a snapshot of the videos. The first 2 seconds (120 frames) are input to the model and then rest of the time is prediction made by the trained model. For example, we generate a walking video where the first half appears smooth and reasonable qualitatively, even though the pace is not in sync with the target. However, the latter half of the predictions either freeze or become erratic as the gradients vanish or diverge, and the predictions extend farther out pas the length of our training labels.

We believe this is due to the autoregressive nature of the model. The ST transformer takes [0:n] input window and predicts [1:n+1] output window. The model then takes the output as the input for the next iteration until the final output window is [1000:n+1000] in our case. Since the output depends on its previous states, small noises can accumulate and eventually lead to extremes. Walking, running, and jogging are more repetitive motions which the model can capture. It is for the other transient motions such as catching/throwing and jumping that the predictions stop moving after approximately 120 output frames. It is is possible that for such compound motions the true output has little to do with the input, and the probability of everything the model predicts and thus its motion drops to zero.

3.3.2 STT Embedding

When transformers are used for natural language tasks, the embedding is a learned representation for text where words with the same meaning have a similar representation. In the transformer model, an embedding-like layer is added to the architecture. We want to see if the trained model will have any meaningful structure to the embedding. The embedding weights are extracted and their shapes take the shape [number of joints, joint size, embedding size]. To better visualize the weights as a 2D image, the first two axes are combined. In Figure 5, the upper left image is the entire embedding matrix shown in grayscale. White represents maximum weight values and black represents minimum values. To further investigate the weights, they are divided into left, right, and center joints. Left and right include elbows, knees, hips, etc., while center includes spines, neck, etc. A closer look at the separated images shows some subtle patterns. The pixel brightness is less contrastive at the lower left corner of the left and right visualizations, such

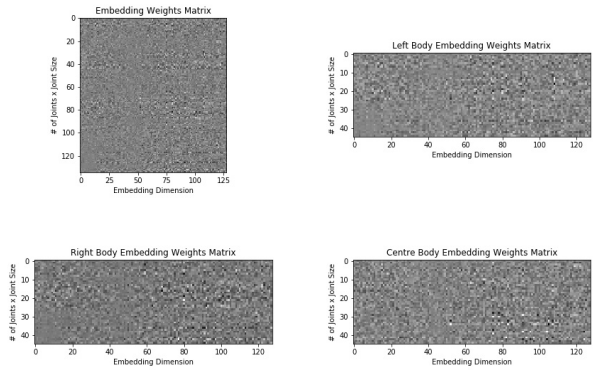

Figure 5. The visualization of joint embedding. In reading order: entire embedding, left body joint, right body joint, center body joint.

that there are more gray pixels and fewer black and white pixels. In addition, all of these visualizations have a vertical band of gray pixels near columns 40-50. Since the left and right joints are ordered the same way in the matrix (hip, knee, collar, shoulder, elbow), it is a qualitative indication that the model indeed learned some representation.

3.3.3 Ablation and Augmentation Studies

Figure 6 shows the results of all 14 STT ablation/augmentation studies. We discuss several with the most interesting results here, and attach the remaining discussions in Appendix A. We limit each trial to 12 hours to match [10].

Study 10: No Temporal Attention

The heart of the ST Transformer model[10] is the spatial-temporal attentions. We would like to know what happens if remove the temporal portion. In this study, we replace the temporal attention block with another spatial attention block, such that there are two spatial attention blocks in parallel. The remaining architecture stays the same as the original STT [10]. As suspected, study 10 in Figure 6 shows quite poor results. Attention which focuses only on inter-joint relations and not across time will cause severe degradation to the performance. Another reason is the number of learnable parameters decreased, since the temporal attention is applied on every joint so there are more parameters accumulated. In other words a temporal attention block has more learnable parameters than a spatial attention block in the ST Transformer [10].

Study 12: No Positional Encoding

Positional encoding is introduced in [4] to inject information about the relative or absolute positions of the tokens in the sequence. In this case, the tokens are the poses. We remove the positional encoding from the benchmark to see what happens to the performance. As seen in Figure 6, study 12 shows slightly worse results than the benchmark. An interesting observation here is study 12 results are better than study 11 (temporal only architecture) after 100ms, but worse after 400ms. This indicates the positional encoding provides some knowledge to the model to learn longer term prediction.

Study 13: Seven Attention Layers

We notice an interesting pattern when plotting the gradients of the weights $\delta Loss/\delta W$ against their means and standard deviations for individual layers from studies 01-12. Generally, the last few layers in the architecture have higher variance compared to the rest. This is an indication that later layers tend to learn more from the loss than other layers. Some of the plots are shown in Figure 7. Most of the plots show wider standard deviations in the last few layers, or a trend that the variances are monotonically non-decreasing between each successive layer. This behavior is also observed in [5]. There are some exceptions in our studies, which are "No Spatial Attention" and "No Positional Encoding". For "No Spatial Attention", the earlier layer gradients have wider standard deviations such that the behaviour is reversed. This could be due to the fact there are more learnable parameters in this model and with the given time the training could continue until the behaviour is reversed. For "No Positional Encoding", the same reversed behaviour is observed but is more severe. The first layer gradients have wide deviations while others are narrow. This means the positional encoding information can be passed along throughout the layers for the model to learn the motion sequence.

Given these observations, we would like to understand how decreasing the number of layers from eight to seven affects the performance and gradients. Study 13 in Figure 6 shows the joint angle difference is similar to the benchmark, meaning one layer fewer produces negligible loss. However, in weight gradients (7 Attn Layer at bottom right hand corner) in Fig 7, the gradients in the first layer have wider standard deviation compared to the rest. From the second to the last layer, the trend seems to be increasing in standard deviation. The implication here is that the weight gradients may not be a clear indicator of how well the model performs, but the gradients are directly tied to how weights are updated in each step. If the gradient distribution can change dramatically from eight layers to seven, this could mean the training process can become unstable. We see in Figure 8 that the joint loss becomes noisy early in training. It is possible that stabilizing the gradients throughout the layers can increase the learning speed in the training process which leads to shorter learning time and/or increased accuracy.

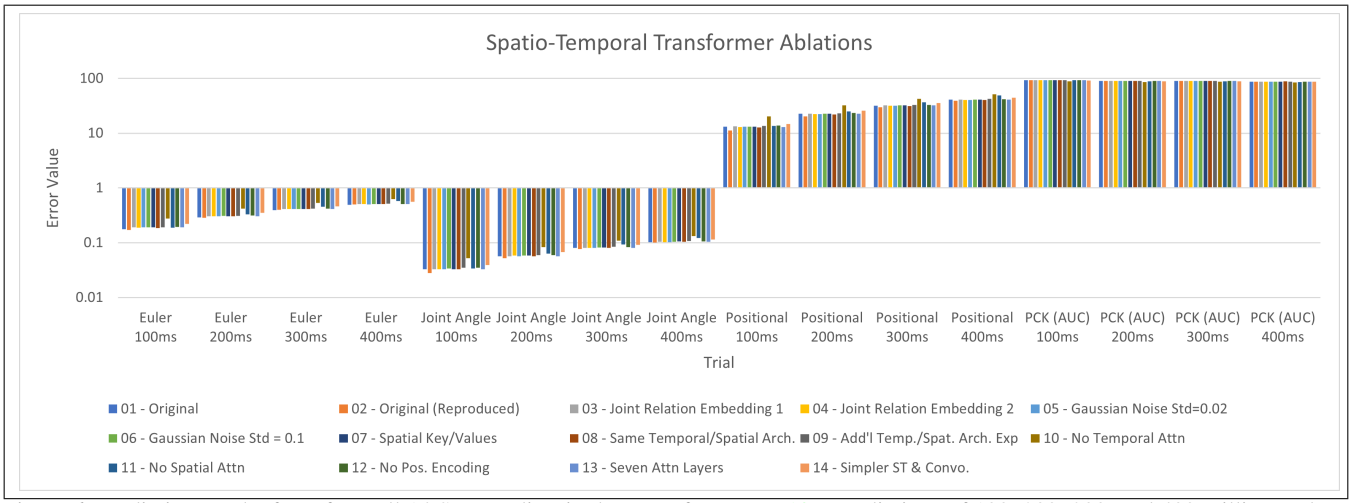

Figure 6. Prediction results from from all 14 STT studies, in the same format as [10]. Predictions of 100, 200, 300, and 400 milliseconds are included. For Euler, joint angle, and positional, lower is better. For area under the curve (AUC), higher is better.

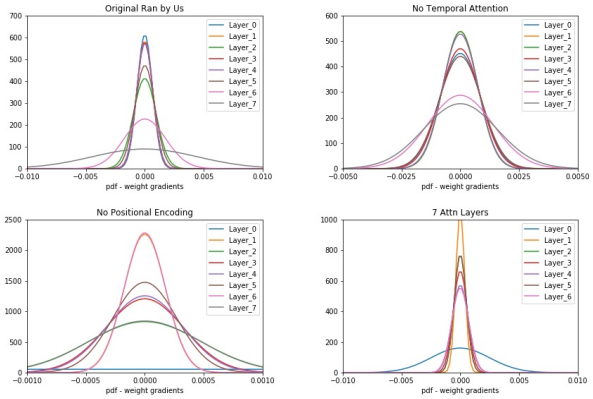

Figure 7. The $\delta Loss/\delta W$ distribution among layers. Most of them display higher standard deviation in the later layers.

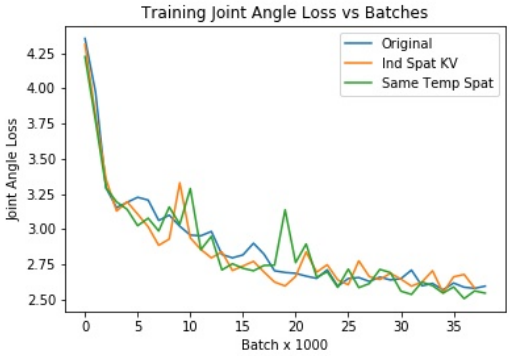

Figure 8. Typical training process in ST Transformer model. The loss is quite noisy early on in the training process.

4. Conclusion

We surveyed motion taxonomy, implemented a GAN model, and and studied STT architectures as they relate to predicting human body motion. We found that although well-implemented GANs can achieve better performance than STTs in theory, they are still more brittle and perform far worse with only slight implementation and tuning errors. In contrast, numerous parts of an STT such as the spatial attention mechanism can be ablated with only minimal or moderate degradation in performance. This effort used data with pre-cleaned and pre-extracted features, but future work may focus on raw video to generalize to a broader range of use cases. A precursor to that would be to research intelligent embedding schemes to make high-dimensional inputs and outputs tractable.

5. Work Division

We took a federated approach, where each person conducted fairly autonomous experiments. Wayne Chen did everything related to the Spatio-Temporal Transformer sections, including the appendix and Figure 1. Chris Forgach did everything related to the GAN sections, wrote much of the common sections and some broad editing, and minor administrative tasks like making the GitHub organization and preprocessing the DIP data set. Austin Patchin did everything related to the Motion Taxonomy sections, as well as some broad editing and some writing of common sections. See Table 2 for a concise summary of the work breakdown.

References

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

[2] Emad Barsoum, John Kender, and Zicheng Liu. Hp-gan: Probabilistic 3d human motion prediction via gan, 2017. 3

[3] Yu Sun David Paulius, Nicholas Eales. A motion taxonomy for manipulation embedding. 2020. 2, 3

[4] Ashish Vaswani et al. Attention is all you need. 2017. 5

[5] Ruibin Xiong et al. On layer normalization in the transformer architecture. 2020. 5, 10

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[7] Deepak Gopinath and Jungdam Won. fairmotion - tools to load, process and visualize motion capture data. Github, 2020. 2, 3, 4

[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 sparse inertial measurements in real time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 37(6):185:1–185:15, Nov 2018. 1

[9] 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. 1

[10] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks, 2017. 5, 6, 9, 10, 11

[11] Ayumi Shiobara and Makoto Murakami. *Human Motion Generation Using Wasserstein GAN*, page 278–282. Association for Computing Machinery, New York, NY, USA, 2021. 2, 3, 4

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. 1

[13] Austin Wang. Improved transformer architecture for sequence to sequence translation. 2019. 11

Student Name	Contributed Aspects	Details
Chris Forgach	GAN analysis and implementation	Processed data set. Implemented GAN and its analysis. General writeup and editorial work.
Wei En Chen	STT analysis and implementation	STT work including the 14 tweak/ablation studies and their writeups
Austin Patchin	Documentation, Motion Taxonomy	Researched STTs, pre-trained embeddings, and motion taxonomy. Implemented the motion taxonomy embedded Transformer.

Table 2. Contributions of team members.

A. Appendix: Spatio-Temporal Ablations

Study #	Study Names
01	Original - Ran by Authors
02	Original - Ran by Us
03	Joint Relation Embedding 1
04	Joint Relation Embedding 2
05	Gaussian Noise Std = 0.02
06	Gaussian Noise Std = 0.1
07	Individual Spatial Keys & Values
08	Same Temporal & Spatial Architecture
09	Additional Temp. & Spat. Architecture Exp
10	No Temporal Attn
11	No Spatial Attn
12	No Positional Encoding
13	7 Attn Layers
14	simpler ST and Convo

Table 3. Names of the tweaks and ablation done on the ST Transformer model.

Study 01-02: Original

The saved model by authors are provided in the paper [10]. However, to avoid potential difference due to different package versions, the model is ran again by us and is reported here as the second study. As the authors mentioned in the paper, due to the random initialization of the weights results can vary slightly, and we do see that in the results ran by us. The resulting joint relation matrix takes the form shown below.

```
pos = [[1  0  1  1  0  0  0  0  0  0  0  0  0  0  0]
       [0  1  1  0  1  0  0  0  0  0  0  0  0  0  0]
       [1  1  1  0  0  1  0  0  0  0  0  0  0  0  0]
       [1  0  0  1  0  0  0  0  0  0  0  0  0  0  0]
       [0  1  0  0  1  0  0  0  0  0  0  0  0  0  0]
       [0  0  1  0  0  1  1  0  0  0  0  0  0  0  0]
       [0  0  0  0  0  1  1  1  0  0  0  0  0  0  0]
       [0  0  0  0  0  0  1  1  1  1  1  0  0  0  0]
       [0  0  0  0  0  0  0  1  1  0  0  1  0  0  0]
       [0  0  0  0  0  0  0  1  0  1  0  0  1  0  0]
       [0  0  0  0  0  0  0  1  0  0  1  0  0  0  0]
       [0  0  0  0  0  0  0  0  1  0  0  1  0  1  0]
       [0  0  0  0  0  0  0  0  0  1  0  0  1  0  1]
       [0  0  0  0  0  0  0  0  0  0  0  0  1  0  1]
       [0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1]]
```

Study 03-04: Joint Relation Embedding

In the previous section, we discussed that the embedding learned some representation of the joints by qualitative analysis. We want to see if we project a predetermined joint relation into the embedding, similar to how positional encoding is superimposed, perhaps the model can learn from it. There are 15 joints in each pose (frame). We assume any joint and its adjacent joints will have relationships of 1 while others maintain zero.

Study 05-06: Gaussian Noise

We augment each training training input element with Gaussian noise, to mimic signal noise captured during data acquisition. The max and min values in the data should be 1 and -1, respectively since it is in rotational matrix format. A reasonable, small Gaussian noise will be $N(0, 0.02^2)$. As a comparison, we also include a larger Gaussian noise of $N(0, 0.1^2)$. The code is very straight forward since we are only adding `tf.random.normal()` with specified shape, μ, and σ. And we are adding it to the input before the embedding layer. From the studies 5 and 6 in Figure 6, we can see that $\sigma = 0.02$ has no negative impact on joint angles and positions while $\sigma = 0.1$ starts to show loss in performance. Euler angle shows the difference is higher and we speculate that it's because the error is compared on 3 angles per joint while joint angle is compared by extracting the only one angle component from angle-axis representation. Since there is also no gain in performance, it's likely that the evaluation set is very clean that this type of data augmentation provides no benefit.

Study 07: Individual Spatial Keys & Values

In this study we tweak the spatial attention block of the ST transformer. In the original architecture [10], the keys and queries are shared among joints. Below are the original formulas: W is the weights, t is specific frame in sequence, i is i-th joint. N is total number of joints. E is embedding composed of $e^{(1)}$ to $e^{(N)}$.

$$
\begin{aligned}
Q_t^{(i)} &= [(W^{(1,Q,i)})^T e_t^{(1)}, ..., (W^{(N,Q,i)})^T e_t^{(N)}]^T \\
K_t^{(i)} &= E_t W^{(K,i)}, V_t^{(i)} = E_t W^{(V,i)}
\end{aligned} \tag{3}
$$

Keys and values are modified so they take the same form as queries.

$$
\begin{aligned}
K_t^{(i)} &= [(W^{(1,K,i)})^T e_t^{(1)}, ..., (W^{(N,K,i)})^T e_t^{(N)}]^T \\
V_t^{(i)} &= [(W^{(1,V,i)})^T e_t^{(1)}, ..., (W^{(N,V,i)})^T e_t^{(N)}]^T
\end{aligned} \tag{4}
$$

This will result in more learnable parameters but the training speed did not increase dramatically since most of the calculations are parallel. The result of such study (Study 07) in Figure 6 shows the the joint angle loss is still quite close to the benchmark, which indicates the individual keys and values don't bring values to the performance.

Study 08: Same Temporal & Spatial Architecture

In the original ST Transformer [10], the temporal attention block creates Q,K,V for individual joint, and attention mechanism is done for every joint in series. This implementation does significantly increase training time. We want to see how the model performs if we simplify it to take similar form as the spatial attention block. The original temporal QKV from [10]:

$$\begin{aligned} Q^{(n,i)} &= E^{(n)} W^{(n,Q,i)} \\ K^{(n,i)} &= E^{(n)} W^{(n,K,i)} \\ V^{(n,i)} &= E^{(n)} W^{(n,V,i)} \end{aligned} \tag{5}$$

The modified temporal QKV will have the same form as the previous study above in Equation 4. The result (Study 08) shown in Figure 6 perform similarly to the benchmark in joint-angle. This is an indication that the architecture from previous study is capable of learning in the temporal attention block. It's likely the per-joint attention is not bringing too much value while increasing training time.

Study 09: Additional Temp. & Spat. Architecture Exp

Since it is shown we likely don't need per-joint attention mechanism in temporal space in the previous study, we will keep the temporal attention architecture from the study, and revert the spatial attention back to keys and values are shared among joints. More concretely, spatial attention is back to Equation 3. The thought behind this is when we modified the spatial attention block from sharing keys and values to individual ones (Study 07), the performance does not increase. Now when we finish Study 09, we see that in Figure 6 the performance is worse than the benchmark. We speculate this is due to interactions between spatial and temporal attention blocks as they are superimposed at the end of each attention layer.

Study 11: No Spatial Attention

From what have learned in the previous study, it is likely the temporal attention is more critical than the spatial attention due to the number parameters, and also the prediction is on a sequence of 120 frames while there are only 15 joints in each pose. In this study, we replace the spatial attention block with the temporal attention block, as illustrated in Figure 9. Since there are more learnable parameters in the temporal attention block, the modified architecture ends up with more parameters than the benchmark. As shown in Figure 6, study 11 has a better performance than pure spatial attention, confirming our hypothesis. While the spatial attention isn't as important as the temporal attention, without it still degrades the performance slightly compared to the benchmark. It is also noted that the longer term (400ms)

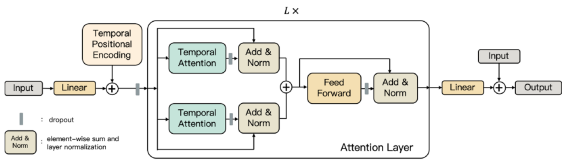

Figure 9. An illustration of the modified ST Transformer with two spatial attention blocks in parallel.

suffers more loss compared to the benchmark, meaning spatial attention can help temporal attention to learn longer term prediction.

Study 13: Seven Attention Layers

When we plot the gradients of the weights, $\delta Loss/\delta W$, in terms of their means and standard deviations for individual layers from all studies above, we find an interesting pattern. Generally speaking the last layer or last couple layers in the architecture have higher variance compared to the rest. This is an indication that later layers tend to learn more from the loss than other layers. Some of the plots are shown in Figure 11. Most of the plots show wider standard deviations in the last couple layers, or a trend that from the first layer to the last, the standard deviation increases. This behaviour is also observed in [5]. There are a couple exceptions in our studies, which are "No Spatial Attention" and "No Positional Encoding". For "No Spatial Attention", the earlier layer gradients have wider standard deviations such that the behaviour is reversed. This could be due to the fact there are more learnable parameters in this model and with the given time the training could continue until the behaviour is reversed. For "No Positional Encoding", the same reversed behaviour is observed but only more severe. The first layer gradients are extremely wide in standard deviation while others are very narrow. This means the positional encoding information can be passed along throughout the layers for the model to learn the motion sequence.

Armed with this knowledge, we would like to understand how 7 layers instead of 8 can affect the performance and gradients. In Figure 6, study 13 shows the joint angle difference is quite similar to the benchmark, meaning one layer fewer produces negligible loss. However, in weight gradients (7 Attn Layer at bottom right hand corner) in Fig 11, the gradients in the first layer have wider standard deviation compared to the rest. From the second to the last layer, the trend seems to be increasing in standard deviation. The implication here is that the weight gradients may not be a clear indicator of how well the model performs, but the gradients are directly tied to how weights are updated in each step. If the gradient distribution can change dramatically from 8 layers to 7 layers, this could mean the training process can become unstable, or in other words, it can be hard

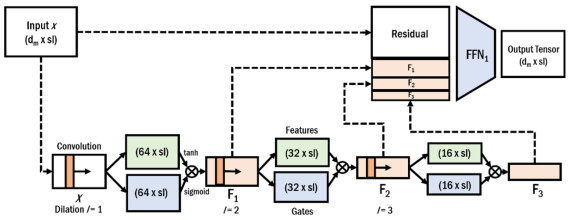

Figure 10. A convolution layer applied before feedforward layer, image taken from [13].

to train the model. In Figure 8, we see that the joint loss becomes noisy quickly early on in the training process. It is possible improving the stability of gradients throughout the layers can increase the learning speed in the training process, which leads to shorter learning time and/or increased accuracy.

Study 14: Simpler ST and Convolution Transformer

In the state-of-the-art ST Transformer [10], the inputs are in the shape of [sequence length, number of joints, joint size]. We wonder if we simply the inputs to [sequence length, number of joints x joint size], how well the model can learn. To further simply the architecture, we get the query, key, value vectors, and attention mechanism by transposing input (embedded) matrices. Each resulting QKV set will be used for one attention head. From previous studies, we learned that temporal attentions are more important than spatial attentions, so we use 32 temporal attention heads with 8 spatial attention heads.

In addition, we are adding a convolution layer as described in [13]. An example of the architecture is visualized in Figure 10. In our model, instead of 3 convolutions in series, we have 5 in series with kernel size [1 x 5], valid padding, and dilation rate of 2 to gradually decrease the joint dimension while keep the same sequence dimension. The feature representations from all convolution outputs are concatenated to the inputs before the feedforward layer. With this architecture, we notice during training, the speed is twice as fast compared to the benchmark even when the number of learnable parameters are about 30% more. We believe it is because the benchmark temporal attention block includes a big for loop to calculate attention mechanism for each joint. Since the training speed is faster, we are able to jam more iterations into the training, given the same 12hrs. From Figure 6, study 14 shows the performance is not as good as the benchmark, but it's comparable to RNN-SPL and LTD-10-10 performance reported in [10].

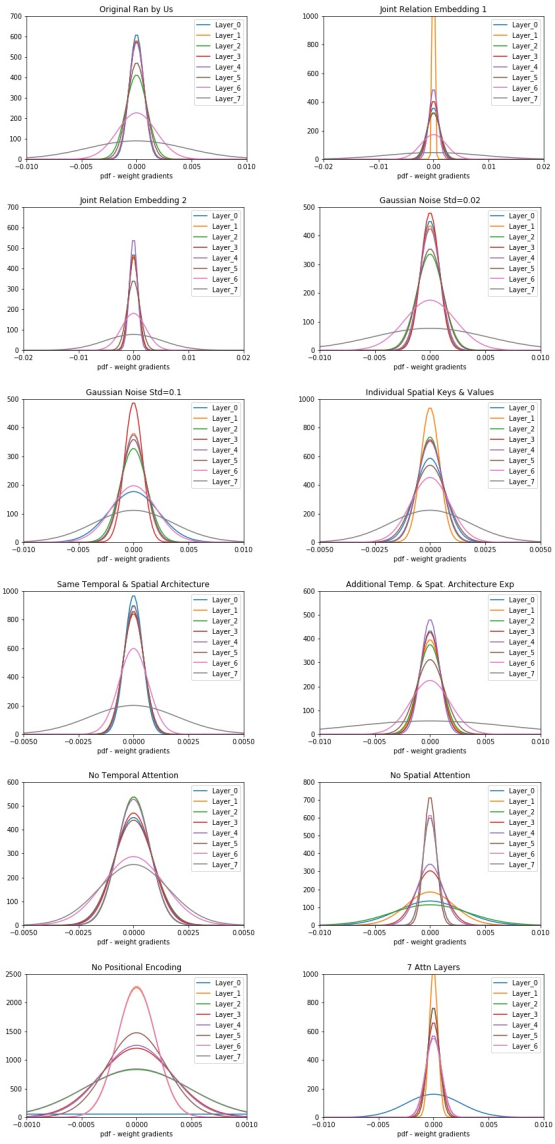

Figure 11. The $\delta Loss/\delta W$ distribution among layers. Most of them display higher standard deviation in the later layers.