MODELING HUMAN MOTION-CAPTURE DATA FOR CREATIVITY

by

Emily Napier

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2023

Dedicated to Hayley Bone.

Table of Contents

Abstract

Human motion-capture data can be represented, modeled, and generated through computational techniques. This thesis explores representations and strategies for querying, interpolating, and sequence modeling of motion-capture data. We employ spectral analysis of motion capture data to facilitate the query and comparison of movements, and identify target features for interpolation. We train a decoder-only transformer model on text-encoded motion-capture data, which we fine-tune for dance generation and movement classification. Our core contributions are defining interpolation and language model training procedures for generating motion-captured dance.

Acknowledgements

First and foremost I would like to thank Sageev Oore and Gavia Gray for their support during my master's degree. Dr. Gray provided me with attentive guidance and mentorship while taking me under her wing with patience and boundless support. Dr. Oore's compassion and thoughtful advice has been invaluable to me in broadening my horizons and expanding the scope of my research perspectives. I could not have completed this work without either of you.

I would also like to thank the members of Dr. Oore's research group for their insightful discussions. Scott Lowe went above and beyond his role as a mentor, and Hana Torabi and Ruis MacDonald were not only collaborators but also, more importantly, friends.

Next, I would like to thank David Rokeby, Douglas Eacho, and Xavier Snelgrove for their profound insights on movement and motion-capture. Thank you to Michael Thaut and Tristan Loria for sharing their knowledge of movement analysis and contributing the Parkinson's gait dataset. Thank you Vlado Keselj for guiding my first project during the program, which led to my first paper.

Finally, thank you Hayley Bone, Diana Rutherford from The Dance Institute, Sara Corkum and Ranna Mirsaeidghazi from Motion Dance Centre, and Solène Bernier for contributing your knowledge and experience as movement educators and performers.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Funding for this research was provided by CIFAR, NSERC, and Research Nova Scotia.

Chapter 1

Introduction

Advancements in machine learning have presented new opportunities for effective modeling and analysis of human movement. In this thesis we consider human movement through the lens of motion-capture data, and test our models in creative applications. We explore ways of representing motion-capture data through spectral analysis, and implement state of the art machine learning techniques to model motion-capture sequences. We focus on generative modeling for dance data through manipulation of spectral features and generative language modeling, and focus on analysis of data through classification of movement. Identifying patterns across individuals that are invariant to individual skeletal structure is an important constraint for both practical and creative settings, making it a candidate for artificial intelligence based solutions.

The field of modeling human movement as motion-capture data, skeleton keypoints, or body meshes has been investigated by computer vision and sequence modeling researchers for applications including dance, sports, computer graphics and health and safety. The target audience of this work is the machine learning for performing arts and machine learning community. One of the common goals for this community is capturing real-time human movement for interactive systems and movement analysis. We aim to identify meaningful representations and models of movement through computational techniques which capture bio-mechanical properties as well as artistic or practical intent, and test these representations on downstream tasks. In dance systems this includes prompting and supporting creative practices where artists perform artificially generated or conceptualized movement, use artificially generated movement in the choreographic process, and conduct virtual rehearsals. These goals differ from the computer graphics community who create visually convincing or appealing movement digitally.

In this work we employ representations of motion-capture data that are suitable for artificial-intelligence-based solutions. We recognize similarities in the temporal

relationship of text and motion-capture data, and implement techniques to compress and discretize motion-capture data as a text-based encoding that is suitable for language model architectures. In doing so we make adjustments and accommodate the higher dimensionality and continuity of the data. Similar to the natural language processing field, we aim to learn embeddings of motion-capture data that can be used for a range of downstream tasks. We explore techniques for representing human movement in a way that it can be compared, manipulated, predicted, generated and classified.

1.1 Thesis outline

This thesis is based on two papers that I co-authored, which explore two questions about human motion-capture processing:

1. The first paper, Spectral Analysis for Dance Movement Query and Interpolation [50], considers how to transform motion-capture data for query and interpolation.

2. The second paper, Sequence Modeling of Motion-Captured Dance [49], considers how to generate and classify motion-capture data.

The first paper, discussed in Chapter 3, compares transformations as target metrics for query and interpolation. We evaluate our results through quantitative and qualitative analysis, with input from experts in the dance field. Our dance movement query identifies movements with similar spectra, and our interpolation technique inspired by neural-style transfer demonstrates generation of novel and danceable outputs. Quantitative results are analyzed by the k-nearest neighbors (kNN) algorithm, and qualitative results are discussed in terms of the potential utility of these algorithms in educational and professional dance settings.

The second paper, discussed in Chapter 4, trains language models (LMs) on motion-capture data. We evaluate spectral and machine learning transformations within a lossy-encoding procedure for converting motion-capture data to text, and train models on this text-based corpus. We attempt to compress the data in both the joint rotation and time dimensions. Our training procedure involves pre-training models on the largest publicly available motion-capture dataset and fine-tuning them on task specific datasets. We add classification heads for multiclass classification tasks and generate novel motions with conditioning tokens and select joints as prompts. We

demonstrate that this approach expands the range of outputs in comparison to our first paper while maintaining control over the generated movements.

The main contributions of the first paper are:

- an exploration of spectral analysis for dance query,
- an interpolation technique for motion generation.

The main contributions of the second paper are:

- a tokenization technique for converting motion-cature data to text,
- a pre-training and finetuning procedure for language models trained on that text,
- conditional generation of motion-captured dance,
- classification of motion-capture data.

The primary contributions of each paper led to generative motion-captured dance data, through interpolation or conditional generation. Both of these techniques work towards the same goals of capturing meaningful representations of motion-capture data, and manipulating the sequences of movement to generate novel outputs. We discuss the potential of future iterations of the work and the utility of each existing technique in rehearsal and performance settings.

Both papers were co-authored by Gavia Gray, who guided and reviewed the technical implementation of my work.

Chapter 2

Background

2.1 Datasets

This work focuses on motion-capture data; specifically we focus on the AIST++ dataset and the AMASS dataset. The AIST++ dataset is a collection of 5.2 hours of dance motion sequences [39] processed from a corresponding video database [63]. The dataset comprises 30 dancers performing dance motions across 10 genres, and includes basic and advanced movements performed with one of 60 musical pieces from 80 to 135 beats-per-minute (BPM). The AMASS [43] dataset is the largest publicly available dataset of human motion, comprised of multiple smaller optical marker-based datasets. It includes over 11000 motions from over 300 subjects, totaling over 40 hours of motion-capture data.

2.2 Pose representations

Motion data is typically modeled as a sequence of frames, wherein each frame is represented using a set of joint angles referred to as a pose, and body shape parameters describing a body model. Both the AMASS and AIST++ datasets are represented in SMPL [40] notation (see Section A.3 for details).

Using body shape parameters, the 3D positions in space occupied by each joint can be recovered by forward kinematics. The sequence of motion can therefore be represented as a sequence of joint angles or a sequence of 3D positions through time. By combining the axis-angle 3D rotation vectors into matrices which describe pose over time, we investigate ways of reducing the information required to describe and model movements. We simplify the dataset by decreasing the frame rate from 60 fps to 20 fps, and by removing joints that have low visual impact on rendered motions (see Section A.4 for details).

2.3 Related Work

Modeling human motion can be divided into a series of problems that can be solved with task-specific or end-to-end architectures [76]. The full pipeline that is addressed in the literature includes:

- motion-capture from either RGB videos, optical-based markers, or inerial measurement units,
- learning motion sequence representations, and
- applying this knowledge to downstream tasks.

The first step, capturing motion data from RBG videos, can be subdivided into capturing 2D or 3D keypoints, and 3D meshes. Popular methods of capturing 2D keypoints include OpenPose [9] and MediaPipe [42], where the most common architectures use convolutional Neural Networks (CNNs). 3D keypoint detection has employed physics-informed modeling [70] or end-to-end approaches for joint mesh and 3D keypoint recovery [60, 34]. The SMPL [40] framework used in this work is a unified representation for motion-capture data that includes 3D meshes and 3D joint angles. This step in the pipeline is not the focus of this work, however the video to SPML tool we use may impact our results.

Learning motion sequence representations is a precursor to performance on downstream tasks. Past work has explored various model architectures including generic neural networks and convolutional networks, temporal and graph convolutional networks, and recurrent neural networks [37, 18, 19]. Motion-capture specific model architectures or loss functions that consider the hierarchical structure of the body model [2], bilateral symmetry [15], or train on motion-capture specific signals such as geodesic rotation error [26] have improved motion-capture modeling. The introduction of the AMASS [43] dataset paired with the emergence of the transformer [66] outperformed standard sequence models such as RNNs and LSTMs. The first transformer-based models predicted continuous motion-capture data, [1] and more recently have been used with discrete motion representations similar to those we discuss in Chapter 4. Auto-Encoders (AEs) and Variational Auto-Encoders (VAEs) [54] are one of the most common models to learn latent space representations of human motion-capture

data, and VQ-VAEs which convert motions to discrete latent space representations have been implemented as a tokenization strategy for modeling motion-capture data with language models [41]. Finally, transformer-based diffusion models [75] have recently been used to model motion-capture data, and show promising results with multi-modal datasets [73].

Improvements in learning motion sequence representations also drive improvement in downstream tasks. With the tools mentioned above, the downstream tasks that we focus on are:

- generative modeling for dance movement, and
- motion classification.

AI Choreographer [38] and Transflower [64] are examples of deep learning based approaches which use the AIST++ dataset with transformers. AI Choreographer conditions dance movement on music, and Transflower adds a normalizing flow to model the modalities together. ChoreoNet [72] builds on generating dance movement conditioned on musical prompts by organizing sequences of motions according to music. We focus on generating movement conditioned on existing examples of dance.

Motion classification, querying, and description look for patterns across movements. These patterns can help identify movements that are correlated with certain activities like sports, or certain emotions or characters in performing arts or video games. Adversarial learning techniques with generative adversarial networks (GANs) [71] have driven improvements in classification tasks, and expanding beyond categorical classification has become possible due to increased availability of paired data. Text-based motion description [74] in particular can be utilized for both motion querying and classification with more diverse outputs. Finally, improving motion sequence modeling by considering spatio-temporal relationships [57], increasing the motion window length, and increases in publicly available dataset size are factors that lead to better performance in both generative and classification tasks.

Detailed background methodology and chapter-specific related work can be found in Section 3.3 for Chapter 3, and Section 4.3 for Chapter 4.

Chapter 3

Spectral analysis for dance movement query and interpolation

3.1 Preface

This chapter is based on the paper presented at the Movement and Computing Conference in 2022 [50]. One of the constraints of being successful in training deep learning models is being able extract important features from the data though representation learning [23]. Time series are often represented spectrally; we wish to understand how to interpret this representation in the context of human movement data.

The intention for this paper was to explore and learn about motion-capture data and the AIST++ dataset, and to implement simple computational tools for dance and movement. We focus on identifying transformations of the data that were effective for query and interpolation tasks. Evaluation of the work included seeking expert user feedback on the utility of the tools in educational and professional dance settings.

My contribution to this work included:

- implementing and testing the STFT and DCT transformations,
- implementing and testing the query algorithm,
- implementing and testing loss functions for the interpolation,
- obtaining expert user feedback,
- co-writing the paper.

3.2 Introduction

Quantifying and generating dance has been explored through various movement analysis, interpolation, and prediction techniques. Systems that generate novel motions

provide choreographers with the ability to explore their own ideas and seek inspiration for new ideas digitally. One of the constraints of these systems is the control with which choreographers can interact creatively with generated motions. This work generates suggestions for choreographers to expand their typical vocabulary of movement, while providing the ability to iterate their creation towards target statistics.

Digital representations of motion, such as the joint angles typically extracted by motion capture systems [24], can be analysed to extract or infer characteristics of a dancer's motion. In this chapter, we demonstrate how spectral features of joint angles can be used to modify, search for and visualize similar motions. We aim to define metrics to quantify and visualize similarity between motions, thus contributing to defining dance creations computationally with artistically relevant results.

First, we use the Short-time Fourier Transform (STFT) of joint angles and the k-nearest neighbors (kNN) algorithm to rank short sequences of the AIST++ database in order of similarity. This quantifies differences between motions, and can return sequences based on an input motion and the target similarity.

Next, we propose a motion interpolation technique that is inspired by, and somewhat parallels, *neural style transfer* [22]. Style transfer modifies images (e.g. paintings or photographs of scenery) to maintain their low frequency structure (related to theme or subject), while adapting their higher frequency characteristics (related to style or color) to match those of another provided image. We demonstrate and confirm through expert user feedback that the high frequency components of motion contain information related to movement complexity and quality, and optimize the Discrete Cosine Transform (DCT) of a motion to visualize the dancer performing a movement with different low frequency components.

We show that this representation of motion could provide a basis of digital tools that demonstrate various levels of complexity in choreography, measure similarity between movements, and generate new movements from provided examples.

3.3 Background

We describe a neural style transfer [22, 46] inspired technique that uses spectral features with automatic differentiation to interpolate movements. In neural style transfer, a white noise image is jointly optimized to contain content based features

of one image and the style and texture of another. The features are computed with convolutional layers, where layers earlier in the network capture local elements like colour and texture, and later layers capture more complex patterns like objects. The white noise image is modified towards the target features through gradient descent. Figure 3.1 shows examples from the original paper [22] where the original image is visualized with different artistic styles.

Inspired by this, we use spectral features of a target motion to drive incremental changes made to an input motion. The original motions consist of a time series of joint angles, and we effectively adjust the joint angles of the input motion in such a way that its spectral features will more closely resemble those of the target motion. In this approach, the time-series of joint angles are comparable to the model parameters in a traditional machine learning system. They are updated according to the objective function, which drives the interpolation by measuring the similarity between the spectral features of the model parameters and the target motion. This optimization is feasible thanks to automatic differentiation, which allows us to compute gradients and make adjustments in the appropriate directions.

3.3.1 Related work

Digital methods of querying and imagining dance have been explored by various dance and computational experts [3]. Early implementations blended sequences of discretized movements, including notations like Laban Movement Analysis (LMA) [67], which dicretizes movements into spatial and temporal components, and Boolean features or matrix representations [48, 47] which define body pose with respect to the surrounding joints (eg. the right foot is in front of the left foot). Query by similarity approaches have been contributed by Muller at al., and BalOnSe [17], which identified similar movements based on their ballet.owl ontology and Labanotation [16], a dance notation system based on LMA. Our work identifies spectral embeddings as a way to compare motions automatically and to reduce the requirement for manual labeling.

To automate generation of new motion sequences, systems have explored altering motions through genetic algorithms [35, 10] or tracing paths through latent space representations of motion capture databases to generate novel motions [5]. To further refine these approaches, target styles can guide the creation of novel movement.

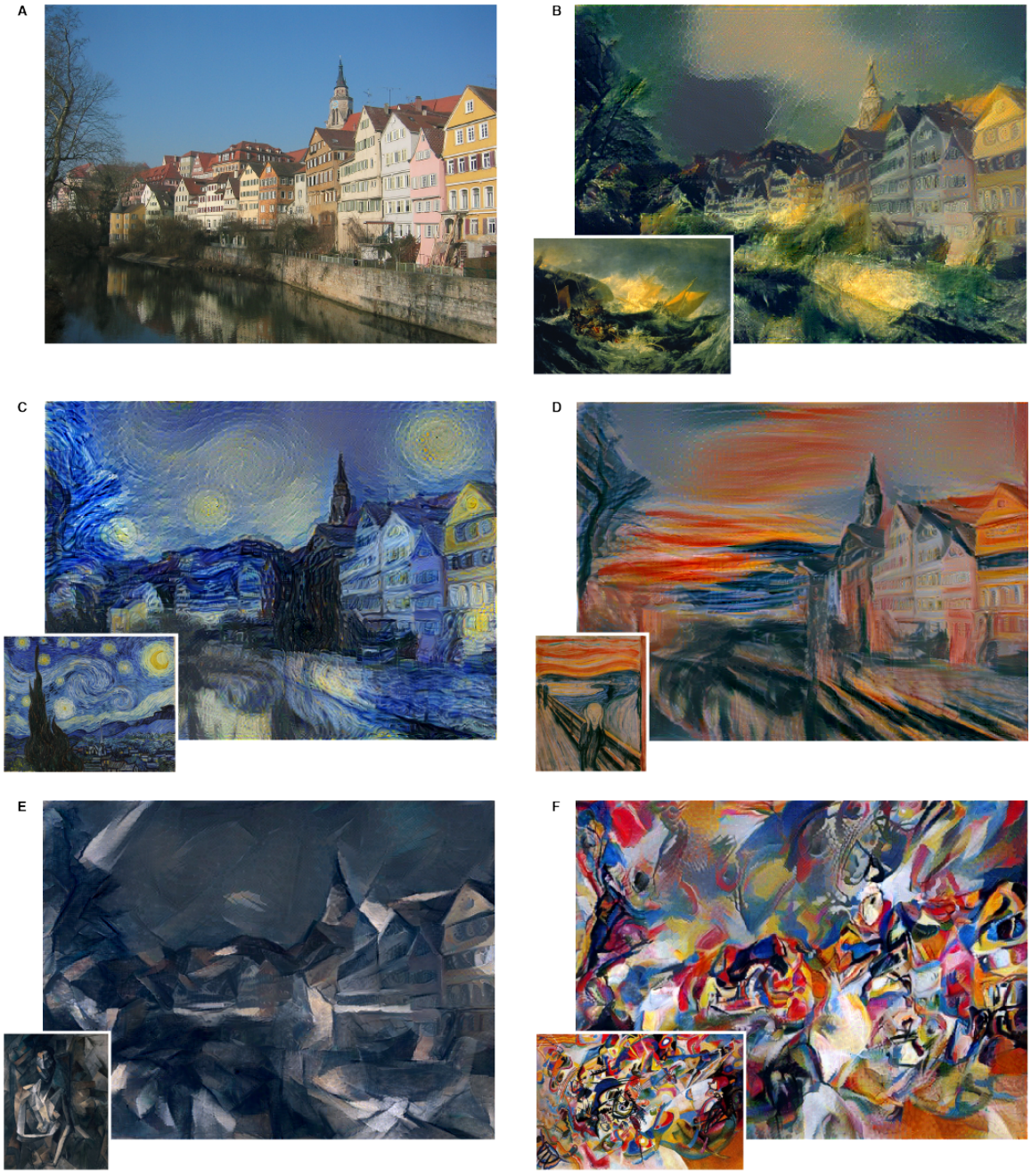


Figure 3.1: Examples of neural style transfer on images.

Style Machines [8] used stylistic Hidden Markov Models to generate motions in a different dance genre. SmartBody [56] and Dong et al. [12] used signal decomposition techniques to separate movement into functional and stylistic components that can be blended to create motions with new effects. Our work compares and identifies transforms that capture various components of movement, and alters them through gradient decent to generate novel and continuous outputs.

Fourier analysis of motion data has been implemented in multiple works, including gait analysis and motion recognition [27, 55]. Computational techniques have also been used in dance education settings to encourage deeper understanding of movement, and Fast Fourier transforms have been suggested as a metric to define the timing of motion [4].

3.4 Motion querying

Motion sequences in the query were defined as 128 frame (2.13 s) sub-sequences of the basic dance videos in the AIST++ dataset. Subsequences captured a short instance of movement that incorporated 2-5 beats dependent on the music's BPM, and the 72 dimensional joint rotation vector that defined the pose. To find the closest matches for a query sub-sequence, $\boldsymbol{S}_{\text{input}}$, it is compared to all other motion sub-sequences, $\boldsymbol{S}_{\text{x}}$, in the dataset. The comparison first computes the STFT spectrograms [58] for both sub-sequences $\boldsymbol{S}_{\text{input}}$ and $\boldsymbol{S}_{\text{x}}$, and then computes their cosine similarity. Matching sequences can then be ranked in order of similarity scores.

The STFT is defined in eq 3.1 for signal $\boldsymbol{S}_{\text{x}}[n]$ and Hann window w[n], which is defined in eq 3.2:

$$\text{z} = \text{STFT}(\boldsymbol{S}_{\text{x}}[m]) = \sum_{n=-\infty}^{\infty} \boldsymbol{S}_{\text{x}}[n]\ \text{w}[n-m]\, e^{-jwn} \tag{3.1}$$

$$\text{w}[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N}\right)\right] \text{ for } 0 < n < N. \tag{3.2}$$

We selected 16 frame (267 ms) windows for the spectrogram to align with timescales that are relevant to rhythm [59] and choreographic timing. Our objective in selecting this window length was to capture the half-beats at the maximum tempo of music in the dataset, which was 130 BPM.

Cosine similarity is described in eq 3.3, where $\boldsymbol{z}_{\mathrm{a}}$ and $\boldsymbol{z}_{\mathrm{b}}$ represent the result of the transformations of motions a and b, which in the case of the motion sequence query are $\boldsymbol{S}_{\mathrm{input}}$ and $\boldsymbol{S}_{\mathrm{x}}$:

$$\mathrm{d}_{\cos}(\boldsymbol{z}_{\mathrm{a}}, \boldsymbol{z}_{\mathrm{b}}) = \frac{\boldsymbol{z}_{\mathrm{a}} \cdot \boldsymbol{z}_{\mathrm{b}}}{\|\mathrm{z}_{\mathrm{a}}\| \|\mathrm{z}_{\mathrm{b}}\|}. \tag{3.3}$$

The performance of the query was measured by accuracy and entropy metrics that compared the genre and dancer labels of the k-nearest sub-sequences, $\boldsymbol{S}_{\mathrm{x}}$, in the query dataset to the labels associated with $\boldsymbol{S}_{\mathrm{input}}$. Accuracy was calculated by the predicted genre or dancer label of $\boldsymbol{S}_{\mathrm{input}}$, which was defined by the majority vote of the labels from the k-nearest sub-sequences. Entropy was calculated based on the labels of the same k sub-sequences, where low entropy indicated that these sub-sequences were from the same category.

Entropy is defined in eq 3.4 where $\mathrm{P}(g_i)$ is the empirical probability of the k-nearest sub-sequences, $\boldsymbol{S}_{\mathrm{x}}$, being from a particular category, g_i:

$$\mathrm{H}(g) = -\sum_{i=1}^{n} \mathrm{P}(g_i|\boldsymbol{S}_{\mathrm{x}}) \log \mathrm{P}(g_i|\boldsymbol{S}_{\mathrm{x}}). \tag{3.4}$$

Our results compare cosine similarity and euclidean distance as common metrics for measuring distances in vector spaces with the kNN algorithm.

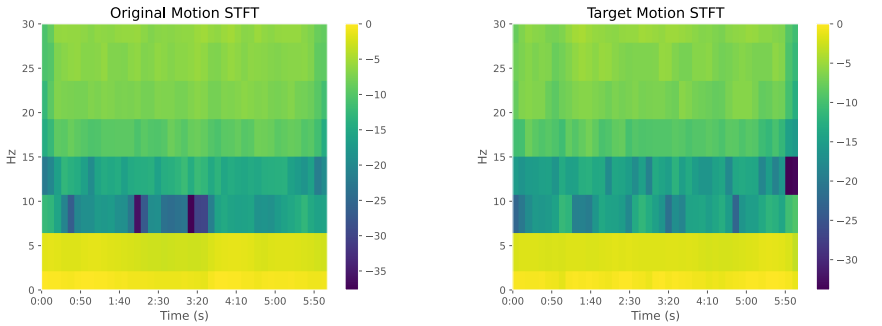
In addition to this experiment that was part of the original paper, we assessed the performance of the DCT and PCA on the query task.

The DCT is defined in eq 3.5 for a motion, $\boldsymbol{S}$:

$$\mathrm{DCT}(\boldsymbol{S}) = \sum_{n=1}^{N-1} \boldsymbol{S}_n \cos\left[\frac{\pi}{n}\left(n + \frac{1}{2}\right)k\right] \text{ for } k = 0, \ldots, N-1. \tag{3.5}$$

The DCT was considered due to its success in JPEG image compression; it is effective in producing a sparse subset of coefficients that carry critical information about an image. Assuming the same principles could be applied to motion data, we consider the DCT in the query procedure. PCA can be used as a dimensionality reduction technique, and is a common feature extraction method in machine learning. PCA is described in Section A.1.

In our original query experiment we only considered the frequency information of the STFT in the form of a spectrogram, making the transformation lossy. The DCT is lossless, so the resulting transform does not reduce the information content of the



(a) STFT spectrum of $\boldsymbol{M}_{\text{input}}$ (b) STFT spectrum of $\boldsymbol{M}_{\text{target}}$

Figure 3.2: Example spectrograms of a motion sub-sequences.

sub-sequence. In order to improve our query results and to compress the sub-sequence in a lossy way, we only consider the low-frequency components of the DCT in the query procedure.

3.5 Motion interpolation

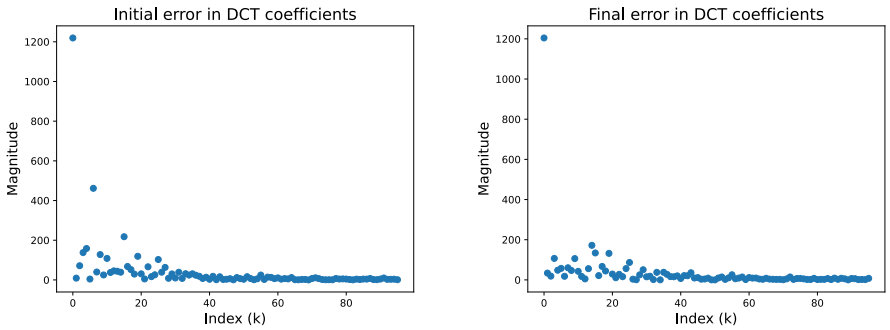
Motion interpolation compared two examples, $\boldsymbol{M}_{\text{input}}$ and $\boldsymbol{M}_{\text{target}}$, from the AIST++ dataset and generated a new motion, $\boldsymbol{M}_{\text{output}}$, that shared features of each. The first motion, $\boldsymbol{M}_{\text{input}}$, was used to create the starting parameters of $\boldsymbol{M}_{\text{output}}$, which was adjusted toward the second motion, $\boldsymbol{M}_{\text{target}}$. A range of spectral features $\boldsymbol{F}$, computed via arbitrary function f, were selected for the optimization, and the mean-squared-error loss between $\boldsymbol{F}_{\text{a}} = f(\boldsymbol{M}_{\text{output}})$ and $\boldsymbol{F}_{\text{b}} = f(\boldsymbol{M}_{\text{target}})$ was calculated in that space.

An example of the function f that we use to compute the spectral features was the STFT. The STFT is shown in Figure 3.2 for an original and target motion pair by computing the principal component of the transformation across all of the joint rotations. This defines a spectral embedding of the STFT, as described in Section A.1.1.

Mean-squared-error (MSE) loss is defined in eq 3.6, where $\boldsymbol{F}_{\text{a}}$ and $\boldsymbol{F}_{\text{b}}$ represent the range of spectral features selected for the optimization for each joint rotation, i:

$$\text{MSE}(\boldsymbol{F}_{\text{a}}, \boldsymbol{F}_{\text{b}}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{F}_{\text{a},i} - \boldsymbol{F}_{\text{b},i})^2. \tag{3.6}$$

An example of the initial and final error (absolute difference) between the spectral



(a) Initial error between DCT coefficients of $\boldsymbol{M}_{\text{input}}$ and $\boldsymbol{M}_{\text{target}}$

(b) Final error between DCT coefficients of $\boldsymbol{M}_{\text{output}}$ and $\boldsymbol{M}_{\text{target}}$

Figure 3.3: Error between the spectral embedding of DCT coefficients before and after interpolation. Error decreased after interpolation, but is still greater than 0 to retain features of $\boldsymbol{M}_{\text{input}}$.

embedding of the DCT coefficients, as described in A.1.1, is shown in Figure 3.3.

Automatic differentiation allows efficient gradient computation of the motion features which can be used to gradually optimize $\boldsymbol{M}_{\text{output}}$ towards $\boldsymbol{M}_{\text{target}}$ with gradient descent. At each step of the optimization the axis-angle rotation vectors of each joint were constrained to valid rotations by their parameterization. The STFT, DCT, and PCA transforms from Section 3.4 were considered as objective functions, f.

Examples of $\boldsymbol{M}_{\text{output}}$ were demonstrated and assessed qualitatively, and the cosine similarity between the STFTs of $\boldsymbol{M}_{\text{input}}$, $\boldsymbol{M}_{\text{target}}$ and $\boldsymbol{M}_{\text{output}}$ were calculated. The performance of the interpolation was assessed based on its ability to move $\boldsymbol{M}_{\text{output}}$ closer to the target both visually and by the similarity of the spectra.

3.6 Results

The comparison of cosine similarity and euclidean distance of the STFT in the kNN algorithm showed that cosine similarity performed better at returning motions of the same genre and dancer. The accuracy was calculated by the majority vote of the labels for the $k = 10$ most similar sub-sequences, and the entropy was defined by eq 3.4 with the $k = 10$ most similar sub-sequences. The accuracy and entropy for both approaches are shown in Table 3.1.

Examples of the query results were demonstrated with $\boldsymbol{S}_{\text{input}}$, one $\boldsymbol{S}_{\text{x}}$ with high ranking $\text{d}_{\cos}$ score according to the metric in eq 3.3, and one $\boldsymbol{S}_{\text{x}}$ with lower ranking $\text{d}_{\cos}$

score. Demo Video 1 (https://youtu.be/mE6lCxYs06k) demonstrates three examples from the original paper results.

	Genre		Dancer	
	Accuracy (%) ↑	Entropy (ban) ↓	Accuracy (%) ↑	Entropy (ban) ↓
Cosine Similarity	**85.0**	**0.126**	**78.3**	**0.238**
Euclidean Distance	83.3	0.149	75.0	0.247

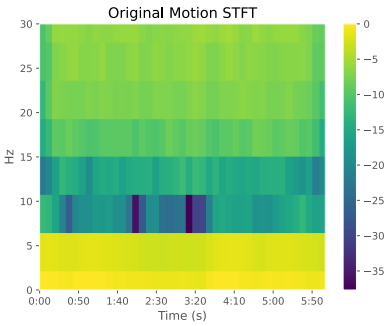
Table 3.1: Accuracy and entropy metrics of cosine similarity and euclidean distance approaches to ranking similarity of motion spectra with the kNN algorithm. Cosine similarity performed better in both metrics and categories.

	Genre		Dancer	
	Accuracy (%) ↑	Entropy (ban) ↓	Accuracy (%) ↑	Entropy (ban) ↓
DCT	88.3	**0.0889**	**83.3**	0.233
STFT	85.0	0.126	78.3	0.238
PCA	88.3	0.116	81.7	**0.220**

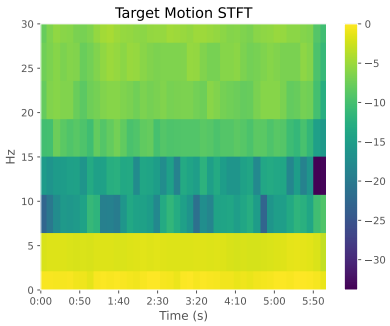
Table 3.2: Accuracy and entropy metrics of ranking similarity of DCT, STFT and PCA motion spectra with the kNN algorithm. The DCT and PCA results are computed with 16 components. The DCT and PCA performed better than the STFT on both metrics and categories.

Table 3.2 compares the kNN query algorithm results with PCA, the DCT, and the STFT over the time dimension with cosine similarity. We demonstrate qualitative results in Demo Video 2 (https://youtu.be/cJPI3bBztms). PCA and the DCT perform slightly better at the query task quantitatively, however we can see that the STFT query results demonstrate movements that are more similar and are a better match for the timing of the original movement. In both examples we can see that the results of the DCT query do not capture similar choreography. PCA performs better than the DCT, however the STFT captures the timing of the query dancer better in these examples.

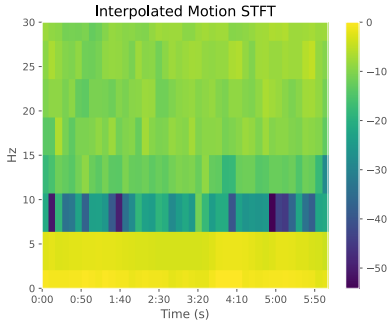
This spectral embedding of the STFT is shown in Figure 3.4 for an example of $\boldsymbol{M}_{\text{input}}$, $\boldsymbol{M}_{\text{target}}$, and $\boldsymbol{M}_{\text{output}}$ from the dataset. Figure 3.4c shows the spectral embedding of $\boldsymbol{M}_{\text{output}}$, which demonstrates that the energy in the high frequency bins increased during optimization. This resulted in jittery, discontinuous movements, which indicated that this approach doesn't retain continuity of motion during optimization due to the separation of frequency and phase information. This meant



(a) STFT spectrum of $\boldsymbol{M}_{\text{input}}$



(b) STFT spectrum of $\boldsymbol{M}_{\text{target}}$



(c) STFT spectrum of $\boldsymbol{M}_{\text{output}}$

Figure 3.4: Spectrograms of $\boldsymbol{M}_{\text{input}}$, $\boldsymbol{M}_{\text{target}}$, and $\boldsymbol{M}_{\text{output}}$. Figure 3.4c demonstrates higher energy (lighter tones) in the high frequency range, which resulted in jittery movements.

that the STFT was not feasible as an optimization target, however the spectral embedding of an example motion from the original paper is shown in Demo Video 3 (https://youtu.be/geI67TnhQDw) to demonstrate how motion correlates with energy in the frequency domain. This also shows the frequency components of motion that were used in the motion query.

We also tested motion interpolation with PCA and the DCT, and found that PCA resulted in jittery movements like the STFT results. The DCT was able to effectively capture and interpolate the low frequency components of movement. To more intuitively understand the effect of using certain frequency ranges to drive the optimization, we reconstructed motions where select DCT frequencies have been filtered out. In Demo Video 4 (https://youtu.be/G6VfHxkEJG0) we show how a motion from the original paper looks after we apply a low-pass filter (LPF) where we have (a) filtered out only high frequency components (above the green line in Figure 3.5),

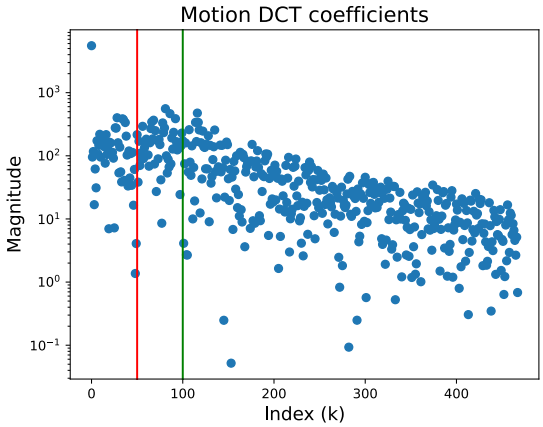


Figure 3.5: DCT coefficients of the low (left of the red line) and low and mid-frequencies (left of the green line) that were kept after filtering. Note that the excluded frequencies (right of the dividers) have lower magnitude and therefore contribute less to the reconstruction.

and (b) filtered out high and mid-frequency components (above the red line in Figure 3.5). Increasing the range of frequencies in the reconstruction naturally leads to a higher-fidelity version of the motion, but the filtered versions of the DCT also have qualitative characteristics that the expert users described and commented on.

The effectiveness of the motion interpolation system was assessed by its ability to convey motion features of both $\boldsymbol{M}_{\text{input}}$ and $\boldsymbol{M}_{\text{target}}$ in the generated motion. Each row of Table 3.3 lists all pairwise cosine similarities between: $\boldsymbol{M}_{\text{input}}$, $\boldsymbol{M}_{\text{target}}$, and the corresponding generated (interpolated) $\boldsymbol{M}_{\text{output}}$ for the DCT-based interpolations presented in the original paper. DCT-based qualitative results can be found in Demo Video 5 (https://youtu.be/-wb3AQ3j1WE) where the asterisk indicates whether the music playing is associated with $\boldsymbol{M}_{\text{input}}$ or $\boldsymbol{M}_{\text{target}}$.

For the DCT-based interpolation we saw that the interpolated motions were more similar to both the original and target motions than they were to each other. This indicates that the interpolated motion exists between the two inputs in the query space, which was confirmed qualitatively from the motion demonstrations and feedback from dance experts. We found that the PCA-based interpolation had poor quantitative and qualitative results with similar issues of jittery motions as seen with the STFT. We show a comparison of the DCT and PCA-based interpolations in Demo Video 6 (https://youtu.be/5lyBIunewV0) and Table 3.4.

Example	O→T	I→O	I→T
1	0.662	0.899	0.847
2	0.539	0.835	0.812
3	0.637	0.866	0.744

Table 3.3: Cosine similarity between original (O), target (T), and interpolated (I) sequences in Demo Video 5. The interpolated sequences had high similarity to the original and target sequences.

	DCT			PCA		
Example	O→T	I→O	I→T	O→T	I→O	I→T
1	0.637	0.866	0.744	0.637	0.991	0.634
2	0.662	0.900	0.847	0.662	0.995	0.687

Table 3.4: Cosine similarity between original (O), target (T), and interpolated (I) sequences in Demo Video 6. The DCT-based interpolations have increased similarity to the target motions (results are comparable to Table 3.3), and the PCA-based interpolations remain very similar to the original motions and fail to demonstrate a valid interpolation.

3.7 Expert user feedback

The results of the original study, with the STFT-based query and DCT-based interpolation methods, were sent to six expert dance artists who are performers, educators, and choreographers. They watched alternate versions of Demo Video 1 (Section 3.7.1), Demo Video 4 (Section 3.7.3), and Demo Video 5 (Section 3.7.2), that did not include text and randomized the order of the motions on the screen from left to right. Expert feedback from sections 3.7.1 and 3.7.2 provide qualitative results of our two main contributions, and feedback from Section 3.7.3 assesses our selection of low frequency components of the DCT as a metric for motion interpolation.

In the assessment of the motion query from Section 3.4 experts were provided with three query results. Each result had one original motion, $\boldsymbol{S}_{\text{input}}$, one high ranking motion, $\boldsymbol{S}_{\text{x},h}$ (i.e. similar according to the similarity metric defined by eq 3.3 and features defined by eq 3.1), and one low-ranking motion, $\boldsymbol{S}_{\text{x},l}$ (i.e. with a low $\text{d}_{\cos}$ score). They were asked to identify which two of the three motions were the most similar.

In the motion interpolation assessment from Section 3.5 they were provided with 3 examples of motion interpolation, and were asked to identify which motions were

the interpolation from $\boldsymbol{M}_{\text{input}}$, $\boldsymbol{M}_{\text{target}}$, and $\boldsymbol{M}_{\text{output}}$.

Finally, in Section 3.7.1, they were asked to identify and comment on the differences between the stronger and weaker dancers shown. The experts were also asked to comment on whether they could conceptualize a utility for these tools in the context of choreography generation or for educational purposes.

3.7.1 Motion sequence query results

In this Section users were asked to identify which two motions were the most similar. Users averaged 94 % accuracy in their responses, with comments indicating that "it was clear which were the most similar". When asked about the utility of a motion query system, 5 out of the 6 indicated that there are scenarios where they could use this tool in their work.

3.7.2 Motion interpolation results

The motion interpolation results showed that identifying the interpolated motion was a difficult task. Users responded with uncertainty and described the task as challenging. We can infer based on the users being unable to identify $\boldsymbol{M}_{\text{output}}$ as an out of distribution example that the interpolated motions were feasible dance steps that were still similar to the original and target sequences.

Use cases for this tool focused on choreography generation and exploration. 4 of the 6 participants indicated they could identify a potential use case.

3.7.3 Low-pass filtered motion

This section assessed whether the low frequency components of the DCT captured basic choreography of the motion while removing complexity. Users were asked to identify which dancer was the better performer, and all users indicated that the dancers demonstrating the filtered version of the motion demonstrated less energy and effort.

3.8 Discussion

Dance performance often contains sections where performers are expected to match each other's pose and timing as similarly as possible. The motion query provides a tool that can measure a dancer's similarity to a target motion or dancer. One expert suggested a use case for improving technique where teachers could "show a visual of the 'correct' movement and compare it with the dancer's own recorded movement, allowing them to adjust and match as necessary". Another user suggested using the query to identify movements from professional dancers and choreographers that are similar to motions students are learning in class.

When asked about choreography use cases, one user suggested "identifying classical variations which contain a particular co-ordination or movement". Another user pointed out that in industry where large volumes of work have been recorded, "teachers, pedagogues, and examining organizations could use this tool to search syllabus work to find linking and progressing steps", and "choreographers and rehearsal directors could use it to browse through archives of rehearsal material and performance footage".

Use cases for motion interpolation included "augmenting variations in choreography and movement" and exploring "more complex and layered choreography". Increasing accessibility to choreographers was another theme highlighted by users, with one suggesting that for "someone facing physical challenges wishing to create movement for people considered more able, the interpolation tool could help that person experiment with movements virtually that they are unable to take on physically". Another user suggested that in their experience "combining movements, especially ones we can recognize with ones that are less familiar, can challenge the body and brain to form new pathways and move differently".

Users described the dancers expressing the filtered movements in Section 3.7.3 as not "using the full range of motion in their joints", having "the complexity of motion" of the other dancer, or "completing movements". The unfiltered motions were described as "using more extension" and having "more dynamic movement qualities". Use cases for this video focused on dance education, particularly when "teaching the same choreography to a class with students of varying levels and capabilities". This indicated that filtered motions could be used to demonstrate simpler versions of the

original motion, and additional spectral features can be added as students are able to express more complexity in their movements.

Our query expands on existing work comparing motion by LMA or binary feature representations by measuring the similarity between their spectral features. In our user study, we found that dance experts thought that our search tool could help choreographers and students examine existing work and improve their ability to replicate motions. Our motion interpolation method suggests a new choreography generation tool that specifies target statistics while optimizing joint angles directly, and expert user feedback supported that defining a target metric for generated motion through another motion was appealing and interpretable to choreographers.

3.9 Summary

This chapter explores spectral analysis as a method for querying and interpolating movements. We found that STFT spectrograms capture the frequency components of movement in a way that is effective for querying tasks, and the DCT is an effective metric for style-transfer inspired interpolation. This demonstrates that spectral embeddings can help us to access semantic information in motion, which allows the style embedding to be altered. Experts were able to conceptualize uses for a search tool that can identify and measure similar movements in both educational and professional settings. The concept of motion interpolation was interpretable to dance experts and created novel movements that are continuous and often dancable. Experts identified opportunities for an interpolation tool in their choreographic process, but noted the challenges of working with digital tools in educational settings.

Chapter 4

Sequence modeling of motion-captured data

4.1 Preface

This chapter is based on the workshop paper [49] presented at the "Machine Learning for Creativity and Design" workshop at NeurIPS in 2022. Our primary goal was to learn embeddings of human motion which capture a wide range of movement and lead to the generation of novel dance sequences. We train a transformer model on discretized motion-captured data, which is adapted to perform downstream generation and classification tasks.

Our main creative objective was to generate movements that have more range than our interpolations from Chapter 3, while offering artists similar controls over the output. We utilize the generative models from Chapters 3 and 4 in rehearsal and performance, and note the utility of the tools in both the choreographic process and as part of a visual performance.

My contribution to this work included:

- implementing and testing tokenization and compression strategies,
- implementing custom positional embeddings,
- training and testing GPT based models,
- implementing and testing conditional generation,
- co-writing the paper.

4.2 Introduction

Previous work in deep learning with motion-capture data has utilized architectures that are common in natural language processing (NLP), namely sequence-to-sequence

models including RNNs, LSTMs, and transformers. The language model we train and our final tokenization method is generic, meaning that this approach could be implemented with more advanced language models as the field evolves. One of the limitations of this approach is that the quantity of publicly available motion-capture data is significantly less than other modalities, however the size of the AMASS dataset makes a transformer language model a natural choice to learn how to compose this form of language.

Existing transformer-based solutions have focused on processing motion-capture as continuous values, however we have elected to discretize and compress the data. With this representation of motion we are able to predict or generate movement following an existing sequence, fill in missing data in a sequence, prompt motion generation with conditioning tokens, and classify movements.

Our downstream tasks for this chapter include:

- motion generation in Section 4.6 and
- motion classification in Section 4.7.

In Section 4.6 we condition generations on prefixed target motion labels including the dancer performing the movement, the genre, and the song it will be paired with. In addition to these conditioning tokens we prompt outputs on the movement of a select joint or subsets of joints forming gestures, or the motion of a specific dance move. This can indicate the trajectory or style of a complete movement, and we demonstrate that this motion-prompting technique reduces freezing seen in other generative models. We demonstrate that example outputs contain characteristics of the prompted conditioning tokens, and by treating human motion-capture data as a long sequence of tokens in a generative context, we build a system that can prompt the model and provide direction for novel outputs.

In Section 4.7 we describe a method for classifying motions with a supervised linear classifier trained on motion labels. We report our classification results using the same tokens as the conditional generation task, further demonstrating our model's ability to encode these properties. We compare our model to Linear Discriminant Analysis (LDA).

4.3 Background

4.3.1 Pose tokenization

This chapter describes the process of converting motion-capture data to a text-based encoding and training language models to model the encoded data. In this encoding technique joint angles or joint angle related compressions were discretized into tokens that were represented as alphanumeric characters, and frames of motion data were encoded as "words" with a space token separating them. We considered various approaches to encoding motion-capture data as text, including a jpeg inspired DCT-based compression algorithm, PCA and ICA dimensionality reduction, and both uniform and quantile binning.

Our DCT-based approach was inspired by the jpeg [68] algorithm for compressing images. The jpeg algorithm includes steps to divide images into tiles, compute their DCT, and select the most informative DCT components. Our implementation tiles the data over the time dimension, computes the DCT of each joint rotation over each tile, and keeps a percentage of the DCT components based on their magnitude. Our final uniform tokenization process benefits from a fixed maximum error, and provides an end-to-end generic solution for training language models on motion-capture data.

4.3.2 Natural Language Processing

We propose language modeling as an approach to modeling human motion as a vocabulary of movements and poses in sequence because language models have been demonstrated as an effective method for learning representations of text [11] and other modalities [28, 13, 29]. Treating motion as a generic sequence of tokens is in contrast to existing work, which has mostly focused on treating values in the sequence as continuous [64, 38, 2, 1]. The advantage of a discrete sequence is training becomes identical to character level modeling. This approach was inspired by recent work in reinforcement learning treating the task as "one big sequence modeling problem" [29].

The primary deep learning architecture we use in this work is the transformer [66]. We use a decoder-only version implemented with the minGPT [33] and mlm-pytorch [69] frameworks. The transformer architecture has demonstrated improved performance [66]

over previous sequence modeling architectures, especially in natural language processing. Causal models such as the transformer are trained in a self-supervised manner with autoregressive learning [25], where during training the model is assessed on its ability to predict the next token. At test time, the final predicted token can be appended to the original sequence to generate novel outputs.

We discuss pre-training and fine-tuning our model on specific tasks and datasets. The pre-training phase is the process during which the model learns the motion-capture domain, and we hypothesize that a model trained on a larger corpus of kinematics data will have a better understanding of the domain and will translate that information to downstream tasks [11]. The fine-tuning phase is the process of training the model or a classifier on a task specific dataset. In this work we pre-train our models on the AMASS dataset, and fine-tune on the AIST++ dataset for dance related generation and classification tasks.

Our training procedure is inspired by recent work in pre-training and fine-tuning language models for dataset specific applications, and shows similar model behavior as those trained with English text datasets [11]. Namely, pre-training on a large corpus develops an embedding for language that models patterns in words and sentence structure, and fine-tuning identifies patterns specific to the task and dataset. This can be seen in our experiment results, where our model is compared to traditional baselines on downstream classification tasks.

Training our models with causal or masked attention allows us to predict future tokens based on past information or to fill in missing information in motion sequences. The flexibility of the attention mechanism allows us to alter our model to consider the temporal and spatial relationships in motion-capture data, and to make adjustments for downstream generation and classification tasks.

One of the constraints of language models is that the self-attention mechanism has quadratic space and time complexity with respect to the length of the context window. Previous work training transformer architectures on motion-capture data has measured benchmarks up to 400 ms [2], which we anticipated being insufficient for dance and gait-related modeling. Our data compression techniques and custom algorithm described above attempt to reduce the effect of this constraint by creating a compression strategy that would allow for 2 seconds of data in the context window.

Another constraint of character-level language models is the tendency to freeze and repeat the last known character when inferring novel sequences. We use the structure of our encoded text to our advantage by conditioning the model on subsets of joint angles listed in Section A.4, which provide a partial prompt that is consistent and available over time.

4.3.3 Sequence modeling

In order to process SMPL data x, we encode x into a discrete sequence x'. The model f_θ takes x' as input and predicts the sequence y' as output. y' can be decoded to get an SMPL sequence y as the final output, where:

- j is the number of joints
- r is the number of joint rotation dimensions
- t is the number of frames
- $x \in R^{j \times r \times t}$
- $y \in R^{j \times r \times t}$
- $x' = (x_1, \ldots, x_{\mathrm{T}}); T = j \times r \times t$
- $y' = (x_2, \ldots, x_{\mathrm{T}+1}); T = j \times r \times t$

In this work f_θ is a decoder-only transformer, and uniform tokenization is the encoding procedure to convert between x and x' or y and y'.

The transformer architecture contains two mechanisms that we discuss in this section that distinguish it from recurrent neural networks. We experiment with and adapt these functions to our dataset and downstream tasks:

1. An attention mechanism that allows or constrains the flow of information between input and output pairs, and has no internal state.
2. A positional embedding that informs the model of the position of time steps in the sequence. The transformer processes paired inputs and outputs invariant to their ordering so it has to infer or be told positional information.

The attention mechanism which calculates the weighted attention from the query (Q), key (K), and value (V) vectors and the length of the key, d_k is shown in Equation 4.1.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.1}$$

In the context of self-attention we compute weights that indicate the elements in the sequence that are most informative of the current element. We can think of the query as an embedding related to the current time step, and the keys as embeddings related to the other time steps. The attention mechanism computes the dot product of the query and keys and generates logits that identify the correlation between the current element and the rest of the sequence. The magnitude of a dot product scales up with sequence length, so the division by the square-root of the sequence length balances the resulting logits. The softmax function, which is described in Equation 4.2, converts the logits to probabilities which sum to 1.

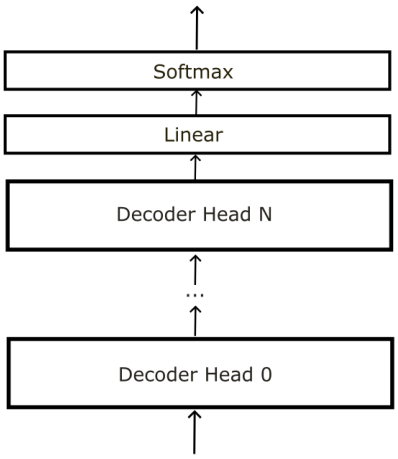
The softmax function is defined for a vector x as:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{4.2}$$

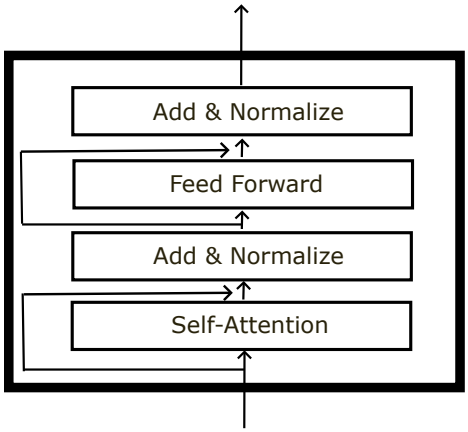
Finally, the probabilities are multiplied by the values associated with each element, resulting in the final weighted attention vector.

Figure 4.1 shows the decoder-only architecture we utilize in this chapter. Figure 4.1a shows the overall architecture of the transformer comprising a series of decoder heads leading to linear and softmax layers. The input to the model, $\boldsymbol{v}_{\text{emb}}$, is the sum of the token and positional embeddings. These embeddings are defined by matrices that encode each vocabulary token or positional token into a vector of length N_{emb}. At the output of the final decoder head, the $\boldsymbol{v}_{\text{emb}}$ is passed through a linear layer to generate logits, and a softmax layer to generate probabilities of each token in the vocabulary at each time step. The final sequence is selected according to these probabilities. In this case we implement a greedy search, however alternate methods such as beam search could be more effective for some sequence modeling problems.

Figure 4.1b shows the interior of a decoder head. The basic building blocks are feed-forward networks (FFNs) and attention blocks which are stacked with residual connections and normalization operations between them. At each block an embedding



(a) Transformer decoder block diagram



(b) Transformer self-attention head block diagram

Figure 4.1: Block diagrams showing the high-level components of a decoder-only transformer. Figure b shows the interior architecture of the decoder heads in Figure a.

vector, $\boldsymbol{v}_{\text{emb}}$, is passed into the decoder head where the self attention block computes the weighted attention vector, $\boldsymbol{v}_{\text{attn}}$. The input and attention vectors, $\boldsymbol{v}_{\text{emb}}$ and $\boldsymbol{v}_{\text{attn}}$ are summed, normalized, and passed through a FFN. The output of the FNN, $\boldsymbol{v}_{\text{ffn}}$ is added to the the normalized sum of $\boldsymbol{v}_{\text{emb}}$ and $\boldsymbol{v}_{\text{attn}}$, and the resulting output vector, $\boldsymbol{v}_{\text{output}}$ is passed to the next decoder head or to the final linear and softmax layers of the transformer.

Previous implementations of generative motion-capture modeling which we aim to build on utilized mean-squared error as an objective function. Mean squared error (MSE) loss is defined in equation 4.3 where y_i represents the true or target value of the ith sample and $\hat{y}_i$ represents the predicted value of the ith sample.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.3}$$

In this case SMPL data is predicted as a float, and does not require tokenization or discretizing preprocessing steps. Our implementation tokenizes the data, and trains the model with a cross-entropy loss function. Cross-entropy loss is described in equation 4.4 where y_{ic} denotes the true label (ground truth) for the ith sample being the cth character in the token vocabulary. y_{ic} takes the value of 1 if the prediction is correct, and 0 otherwise. $\hat{y}_{ic}$ indicates the predicted probability for the ith sample

being character c as generated by the model.

$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic}) \tag{4.4}$$

The main disadvantage of cross-entropy loss is that it does not reflect the distance between tokens in space. Assuming that our tokenization scheme results in bins of 1 degree of rotation, the cross-entropy loss is not directly informed that a prediction with 45 degree error is better than a prediction with 90 degree error. With MSE loss, this relationship is modeled explicitly. The main advantage of cross-entropy loss is that the model is less likely to converge to an average prediction, making it a better candidate for generative outputs.

4.4 Pose tokenization

Our tokenization strategy had two goals: to convert motion-capture data to text so it can be used in a generic language model architecture, and to minimize the sequence length. In order to train a generic language model architecture on motion-capture data we converted the data to a text-based encoding, or a one-dimensional sequence of characters. In each case we represented the frames, or frame-related compressions, as words, and separated them with a space token. We converted the poses in each word to a string of alpha-numeric characters, which were defined by our encoding strategy.

The second goal maximized the information in the context window of the language model by representing motions in as few characters as possible. We aimed to capture a minimum of 2 seconds of movement that could represent the periodicity of movement in dance related activities. To achieve this we investigate methods of compressing motion-capture data in SMPL format with spectral analysis, dimensionality reduction, and generic binning strategies. We consider the efficacy of these techniques in the context of using the text-based encoding in our language model architectures. Figure 4.2 shows an example of the final text encoding with frame and joint index references.

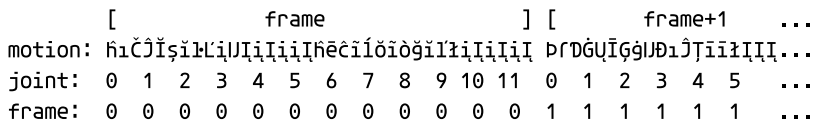


Figure 4.2: Example of motion data as encoded text.

4.4.1 Methods

Each of our tokenization schemes incorporated either uniform or quantile binning techniques to convert poses or pose-compressions to integers which were then matched to a dictionary of alphanumeric characters. Uniform tokenization follows the method described in [29], which we describe in more detail below.

Let x be a continuous variable that we want to quantize, and let $q_1, q_2, ..., q_n$ be the n quantization levels or bins. We assume that the bins are uniformly spaced, so that the distance between adjacent bins is the same and can be denoted as Δ.

Then, the quantization operation $Q(x)$ can be defined as:

$$Q(x) = q_k \quad \text{if} \quad q_{k-1} \leq x < q_k$$

where k is the index of the bin that contains x, and is given by:

$$k = \left\lfloor \frac{x - q_1}{\Delta} \right\rfloor + 1$$

Here, $\lfloor \cdot \rfloor$ denotes the floor function, which rounds down to the nearest integer.

Note that the quantized value $Q(x)$ takes on one of the n possible quantization levels $q_1, q_2, ..., q_n$, depending on which bin x falls into.

In the case of quantile binning, the data is divided into bins based on the quantiles or percentiles of the variable distribution. We start by specifying the number of bins, denoted as n. The variable x is then divided into n intervals or bins such that each bin contains an equal number of observations. This means that each bin represents a specific quantile of the variable's distribution.

In the quantile tokenization method every bin has the same number of samples in it, which makes the bin widths inversely proportional to the number of occurances and makes the probability of each token the same. Uniform tokenization uses a consistent bin width, which has two benefits for language modeling. The first is that the maximum reconstruction error is consistent for each token in the dataset, which

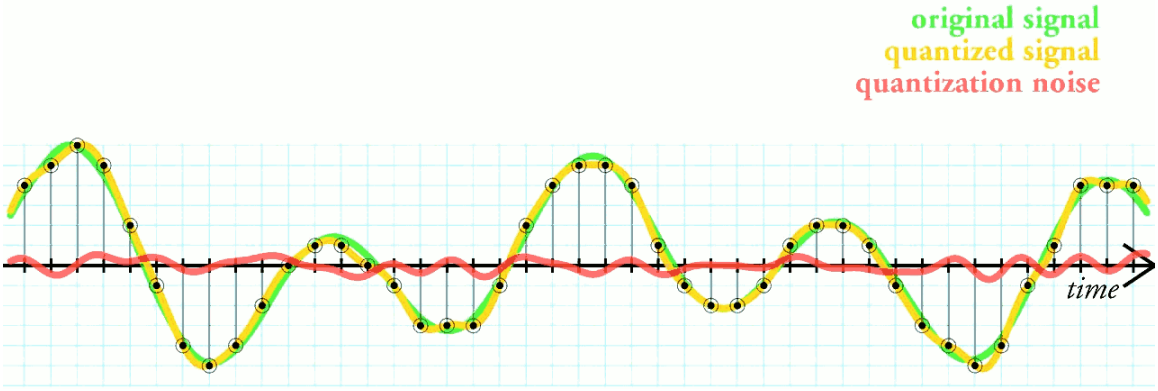
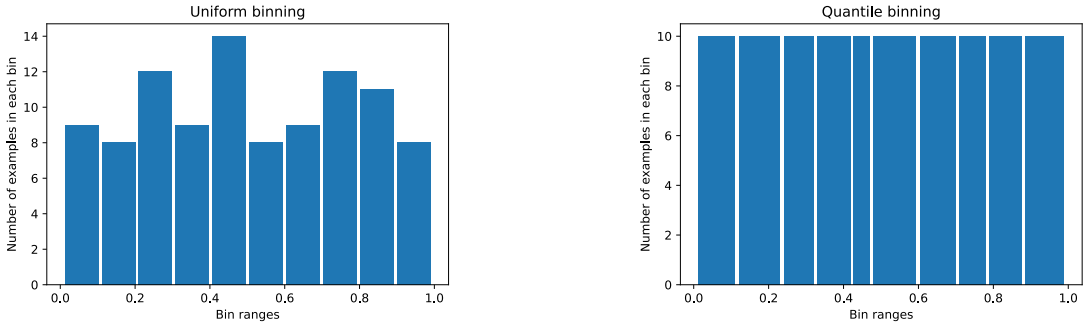


Figure 4.3: Illustration describing the process of quantization.[1]



(a) Uniform binning

(b) Quantile binning

Figure 4.4: Visualizations of uniform and quantile binning strategies. Uniform binning has consistent bin widths resulting in a fixed maximum reconstruction error. Quantile binning has consistent bin frequency with increasing reconstruction error when poses are less common.

means that we can consider a diverse set of movements without representing improbable poses poorly. The second is that the model can learn the probability of a token occurring in human movement more easily, since the tokens that are closer to common postures and gestures will be represented more frequently. This is comparable to the frequency of characters in an alphabet. Quantile and uniform binning are visualized in Figure 4.4.

In Chapter 3 we demonstrated that the DCT captured the low-frequency components of motion in a way that was effective for a motion interpolation task. Building

[1]Attribution: Gregory Maxwell, CC BY 3.0 https://creativecommons.org/licenses/by/3.0, via Wikimedia Commons.

on this result, we design a jpeg-inspired algorithm to compress the data. The components of the jpeg algorithm which we take inspiration from include dividing images into "tiles", which contain 8x8 pixel subsections of the original image, computing the DCT for each color channel across the tile, and storing the DCT components with resolutions proportional to their impact on human perception of the image. These concepts are translated into our procedure by dividing the sequence into "tiles", meaning series of frames spanning up to one second, computing the DCT of each joint rotation over the tile period, and removing the lowest values in the DCT, with the assumption that these have the least impact on the reconstruction error.

In order to convert the DCT components to alphanumeric characters, we first convert the DCT components to integers with either the quantile or uniform binning strategies. We convert the resulting integers to alphanumeric characters in the bin-to-character dictionary, and used run-length encoding on consecutive DCT components that were set to zero. We replaced these components with a single character that represents the value zero and the length of the run. This reduces the storage requirements and word length of our dataset.

The jpeg-inspired DCT-based compression algorithm attempted to compress the data over the time dimension. To compress the data over the joint rotation dimension we also considered PCA and Independent Component Analysis (ICA) (described in Section A.2). While the DCT-based compression uses run-length encoding to compress the data, PCA or ICA offer a consistent way to reduce the word length.

4.4.2 Results

We report quantitative metrics for our tokenization procedures, however qualitative evaluation equally informed our design choices. We provide demonstrations of reconstructed motions as part of our qualitative analysis, recognizing that for a generative model the output is a visual rendering of movement. The quantitative metrics that we consider in this section include the average or maximum word length, compression factor, duration of motion sequences in the context window, and average or maximum reconstruction error.

Word length refers to the number of characters that make up a word, in this case words are either frames or jpeg-inspired tiles. The fixed word length for a generic

tokenization strategy (uniform or quantile) is shown in equation 4.5 where:

$$\text{Word Length}_{\text{uniform}} = N_{\text{joints}} \times N_{\text{rotations.}} \tag{4.5}$$

The word length for the jpeg-based algorithm has a fixed dependency on the dimensionality reduction on the joint rotation dimension, and a variable dependency on the efficacy of the run-length encoding of the DCT components. The word length for this encoding strategy is shown by equation 4.6 where $D_{\text{joint rotations}}$ is the number of components in the joint rotation after dimensionality reduction, L_{tile} represents the frame length of the DCT tile, and R_{DCT} represents the ratio of DCT components that are kept after run-length encoding.

$$\text{Word Length}_{\text{DCT}} = D_{\text{joint rotations}} \times L_{\text{tile}} \times R_{\text{DCT}} \tag{4.6}$$

The maximum word length, assuming no run-length coding is utilized, would be the same equation with $R_{\text{DCT}} = 1$.

Compression factor refers to the ratio of bytes required to represent the encoded sequence compared to the original sequence. We report this by comparing the number of bytes in the SMPL representation with the number of characters required to represent a motion and the number of bytes required to represent the alphabet. The duration of the sequence in the context window measures how many seconds of the original movement fits within a 1024 token block size after the data simplifications and compressions. The block size is one of the main constraints of language models and limits the "memory" capacity of the model. Ours is selected based on compute resources.

Finally, the reconstruction error is the sum of the error of each joint in the skeleton. This can be reported as geodesic error or rotation matrix error for joint rotations, or as positional error if forward kinematics is applied to the original and reconstructed joint rotations. In this section we refer to geodesic error as shown in equation 4.7, which computes the shortest path between rotations $\mathbf{a}$ and $\mathbf{b}$.

$$\text{Error}_{\text{geodesic}} = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right) \tag{4.7}$$

Table 4.1: Comparison of the jpeg-based algorithm tile length with PCA and ICA encoding techniques. All examples were computed with 10% of the DCT components retained in the encoding and 24 PCA/ICA components over the 72 joint dimensions. The final values were represented by 360 quantile bins converted to alphanumeric characters.

DCT tile length	Joint compression	Avg Word Length ↓	Compression Factor ↑	Duration (s) ↑	Reconstruction Error ↓
15	ICA	45.5	132	5.63	0.0925
30	ICA	86.5	144	5.92	0.0901
60	ICA	169	152	6.06	0.0890
15	PCA	43.5	126	5.88	0.0904
30	PCA	84.8	141	6.04	0.0875
60	**PCA**	166	153	**6.17**	**0.0872**

Our jpeg-inspired compression algorithm was the most promising solution for maximizing the duration of the motion sequence in the context window. Table 4.1 compares the effects of PCA and ICA dimensionality reductions and the DCT tile length on the average word length, compression factor, duration, and the average reconstruction error. The results demonstrate that PCA performs slightly better than ICA, and that increasing the tile length decreases the reconstruction error and increases the duration in the context window. We expect that for the language model to learn the composition of sequences it would require at least 3 words in the context window, meaning that the upper limit of the tile length is bound by the maximum word length and the context window. In our case this limits the maximum word length to 341 characters.

Table 4.2 shows the effect of increasing the number of DCT components or PCA components included in the text encoding. This table demonstrates the relationship between compression and reconstruction error of the encoding algorithm, and Demo Video 7 (https://youtu.be/JME9vbxtH7Q) demonstrates qualitative examples of the encoding techniques with low and high reconstruction error. The figure on the left (low error) encodes 60 frames in the DCT, retains 40% of the coefficients, and includes 15 PCA components. The figure on the right (high error) encodes 60 frames in the DCT, retains 10% of the coefficients, and includes 42 PCA components.

From the results in Table 4.2 we demonstrate that increasing the number of PCA components has a greater impact on reconstruction error than the number of DCT components. We note the best result with 10% of the DCT components and 42 PCA components where the mean reconstruction error reaches 0.0476 cm with word length 307. Demo Video 8 (https://youtu.be/UIasLJlegbc) shows qualitative results

Table 4.2: Scaling the PCA and DCT components included in the text encoding. All examples were computed with 60 frames per tile and the final values were represented by 360 quantile bins converted to alphanumeric characters.

DCT components (%)	PCA components	Avg Word Length ↓	Compression Factor ↑	Duration (s) ↑	Mean Reconstruction Error ↓
10	15	102	257	10.04	0.129
10	24	170	150	6.02	0.0886
10	33	239	108	4.28	0.0577
10	**42**	307	85.5	3.33	**0.0476**
20	15	186	144	5.50	0.125
30	15	201	133	5.09	0.125
40	15	186	144	5.50	0.125

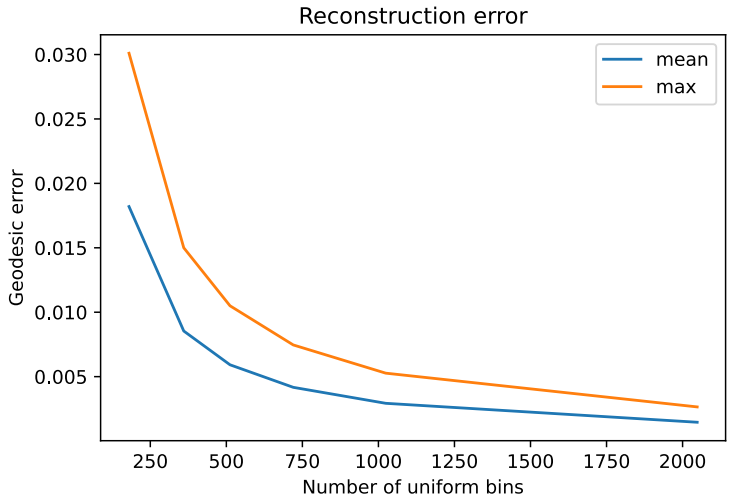


Figure 4.5: Mean and maximum geodesic error with respect to the number of uniform bins. The inflection point in the graph informed our decision to tokenize the data with 360 uniform bins.

of this encoding strategy for 3 examples of motion. We see that while the effects of the tokenization algorithm are visually undetectable during each tile, there are discontinuities at the boundaries that create a visible disturbance in the sequence. These are visible in Demo Video 8 at 1 Hz frequency. This result also indicates that increasing the tile length decreases the number of discontinuities, resulting in the inverse relationship between tile length and reconstruction error shown in table 4.1.

Based on these results our final tokenization strategy in Section 4.5 employs uniform tokenization. Figure 4.5 shows reconstruction error with respect to the number of bins in the uniform tokenization. We selected 360 bins where the maximum quantized error was 1 degree of rotation for each dimension of the axis-angle vector.

4.5 Language modeling

After tokenizing the AMASS and AIST++ datasets we train the generic minGPT [33] and mlm-pytorch [69] frameworks on our combined dataset with no alterations. Both frameworks are decoder-only transformer architectures based on GPT2, which contains 12 attention heads, 12 layers, and 768-dimensional embeddings. The flexibility of the attention mechanism allows us to alter our model for conditional generation and classification tasks during the fine-tuning phase of training, and to test both causal and masked attention mechanisms.

In this work we pre-train the models on the AMASS and AIST++ datasets, and fine-tune the model on the AIST++ dataset for dance generation and motion classification. Pre-training refers to a training regime that precedes the "fine-tuning" training on a task of interest. The goal is that the model may perform downstream motion modeling tasks more effectively after learning to model generic motion accurately. Our model's performance on an autoregressive generation benchmark is described in Section 4.5.3, however as this is a generative model quantitative benchmarks will always be lacking in describing the performance of the model [62]. We demonstrate and assess our generative outputs qualitatively with demo videos.

In Section 4.5.3 we discuss the results of training both causal and masked models, compare the results of pre-training and training from scratch, the effect of simplifying the dataset, and the effect of a custom positional embedding based on the structure of our encoded data. Finally, we describe our methods and results for conditional generation and classification tasks in Sections 4.6 and 4.7.

4.5.1 Datasets

Training on a large dataset of tokens allows for a more flexible model that can learn useful representations. This approach to representation learning is core to deep learning [23, p.524]. However, there is a tradeoff in the inductive priors an architecture might assert versus the effect of data. For example, vision transformers are typically described as more generic than convolutional networks, and they are able to learn more useful representations, but this effect was only seen on larger datasets [14].

In Table 4.3 we investigate the relative information content of different datasets

Table 4.3: Comparison of relative information content of datasets. Size is reported in bits per token for generative models trained on each dataset. The reported bits/frame was trained on all joints rather than the subset used elsewhere in this paper.

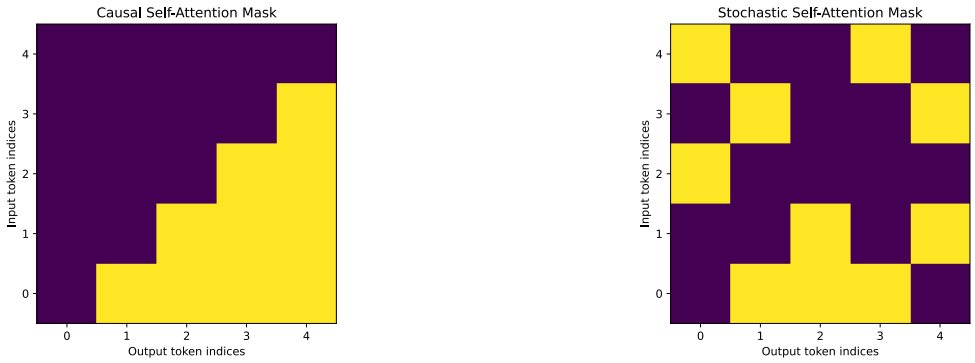
Name	Description	Size (bits)
ImageNet	Image Database	179G (3.57 bits/pixel) [65]
The Pile	Text Database	837G (2.45 bits/token) [7]
AMASS	Motion Database	0.76G (88.6 bits/frame)

used in deep learning. The information content here is quantified by looking at the potential compressed size using optimal entropy coding under a competitive generative model of the data, including our own. The result suggests that pre-training on AMASS may be limited by the dataset size, and further that the motion capture data publicly available may be insufficient to train generative models that are as powerful as those operating in other modalities.

In this work we consider a bilateral data augmentation to accomodate the relative dataset size. We mirror the data through a matrix multiplication that inverts the direction of the appropriate joint rotations to perform motions on the left and right side of the body. Details of this operation can be found in Section A.5. We list a number of other data augmentations that could be applied to motion-capture data in Section 5.1.

4.5.2 Methods

Causal modeling typically refers to the practice of training a model to autoregressively predict the next token [25]. At test time, the final predicted token can be appended to the original sequence iteratively to generate novel outputs. Masked modeling typically refers to training a model to fill in a sequence where a portion of the tokens have been randomly masked. In our model, we randomly masked 15% of tokens in the training sequence and allowed the model to attend to the remaining 85%. Our causal model learns autoregressive joint angle prediction for motion prediction and generation tasks, and our masked model learns to fill in sequences where there could be a joint occluded or some of the data could be missing. Figure 4.6 shows causal and stochastic attention mechanisms for the Causal Language Model (CLM) and Masked Language Model (MLM).



(a) Causal Self-attention

(b) Stochastic Self-attention

Figure 4.6: Visualizing the attention masks of the CLM and MLMs. Light areas are time steps that the model can attend to.

To simplify the modeling task, we reduced our frame rate to 20 seconds and we only include 13 of the original 24 joints. A complete list of these joints can be found in the appendix A.4 and Figure 4.7 demonstrates these joints on the SMPL skeleton.

Our initial tests are run on models with 26 million parameters based on the estimated parameter requirements [32] for our 285M token dataset. The models were pre-trained for 7500 iterations on the AMASS dataset processed with the data splits defined by Rempe et al. [54], and the AIST++ [39] dataset.

Inspired by spatio-temporal transformers for motion-capture data [1] we implemented a custom positional embedding that informed the attention mechanism of the frame and joint rotation index. Each character in the sequence was mapped to a frame and joint index, which mapped to two positional embeddings that were added to the token embedding. We summed them together the same way a traditional absolute position embedding is used in a transformer [66]. Figure 4.8 demonstrates the frame and joint indices for a motion sequence.

After pre-training the models, we fine-tuned them on the AIST++ dataset. The fine-tuning process reduced the bits/frame result further than training from scratch, and incurred an absolute quantization error that was not visually noticeable.

4.5.3 Results and discussion

We train our models with the batch size set to 128, the learning rate set to 2E-4, and embedding, attention, and residual dropout set to 0.1. We tested various batch

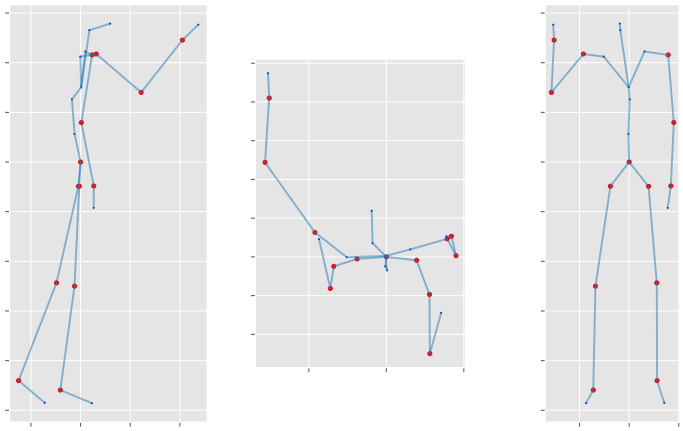
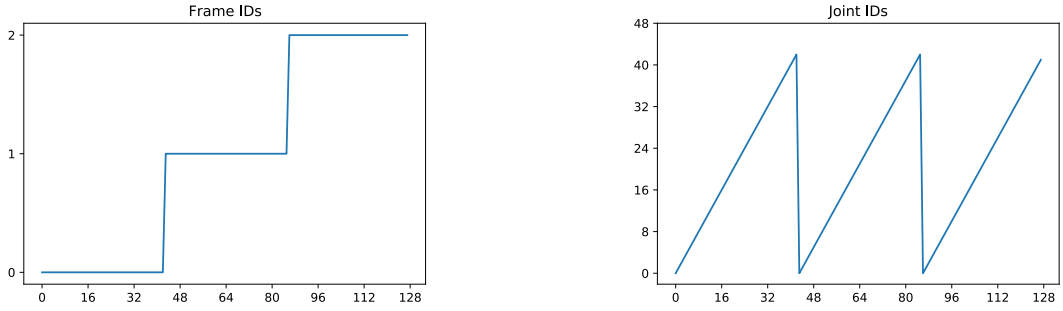


Figure 4.7: 13 of the 24 available joints were modeled.



(a) Frame index.

(b) Joint index.

Figure 4.8: Visualizing the frame and joint indices of a motion sequence with 43 tokens per word.

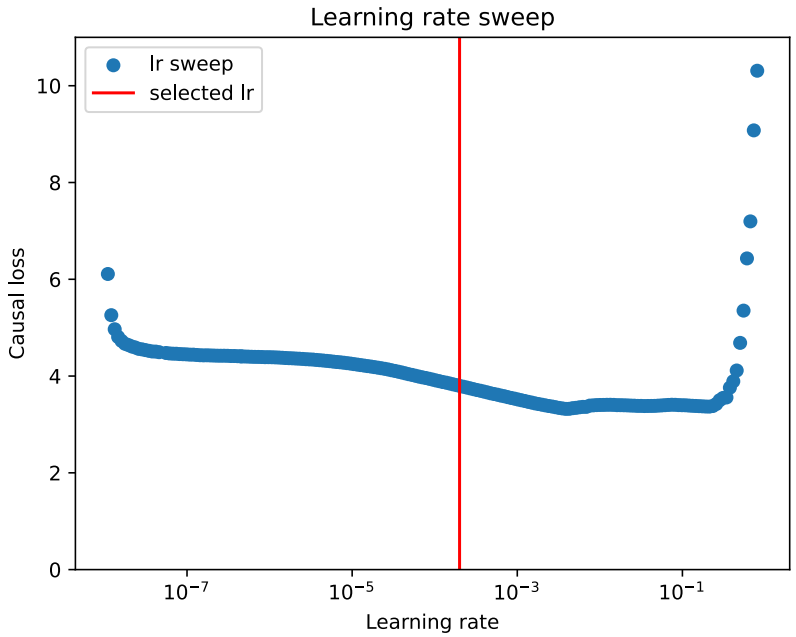


Figure 4.9: The selected learning rate is 10 times less than the inflection point where the learning rate causes an increase in the causal loss.

sizes with negligible changes in performance, and focused on tuning the learning rate. We ran a learning rate sweep shown in Figure 4.9 with our encoded data. This technique [31] plots the loss with respect to the learning rate over the plausible range of values, and shows where the learning rate passes the optimal range. The suggested value is 10 times less than the inflection point on the graph, which was 2E-4 in our case.

Table 4.4 compares the results of training the causal and masked models with the standard absolute positional embedding and our custom positional embedding. The loss curves of the causal model showed that the custom embedding helps the model learn faster, however the models converged to the same loss. The masked model improved significantly from the custom embedding and reached a lower loss and bits/frame rate. The final causal and masked losses are not directly comparable, and we found that the causal model performed better at generative tasks which was our primary goal. We continued our experiments with the causal model due to the nature of our downstream tasks.

Table 4.5 shows the effect of dataset simplifications on model performance. Here

	Causal		Masked	
Embedding	bits/frame	loss	bits/frame	loss
Absolute	106	1.75	256	4.24
Custom	104	1.72	**90**	**1.49**

Table 4.4: Comparison of causal and masked model performance with absolute positional embeddings and our custom positional embedding.

Table 4.5: Performance after reducing the frame rate and removing joints. Note that the causal loss, or per character loss, is lowest when the complexity of the dataset is lowest.

Data simplifications	Causal bits/frame	Causal loss
13 joints, 20 fps	**104**	**1.85**
24 joints, 20 fps	273	2.63
13 joints, 60 fps	151	2.69
24 joints, 60 fps	199	1.92

we train the causal model from scratch on the AIST++ dataset with frame rate and joint reductions. The results show that both simplifications have a significant impact on the final performance.

We demonstrate the effect of pre-training and fine-tuning in Table 4.6. We show the model performance after pre-training on the AMASS and AIST++ datasets and fine-tuning on the AIST++ dataset, and compare it to training the model from scratch. We test two learning rate options for the fine-tuning stage, and find that decreasing the learning rate did not improve the final loss. The table shows that pre-training and fine-tuning the model improved performance, and the qualitative results demonstrated that pre-training made the output movements feasible.

Table 4.6: Comparison of pre-training on AMASS and the AIST++ dataset with and without fine-tuning and training the model from scratch on the AIST++ dataset. Note that the loss after fine-tuning is significantly lower than training from scratch.

Training regime	Causal bits/frame	Causal loss
Training from scratch	100	1.66
Pre-training	104	1.72
Fine-tuning (lr=2e-4)	**86**	**1.42**
Fine-tuning (lr=1e-4)	86	1.42

Table 4.7: *AMASS results* comparing to the work of Aksan et. al. [1] and the results used for comparison in their paper in the style presented in their paper. ↓ indicates metrics where lower is better and ↑ indicates metrices where higher is better. * indicates the model was evaluated by Aksan et. al. [1] rather than the original authors. Our model is referred to here as *LM*.

	Euler ↓				Joint Angle ↓				Positional ↓				PCK (AUC) ↑			
milliseconds	100	200	300	400	100	200	300	400	100	200	300	400	100	200	300	400
Zero-Velocity [45, 2]	1.91	5.93	11.36	17.78	0.37	1.22	2.44	3.94	0.14	0.48	0.96	1.54	0.86	0.83	0.84	0.82
Seq2seq [45, 2]	2.01	5.99	11.22	17.33	0.37	1.17	2.27	3.59	0.14	0.45	0.88	1.39	0.86	0.84	0.85	0.83
QuaterNet [51, 2]	1.49	4.70	9.16	14.54	0.26	0.89	1.83	3.00	0.10	0.34	0.71	1.18	0.90	0.87	0.88	0.85
DCT-GCN (ST)* [44]	1.23	4.00	8.05	13.04	0.24	0.77	1.60	2.66	0.09	0.31	0.63	1.06	0.92	0.89	0.89	0.87
DCT-GCN (LT)* [44]	1.27	4.18	8.37	13.38	0.24	0.80	1.65	2.71	0.09	0.31	0.65	1.07	0.91	0.89	0.89	0.87
RNN-SPL [2]	1.33	4.13	8.03	12.84	0.22	0.73	1.51	2.51	0.08	0.28	0.57	0.96	0.93	0.90	0.90	0.88
Transformer	1.30	4.01	7.88	12.69	0.22	0.73	1.52	2.54	0.08	0.28	0.58	0.97	0.92	0.90	0.90	0.88
ST-Transformer	1.11	3.61	7.31	12.04	0.20	0.68	1.45	2.48	0.08	0.27	0.57	0.97	0.93	0.90	0.90	0.88
LM	1.01	3.28	5.97	8.78	0.097	0.295	0.520	0.761	0.200	0.562	0.943	1.32	0.505	0.399	0.350	0.354

Table 4.7 shows our best model performance compared to existing baselines. The metrics listed reflect the ways transformer models for motion modeling are compared in the literature. These models are typically trained on continuous 3D joint angles, which can be represented using at least six formalisms. Errors between angles may be similarly computed in various ways. For example, geodesic error is the size of the minimum rotation in radians to rotate from one angular orientation to another. In Table 4.7 this metric is called *Joint Angle*. The remaining metrics in Table 4.7 are described by Aksan et. al. [2]. Briefly:

- *Euler* is the RMSE between the joint angles expressed as Euler angles
- Two are defined over positions computed from the predicted joint angles using a predefined forward kinematics model:
  - *Positional* is the MSE between positions in 3D space
  - *PCK (AUC)* is ratio of joints within a spherical threshold around the target position in 3D space

Our model tends to outperform baselines on joint rotation related metrics, and perform worse than the baseline in 3D joint position related metrics. This is likely due to the lack of 3D joint position data in our model, and we discuss a training procedure that could improve positional error in Section 5.1.

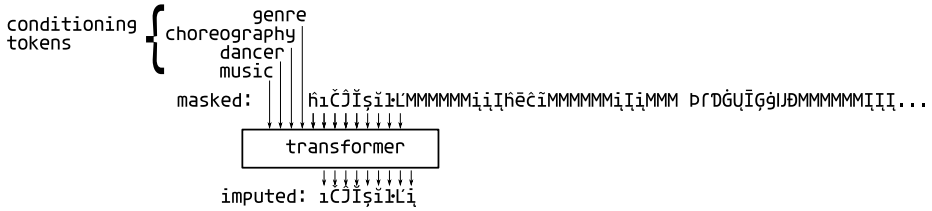


Figure 4.10: Illustration of how the transformer operates on an example of text with conditioning tokens prepended to the sequence. Masked tokens are denoted with "M", the causal model moves from left to right inferring masked tokens.

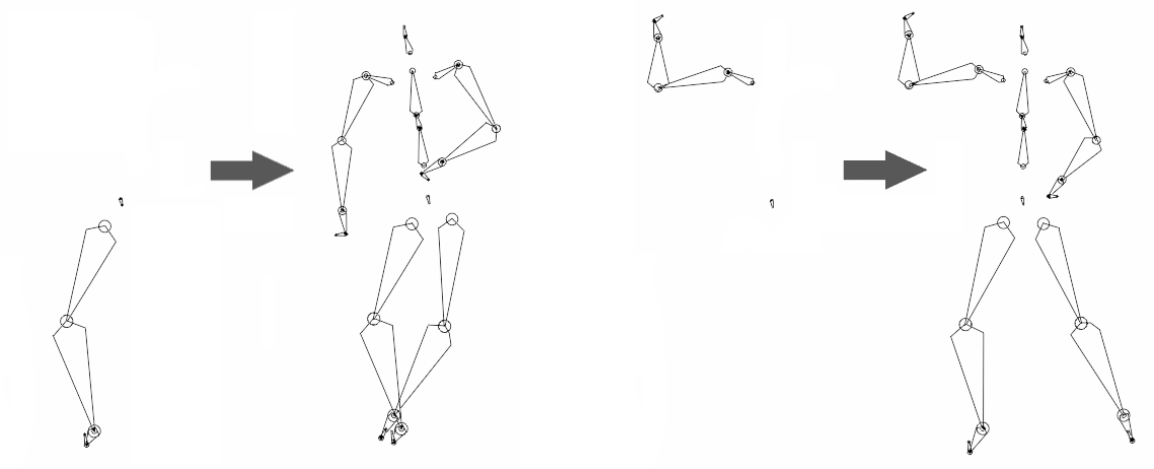
4.6 Conditional generation

Fine-tuning on the AIST++ dataset resulted in a model that outputs novel dance movement. In order to control the output, we use select joints as input and fine-tuned the model with additional conditioning tokens at the beginning of the sequence as shown in Figure 4.10. The conditioning tokens indicate the dancer, dance genre, music, and choreography that define each example in the database.

We demonstrate examples where a partial skeleton is used as a prompt - including the root joint and the 3 joints defining an arm or leg. Figure 4.11 demonstrates an example of how the input is selected. The remaining joints were replaced with mask tokens, and the causal model iteratively filled in the masked tokens while attending to the existing motion sequence.

The conditioning tokens were taken from the AIST++ filenames, which include a selection of 30 dancers, 10 genres, and 60 music files. Each dance genre had at least 10 basic choreography options, which were repeated examples of basic movements. We included a token indicating whether the movement came from this basic movement vocabulary or whether it was an advanced movement selected by the dance artist, and primarily used the advanced movement prompt for generating outputs.

Demo Video 9 (https://youtu.be/RXv89p7443g) shows an example of the model inputs (left) compared to the original or context motion (right) from the AIST++ dataset. Demo Video 10 (https://youtu.be/z8I2xkqaZ6U) shows the effect of conditioning tokens on rendered examples, where the example on the left shows the context motion, the middle example shows a motion generated from the root orientation, translation, and right arm joints of the context motion, the "waack" genre token, dancer ID 26, and song "mWA2", and the example on the right shows a motion



(a) Right leg as the context motion.

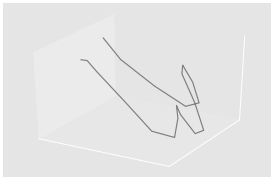
(b) Right arm as the context motion.

Figure 4.11: Illustration of the joints used as context for the model, and the generated output. The video of motion b can be found in Demo Video 10.

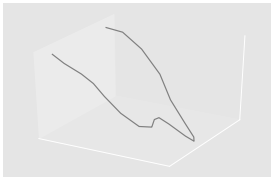
generated from the same joints, the "krump" genre token, dancer ID 29, and song "mKR2".

We can see that conditioning tokens promote diversity in the generated output, and promote common movements from the target conditions. For example, movements conditioned on "waack" tokens demonstrated more circular arm movements, movements conditioned on "krump" tokens demonstrated more downward movements with the arms and legs, and movements conditioned on "house" tokens demonstrated more bending in the knees, leading to more frequent changes of weight placement. Figure 4.12 demonstrates an example of how the conditioning tokens affect the trajectory of a gesture. It shows (a) the 3D position of the right wrist in the context motion from the krump genre, (b) the 3D position of the left wrist in the generated motion conditioned on the waack genre, and (c) the 3D position of the right wrist in a waack motion sampled from the dataset. These plots attempt to demonstrate the model's ability to generate movements that follow the symmetrical patterns found in dance movement, and how conditioning tokens affect the generated motion.

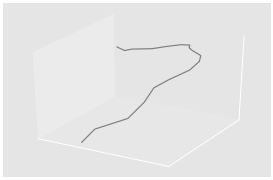
Conditioning dance generation on a select set of joints allows us to generate long dance sequences that maintain a certain coherence (via the user-controlled joints) despite the limitation of the available model time window. By further conditioning the output on individual dancers, genres, or musical properties, choreographers can



(a) 3D position of the right wrist during the context motion.



(b) 3D position of the left wrist during the generated motion, conditioned on waack tokens.



(c) 3D position of the right wrist during a waack motion.

Figure 4.12: Visualizing the effect of conditioning tokens on joint trajectories in 3D space.

explore how specific dancers or dancers with specific dance backgrounds might adapt to their gestures and cues, or generate motion for tasks they are less familiar with.

The success of this style of modeling opens up a vast array of approaches from related fields surrounding language modeling. Long-range models could increase the available time window [61, 28]. Prompt tuning would allow distillation of a dancer's style to a vector [36, 21], and any modality that may be paired with motion may be contrastively learned [52].

We utilized outputs from this model in rehearsal and live performance where dancers learned and performed generated movements. Example movements that were used in this performance are shown in Demo Video 11 (https://youtu.be/yRtEKLmzeH4). In this work, the dancers and choreographers used and interacted with both the conditional generation and motion interpolation tools. Interpolated motions tended to be more effective as creative prompts for short movements, while outputs from the conditional generation tool were able to generate longer sequences of novel choreography. One of the constraints of working with both tools, especially the conditional generation tool, was that the generated movements were not always danceable or feasible to perform. Another constraint was that the timing of the movements changed speed and offset over time. When the joint-based prompts contained information that could be related to beats or timing, the generative model would often coordinate the rest of the joints with the movement. If the prompt was stationary or did not contain movement that related to the musicality of the overall movement, the model often generated off-beat or awkward timing. The choreographers and dancers had to adapt

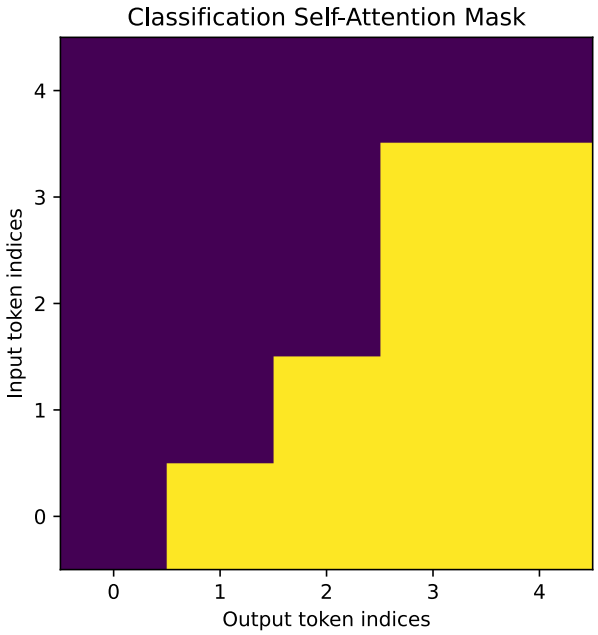


Figure 4.13: Visualizing the attention mask of the classifier. The motion sequence has a causal mask as before, but the classification tokens (the last 2 indices) can attend to any token in the context window.

the model outputs based on these constraints, however the choreographer noted that the creative prompts reduced choreography ideation and rehearsal time.

4.7 Motion classification

Motion classification is one of the most common downstream tasks for human motion modeling. We aim to classify dance data by the same dancer, genre, and choreography tokens that we used in our conditional generation task. In order to classify movements, we assign the final tokens in the context window as multi-class logits, and train a classifier head with a single linear layer mapping to the multi-class labels with a cross-entropy loss function. In this case the causal mask makes the classification task more difficult for earlier class predictions, so the causal mask is removed for predicting classification logits. The attention mask for this operation is shown in Figure 4.13.

In order to classify the smaller AIST++ dataset, we use our pre-trained causal model that was trained on the AMASS and AIST++ datasets. We take inspiration

Table 4.8: Performance of our classification model on the AIST++ dataset. The majority class prediction is the baseline without either model. Results show the accuracy as a percentage.

Method	Dancer	Situation	Genre
MCP Baseline	5.32	58.6	10.5
LDA	51.8	76.0	54.4
Our model	**80.7**	**97.1**	**89.1**

from BERT [11] in our fine-tuning procedure by decreasing the learning rate to 2e-5 and decreasing the batch size to 16. We fine-tune the transformer weights for the autoregressive joint angle prediction task jointly with the linear classifier on the target datasets, and report the classification accuracy and compare our results to LDA.

Table 4.8 shows the classification results on the AIST++ dataset for the dancer, genre, and situation. The situation refers to whether the movement comes from the collection of basic or advanced movements. The classifier outperforms the majority class prediction (MCP) and LDA baselines.

4.8 Summary

This chapter defines a training procedure for SMPL motion-capture data with generic language model architectures. We explore motion tokenization strategies comprising a jpeg-inspired algorithm and uniform tokenization of the data. The jpeg-inspired algorithm utilized data compression techniques that were successful in Chapter 3, however the final result showed visible discontinuities from tiling the time dimension. Our final tokenization scheme benefits from being generic, and we believe it could be utilized with other formats of motion-capture data and can be improved on as data tokenization advances.

Our language model training experiments show that pre-training was an effective way to improve performance on downstream tasks, and that simplifying the dataset and adjusting the positional embeddings to the data resulted in the model learning faster and converging to lower loss in most cases. We show that fine-tuning a causal model on dance data with conditioning tokens informing the model of global movement patterns was an effective way to add control over the outputs, and that adding

subsets of joints as a prompt offered further control and prompted more cohesive outputs. Finally, we found that the model was able to effectively classify labels in the AIST++ dataset and beat the LDA baseline. We offer suggested improvements for future work in Section 5.1.

Chapter 5

Conclusion

This work provides an exploration of learning human movement patterns from data using machine learning and deep learning procedures. We explore representations of motion that can effectively capture and compress motion-capture data, develop tools to generate novel movement, and learn global patterns in the data. We propose that common NLP techniques can be applied to motion-capture data when the data is discretized, and with minimal structural changes to the model architecture. We believe that this work and other work like it provides a basis for training LLMs on motion-capture data, and that our ability to model this modality will improve with the NLP field.

In Chapter 3 we develop techniques for motion query and interpolation. The query tool uses the cosine similarity between STFT spectrograms of motions from the AIST++ dataset to identify motions that are similar, and assigns a similarity score between motions. Expert user feedback indicates that this tool would be effective in educational settings to compare student performance with each other and to more advanced dancers, and in professional settings where choreographers have to search for and compare movements from past performance or rehearsal footage. The interpolation tool uses the DCT to interpolate low-frequency components of movement, which capture a simplified version of movement and its choreography. We demonstrate that the DCT captures the movement better than the STFT or PCA, and ask experts to comment on the outputs. Experts found that while motion interpolation and digital tools in general could be challenging to use in an educational setting, performance experts and choreographers were able to identify specific cases where they could use the tool for creative prompts.

Chapter 4 focuses on the technical process of training language models on motion-capture data. We explore discretization methods for SMPL formatted motion-capture

data, and report experimental results of training a decoder-only transformer architecture on the data with cross-entropy loss. Our training procedure shows that pre-training on a large corpus of motion-capture data was an effective way of learning the motion-capture domain for downstream tasks. In our generative dance work we fine-tuned the model on the AIST++ dataset for both classification and conditional generation based on genre, dancer, and choreography labels. We describe a scenario where we implemented the model in a rehearsal setting, and note the challenges and main contributions of the model to the creative process. We aim to provide context in both machine learning research and practical dance application, enabling exciting new directions in both understanding and composing dance with the help of machine learning.

5.1 Future work

Future work for this thesis includes improving the motion-capture data tokenization scheme, and improving the language modeling capacity. For our tokenization scheme we believe that wavelet transforms [20] may be an effective way to compress motion-capture data that would be a non-real time transform that could be expanded based on the context window. PoseGPT [41] used a variational auto-encoder to convert poses to a discrete latent space, which we believe is a more promising direction for this work. The constraints of language models are being addressed in NLP research, and faster attention mechanisms and long-form transformers are promising directions to allow us to increase the context window of our models and potentially make data compression schemes like the jpeg-inspired algorithm more feasible.

The largest constraint that motion-capture modeling faces is the amount of publicly available data, and we believe that the performance of our model will scale with dataset size. To accommodate this issue with the currently available data, we suggest the following data augmentations that could potentially improve training results:

- Increasing/decreasing the frame rate of the motions. This could help the model to understand the speed and resolution of movements.

- Selecting different subsets of joints. This could demonstrate that movements can be performed with just the upper or lower body, and could help the model

to learn the dependencies between joints.

- Rotating the starting orientation of the root joint on the horizontal plane. This would help the model to map out 3-dimensional space without over-fitting to the directions individuals were facing when the movement was recorded.

In addition to these data augmentations, adding tasks to the pre-training procedure may also improve the model's understanding of the motion-capture domain. Training the language model on the SMPL joint angles and mesh parameters, the 3D positions resulting from forward kinematics, and ground contacts that can be recovered from models like HuMoR [54] are example tasks that may improve performance. This idea reflects the T5 [53] paper, and depends on the assumption that advances in data and training augmentations in NLP may also apply to motion-capture data.

Bibliography

[1] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. Attention, please: A spatio-temporal transformer for 3D human motion prediction. *CoRR*, abs/2004.08692, 2020.

[2] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured prediction helps 3D human motion modelling. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. First two authors contributed equally.

[3] Sarah Fdili Alaoui, Kristin Carlson, and Thecla Schiphorst. Choreography as mediated through compositional tools for movement: Constructing a historical perspective. In *Proceedings of the 2014 International Workshop on Movement and Computing*, MOCO '14, page 1–6, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Yoav Bergner, Shiri Mund, Ofer Chen, and Willie Payne. First steps in dance data science: Educational design. In *Proceedings of the 6th International Conference on Movement and Computing*, pages 1–8, 2019.

[5] Alexander Berman and Valencia James. Towards a live dance improvisation between an avatar and a human dancer. In *Proceedings of the 2014 International Workshop on Movement and Computing*, MOCO '14, page 162–165, New York, NY, USA, 2014. Association for Computing Machinery.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[7] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022.

[8] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 183–192, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[9] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019.

[10] Kristin Carlson, Thecla Schiphorst, and Philippe Pasquier. Scuddle: Generating movement catalysts for computer-aided choreography. pages 123–128, 01 2011.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Ran Dong, Yangfei Lin, Qiong Chang, Junpei Zhong, Dongsheng Cai, and Soichiro Ikuno. Motion feature extraction and stylization for character animation using hilbert-huang transform. In *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*, ACM ICEA '21, page 16–21, New York, NY, USA, 2022. Association for Computing Machinery.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[15] Xiaoxiao Du, Ram Vasudevan, and Matthew Johnson-Roberson. Bio-LSTM: A biomechanically inspired recurrent neural network for 3-d pedestrian pose and gait prediction. *IEEE Robotics and Automation Letters*, 4(2):1501–1508, apr 2019.

[16] Katerina El Raheb, Theofilos Mailis, Vladislav Ryzhikov, Nicolas Papapetrou, and Yannis Ioannidis. Balonse: Temporal aspects of dance movement and its ontological representation. pages 49–64. Springer International Publishing, 2017.

[17] Katerina El Raheb, Nicolas Papapetrou, Vivi Katifori, and Yannis Ioannidis. Balonse: Ballet ontology for annotating and searching video performances. pages 1–8, 07 2016.

[18] Benjamin Filtjens, Pieter Ginis, Alice Nieuwboer, Peter Slaets, and Bart Vanrumste. Automated freezing of gait assessment with marker-based motion capture and multi-stage spatial-temporal graph convolutional neural networks. *Journal of NeuroEngineering and Rehabilitation*, 19(1), may 2022.

[19] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics, 2015.

[20] Jules Françoise, Gabriel Meseguer-Brocal, and Frédéric Bevilacqua. Movement analysis and decomposition with the continuous wavelet transform. In *Proceedings of the 8th International Conference on Movement and Computing*, MOCO '22, New York, NY, USA, 2022. Association for Computing Machinery.

[21] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

[22] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[24] George E Gorton III, David A Hebert, and Mary E Gannotti. Assessment of the kinematic variability among 12 motion analysis laboratories. *Gait & posture*, 29(3):398–402, 2009.

[25] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. 2013.

[26] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José M. F. Moura. Adversarial geometry-aware human motion prediction. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 823–842, Cham, 2018. Springer International Publishing.

[27] Yan Guo, Gang Xu, and Saburo Tsuji. Understanding human motion patterns. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 325–329. IEEE, 1994.

[28] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021.

[29] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as p.524one big sequence modeling problem, 2021.

[30] I.T. Jolliffe. Definition and derivation of principal components. In *Principal Component Analysis*, pages 1–6. Springer, 1986.

[31] Jeremy Jordan. Setting the learning rate of your neural network. https://www.jeremyjordan.me/nn-learning-rate/.

[32] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[33] Andrej Karpathy/karpathy. mingpt. https://github.com/karpathy/minGPT.

[34] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation, 2020.

[35] François-Joseph Lapointe and Martine Époque. The dancing genome project: Generation of a human-computer choreography using a genetic algorithm. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, page 555–558, New York, NY, USA, 2005. Association for Computing Machinery.

[36] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[37] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics, 2018.

[38] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. AI choreographer: Music conditioned 3D dance generation with aist++, 2021.

[39] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3D dance generation, 2021.

[40] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[41] Thomas Lucas*, Fabien Baradel*, Philippe Weinzaepfel, and Grégory Rogez. Posegpt: Quantization-based 3D human motion generation and forecasting. In *European Conference on Computer Vision (ECCV)*, 2022.

[42] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.

[43] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: archive of motion capture as surface shapes. *CoRR*, abs/1904.03278, 2019.

[44] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

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

[46] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

[47] Meinard Muller, Andreas Baak, and Hans-Peter Seidel. Efficient and robust annotation of motion capture data. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, page 17–26, New York, NY, USA, 2009. Association for Computing Machinery.

[48] Meinard Muller, Tido Röder, and Michael Clausen. Efficient content-based retrieval of motion capture data. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, page 677–685, New York, NY, USA, 2005. Association for Computing Machinery.

[49] Emily Napier, Gavia Gray, and Sageev Oore. Sequence modeling of motion-captured dance. In *Workshop on Machine Learning for Creativity and Design*, 2022.

[50] Emily Napier, Gavin Gray, and Sageev Oore. Spectral analysis for dance movement query and interpolation. In *Proceedings of the 8th International Conference on Movement and Computing*, MOCO '22, New York, NY, USA, 2022. Association for Computing Machinery.

[51] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 299, 2018.

[52] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[53] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

[54] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3D human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

[55] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[56] Ari Shapiro, Yong Cao, and Petros Faloutsos. Style components. In *Proceedings of Graphics Interface 2006*, pages 33–39, 2006.

[57] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Decoupled spatial-temporal attention network for skeleton-based action recognition, 2020.

[58] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT)*. `http:`http://ccrma.stanford.edu/ jos/mdft/`//ccrma.stanford.edu/~jos/mdft/`, accessed (10th Feb 2022). online book, 2007 edition.

[59] Kun Su, Xiulong Liu, and Eli Shlizerman. How does it sound? *Advances in Neural Information Processing Systems*, 34, 2021.

[60] Yu Sun, Qian Bao, Wu Liu, Yili Fu, Black Michael J., and Tao Mei. Monocular, one-stage, regression of multiple 3D people. In *ICCV*, 2021.

[61] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021.

[62] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models, 2015.

[63] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 501–510, Delft, Netherlands, November 2019.

[64] Guillermo Valle-Pérez, Gustav Eje Henter, Jonas Beskow, André Holzapfel, Pierre-Yves Oudeyer, and Simon Alexanderson. Transflower: probabilistic autoregressive dance generation with multimodal attention. 2021.

[65] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[67] Rudolf Von Laban. *Modern educational dance*. Princeton Book Company Pub, 1975.

[68] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[69] Phil Wang/lucidrains. mlm-pytorch. https://github.com/lucidrains/mlm-pytorch.

[70] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos, 2021.

[71] Liang Xu, Ziyang Song, Dongliang Wang, Jing Su, Zhicheng Fang, Chenjing Ding, Weihao Gan, Yichao Yan, Xin Jin, Xiaokang Yang, Wenjun Zeng, and Wei Wu. Actformer: A GAN-based transformer towards general action-conditioned 3D human motion generation, 2022.

[72] Zijie Ye, Haozhe Wu, Jia Jia, Yaohua Bu, Wei Chen, Fanbo Meng, and Yanfeng Wang. ChoreoNet: Towards music to dance synthesis with choreographic action unit. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, oct 2020.

[73] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations, 2023.

[74] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model, 2022.

[75] Mengyi Zhao, Mengyuan Liu, Bin Ren, Shuling Dai, and Nicu Sebe. Modiff: Action-conditioned 3D motion generation with denoising diffusion probabilistic models, 2023.

[76] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. Learning human motion representations: A unified perspective, 2023.

Appendix A

Appendix

A.1 PCA

PCA is a dimensionality reduction technique that captures the axes along which the data has maximum variance [30](1-6). PCA can be expressed as a weight matrix $\boldsymbol{W}$ transforming an input matrix $\boldsymbol{X}$ to produce $\boldsymbol{T}$,

$$\boldsymbol{T} = \boldsymbol{W}\boldsymbol{X}. \tag{A.1}$$

For example, when $\boldsymbol{X}$ is a matrix of joint angles, the dimension being reduced may be the time dimension, allowing PCA to capture components of joint motion through time.

To find the PCA weight matrix $\boldsymbol{W}$ we apply singular value decomposition (SVD) after centering the data to produce $\tilde{\boldsymbol{X}}$,

$$\tilde{\boldsymbol{X}} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^* \tag{A.2}$$

$$\boldsymbol{W} = \begin{bmatrix} \mathbf{v}_1^* \\ \mathbf{v}_2^* \\ \vdots \\ \mathbf{v}_N^* \end{bmatrix}, \tag{A.3}$$

where N is the number of principal components we have chosen.

A.1.1 Spectral embedding based on PCA

The motion data used in this work came from pose, or joint angle rotations, in SMPL [40] format. In Chapter 3 the STFT was calculated for each axis angle rotation of the joints, creating 72 sets of spectral features. To interpret this visually the features for each joint rotation can be plotted individually as a spectrogram, however

this is not a practical representation to aid in understanding the motion of the body as a whole.

In Sections 3.5 and 3.6 PCA is fit to the set of STFT spectral features, and we plot the first principal component. This shows a more general representation of the spectra of the motion instead of detailed features of a single rotation.

A.2 ICA

ICA is a statistical technique used to separate a set of mixed signals into their underlying independent components [6](591). Given a matrix of mixed signals $\mathbf{X}$, ICA aims to find a matrix of unmixing vectors $\mathbf{A}$ such that the recovered signals $\mathbf{S}$ can be obtained as:

$$\mathbf{S} = \mathbf{AX}. \tag{A.4}$$

In ICA, the goal is to estimate the unmixing matrix $\mathbf{A}$ such that the components in $\mathbf{S}$ are statistically independent. In the context of SMPL motion-capture data, we apply ICA over the joint rotations to isolate signals that capture components of joint motion over time.

A.3 SMPL pose representations

The SMPL [40] framework models human movement as a series of frames that contain pose and body shape parameters. A complete frame of motion can be described by:

- $\vec{\theta}$: pose, composed of axis-angle 3D rotation vectors $\vec{\omega}$: $\vec{\theta} = [\vec{\omega}_0^T, ..., \vec{\omega}_K^T]$ for K angles.
- $\vec{\beta}$: shape parameters.

$\vec{\theta}$ contains the 24 joints listed in section A.4 as axis-angle rotations. The axis-angle format consists of an axis of rotation and an angle of rotation around that axis, resulting in a 3-dimensional rotation vector. Matrices that define motions contain 72 joint rotations by N frames.

$\vec{\beta}$ is a 10-dimensional vector which contains parameters that alter a 3D mesh of the human body. This affects features such as height and bone length, and is used to recover joint positions through forward kinematics in this work.

A.4 Data simplification: Joint removal

In Chapter 4 we select a subset of joints from the SMPL formatted data to simplify our model. The full list of SMPL joints and whether they were included in our work is shown in Table A.1.

SMPL joints	Included in Chapter 4
Pelvis	Yes
Left Hip	Yes
Right Hip	Yes
Spine	No
Left Knee	Yes
Right Knee	Yes
Thorax	No
Left Ankle	Yes
Right Ankle	Yes
Upper Thorax	No
Left Toe	No
Right Toe	No
Neck	No
Left Collar	No
Right Collar	No
Jaw	No
Left Shoulder	Yes
Right Shoulder	Yes
Left Elbow	Yes
Right Elbow	Yes
Left Wrist	Yes
Right Wrist	Yes
Left Hand	No
Right Hand	No

Table A.1: SMPL joint table.

A.5 Bilateral data augmentation

The bilateral data augmentation performs a matrix multiplication on the original movement to mirror symmetrical joint angles. Demo Video 13 (https://youtu.be/RWaAtb1lG9M) shows an example movement and the augmented version.