Lab Vision Systems SS23

Human Pose Prediction using a Convolutional *MotionMixer*

Jon Breid

Rheinische Friedrich-Wilhelms-Universität Bonn, Regina-Pacis-Weg 3,
D-53113 Bonn, Germany, https://www.uni-bonn.de

Abstract. Understanding and predicting human motion is critical for effective human communication, human-computer interaction and various real-world applications, including human tracking and virtual or augmented reality. Reducing computational delay through pose prediction improves interactions and safety, particularly in human-robot collaborations such as production lines.
Part of human pose prediction is the prediction of human poses from past poses, that are encoded by joint positions. This is the problem addressed in this work.
In this work we present a motion prediction system based on the *MotionMixer*[3] framework using convolutional layers. A *CRMixer* block is presented that learns the data with 2-dimensional convolutional kernels, and a *CSTMixer* block that learns about temporal and spatial information separately with one-dimensional convolutions. The different architectures are tested with different kernel sizes. Our system shows promising results on the Human3.6m dataset, accurately predicting ten frames over 400 ms with ten seed frames.

1 Introduction

The ability to predict and understand human motion is important for effective human communication and interaction. This is also the case for human-computer interaction, like tracking of human movement and posture. It can lead to better interaction not only in the real world, such as human tracking, but also in virtual or augmented reality. Lag can be reduced by reducing computation time by pre-computing information based on predicted poses. Also in human-robot interaction, predicting human movement is an important aspect to increase safety when humans interact with robots [12], for example in production lines.

The general problem of predicting human motion and poses can be divided into two sub-problems. One part of the problem is to recognise human poses from sensor data, such as images. Key points for joints are calculated and used to generate a skeleton-like structure for a given pose. There are many different approaches to human pose estimation, as shown by Qi et al. [5]. For example, Toshev et al. [13] use deep neural networks to predict joint positions. The second part of the problem is to predict future poses based based on past joint

positions. This work concentrates on the second part of the problem. There are many different approaches to solve the problem of predicting human poses. Martines et al. [10] achieve good results with a relatively simple architecture based on GRUs [4]. Torkar et al. [12] predict human motion to increase safety in human-robot interaction. Aksan et al. [1] use Transformer to predict poses. And Bouazizi et al. present a *MotionMixer* [3] to learn about the spatial and temporal information to predict poses.

This work presents a motion prediction system based on the *MotionMixer* architecture using convolutional layers. Different architectures are tested and compared. The system shows good results on the Human3.6m dataset [8] when predicting ten frames with a total length of 400 ms with ten seed frames.

2 Method

In the following section, the dataset and data format used for training and evaluation and the architecture of the human pose prediction system are presented in detail. The proposed system is implemented with PyTorch [11].

2.1 Dataset

Human3.6 dataset The dataset used in this paper is based on the Human3.6m dataset [8], which consists of videos of actors performing 15 specific actions such as 'walking', 'sitting' and 'clapping'. The dataset used here consists of pre-calculated joint coordinates that are extracted from the original Human3.6m dataset. Each pose is encoded by 33 joints as a skeleton-based model in exponential map format with fixed bone lengths. 17 of the original 33 joints are used in this work. An example pose with 17 joints and the skeleton connecting them is shown in Figure 1. To also test joint coordinates that are given in three dimensional euklidean space, the exponential map format can be converted to euklidean space. The original data is downsampled from 25 Hz to 25 Hz by taking every second pose frame. The number of seed and prediction frames is set to 10. However, longer prediction sequences are also generated for further evaluation of the results. The given dataset is divided into a training set (actors *S1, S6, S7, S8, S9*), a validation set (actor *S5*) and a test set (actor *S11*) according to the official division.

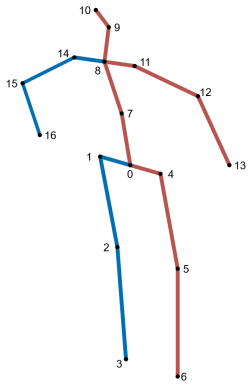

Fig. 1. Representation of a single pose in the dataset with the 17 joints in use and their indices.

AIS Dataset In addition, another dataset consisting of eight sequences of a human moving is used to evaluate the generalisability of the models. Each pose

is encoded by 19 joints as a skeleton-based model. The joint coordinates are given in three-dimensional Euclidean space. The 19 joints are transformed into 17 joints according to Figure 1. As this dataset does not use a joint for the spine (joint index 7 in Figure 1), the location for this joint is calculated as the average of joints 0 and 8. Additionally joint 10 is also not used in this dataset, so the position of this joint is calculated as the average of the position of the left and right ear. To further ensure that the figures from this dataset and the Human3.6m dataset are approximately the same size, the joint positions in this dataset are scaled according to the difference in the length of the right thigh in both datasets.

2.2 Model Architecture

The architecture used for the model is based on the architecture of a *MotionMixer* [3] expanded by convolutions. The general architecture of the system presented in this work is shown in Figure 2. The model directly predicts a sequence of human poses from a sequence of input poses and consists of three blocks. A block that computes a pose embedding from the original data, a convolutional mixer block, and a block that predicts the output poses from the embedding dimension.

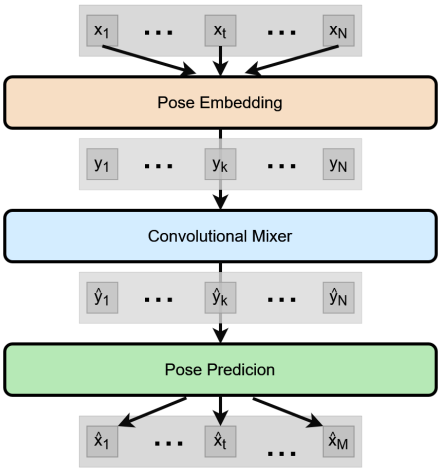

Fig. 2. Diagramm showing the architecture of a *STMixer* block.

Pose Embedding In a first step, each input vector of a sequence is passed through an encoding network consisting of one linear layer. Each pose is passed through the network independently and separately. This network transforms the input vector from its input size to a larger feature size.

Pose Prediction In the final *Pose Prediction* of the network, the processed feature vector is transformed back to the original input size to predict the new poses. The block consists of one linear layers. Each vector is passed through the network independently.

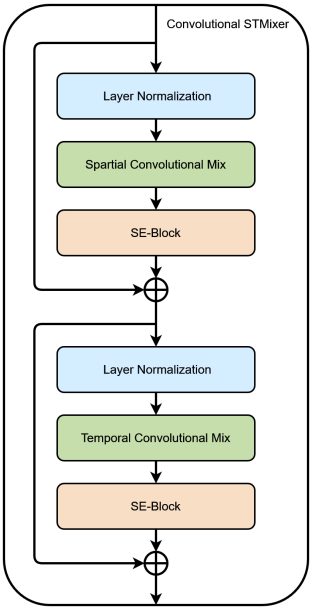

Fig. 3. Diagramm showing the architecture of a STMixer block.

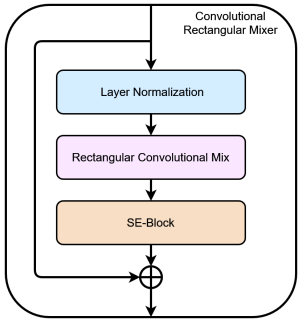

Fig. 4. Diagramm showing the architecture of a *STMixer* block.

Convolutional Mixer The task of the *Convolutional Mixer* is to learn the temporal and spatial dependencies between the joints of the poses. The *Convolutional Mixer* consists of one or more mixer blocks. There are two different architectures for the mixer block presented in this work, a *Convolutional STMixer* and a *Convolutional Rectangular Mixer*.

A Convolutional STMixer is a feed-forward network consisting of two units, each with a skip connection, which processes the input data sequentially as shown in Figure 3. Each of these units start with a layer normalisation block [2], which re-scales and re-centres the activations according to their mean and variance. This is followed in the first part of the block by a *Spatial Convolutional Mix* block and in the second part by a *Temporal Convolutional Mix* block. Each of these blocks consists of one or more one-dimensional convolutional layers, followed by the GeLU activation function [6] and a dropout layer. The *Spatial*

Convolutional Mix block convolves the input data only in the spatial dimension, while the *Temporal Convolutional Mix* block convolves the input data only in the temporal dimension. If there is more than one convolutional layer, the channel size is increased from one to a hidden channel size and then decreased to one again. These blocks are each followed by a squeeze and exitation block (*SE-Block*) [7], which *SE-Blocks* share the same weights. In the following, *Convolutional STMixer* will be referred to as *CSTMixer*.

A Convolutional Rectangular Mixer is a feed-forward network with a skip connection as shown in Figure 4. As with the *Convolutional STMixer*, the first step is to normalize the input. Then the input data is passed through a *Rectangular Convolutional Mix* block, which works similarly to a Spatial or Temporal Convolutional Mix block. The *Rectangular Convolutional Mix* block consists of one or more two-dimensional convolutional layers, each followed by the GeLU activation function and a drop-out layer. If there is more than one convolutional layer, the channel size is increased to a hidden channel size and then returned to the original channel size of one. Two-dimensional convolution simultaneously convolves the input data in the spatial and temporal dimensions by treating the input data as a two-dimensional, single-channel image. In the following, *Convolutional Rectangular Mixer* will be referred to as *CRMixer*.

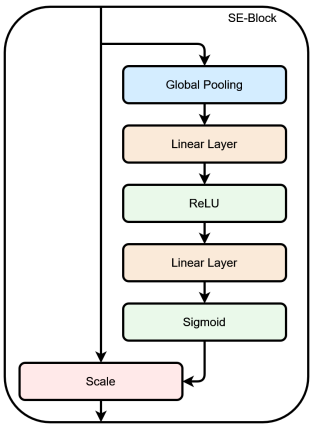

Fig. 5. Diagramm showing the architecture of a squeeze and exitation block (*SE-Block*).

A SE-Block rescales the input data for each time step as shown in Figure 5. The input activations are averaged in the temporal dimension. The output is a vector of the length of the number of frames. This vector is then squeezed to a vector of size $\lfloor \frac{\#frames}{r} \rfloor$ with $r > 0$, followed by a rectified linear unit $ReLU(x) = max(0, x)$. The activations are then passed through another linear

layer which increases the vector length to the number of frames again. This is followed by the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$. The output vector contains a scaling factor for each time step. These scaling factors are used to scale the input data by pointwise multiplication. This re-scaled vector is the output of the *SE-Block*.

3 Experiments

The system is trained with different hyperparameters and different architectures. It is trained to predict the output sequence directly from the input sequence. To predict longer output sequences, the prediction from a previous step is used as input until the desired prediction length is reached. To be able to compare the models directly and to have similar training times, all models are trained for 60 epochs with the optimizer Adam [9]. The model parameters are all randomly initialized before each training. The batch size is set to 64 in all experiments and the learning rate is set to 10^{-2}, decreasing by a factor of 0.1 every 10 epochs. These are the same parameters used in the original implementation of *MotionMixer*. The hidden dimension in each spatial, temporal, and rectangular *Convolutional Mix* block is set to 32 in all experiments, and the number of convolutional layers is set to 3. Additionally, the dropout rate in each of these blocks is set to 0.1. The value of r is set to 8 in all *SE-Blocks*.

The loss functions used are the mean L_2 loss and the mean per joint position error (MPJPE) according to the following formulas where J is the number of joint coordinates, T_f is the number of predicted frames, $\hat{x}_{t,j}$ is the ground truth value and $x_{t,j}$ is the predicted joint value:

$$L_{MPJPE} = \frac{1}{J \cdot T_f} \sum_{j=1}^{J} \sum_{t=1}^{T_f} ||\hat{x}_{t,j} - x_{t,j}||_2 \tag{1}$$

$$L_2 = \frac{1}{J \cdot T_f} \sum_{j=1}^{J} \sum_{t=1}^{T_f} ||\hat{x}_{t,j} - x_{t,j}||_2^2 \tag{2}$$

L_2 is used when the pose vectors are given in three-dimensional angular space. MPJPE is used when the pose vectors are given in three-dimensional Euclidean space.

4 Results

Models with different parameters and the two different architectures for the *Convolutional Mixer* are trained and evaluated. The results are presented in the following section.

The models are evaluated using different metrics. Models trained on pose vectors with angular coordinates are evaluated using the *Euler Angle* metric and a *Geodesic* metric. The *Geodesic* metric calculates the average distance between the rotation matrices for each joint. The *Euler Angle* metric calculates the average difference between the joint angles in the Euler representation of angles. Additionally, the predicted poses are also translated into Euclidean 3D space and evaluated with the mean per joint position error (MPJPE) in millimeter according to Formula 1 and the area under the curve (AUC) value for different thresholds with percentag of correct keypoints (PCK) [14]. Models trained on pose vectors using three dimensional coordinates in Euclidean space are only evaluated using MPJPE and the PCK AUC value. For all models with reasonably good results, the metric values on the validation set of the Human3.6m dataset decrease, or in the case of AUC PCK increase, and converge over time as expected.

4.1 Evaluation

First the two model architectures, with a *CSTMixer* or a *CRMixer* are evaluated when using input data in exponential map format on the Human3.6m dataset. Table 1 shows the results with the best performing models with data in exponential map format.

	10 Frames				25 Frames			
	Geod	Euler	PCK	MPJPE	Geod	Euler	PCK	MPJPE
CSTMixer	0,118	0,675	0,637	52,23	0,17	0,96	0,55	73,4
CRMixer	0,129	0,791	0,647	54,45	0,172	1,007	0,567	73,6

Table 1. Table showing the results for the best performing models using *CSTMixer* or *CRMixer* when using data in exponential map format for 10 and 25 prediction frames. The results are archived on the test set of the Human3.6m dataset. More results for models using different parameters than the two models shown here can be found in the Appendix in Table 5.

The best performing model with *CSTMixer* has a hidden pose feature length of 128, 5 *CSTMixer* layers, a temporal kernel size of 9 and a spatial kernel size of 51. Models with smaller kernel sizes achieve lower scores, indicating that they are not complex enough to learn the behavior in the training data, and larger kernel sizes tend to slightly overfit the training data. The best performing model using *CRMixer* has a hidden pose feature length of 128, 5 *CRMixer* layers, and a kernel size of (9,15).

The results with the model using *CSTMixer* layers when predicting 10 frames are slightly better for Geodesic Error, Euler Error, and MPJPE compared to the model using *CRMixer*. The Geodesic Error is slightly lower at 0.118 compared to 0.129 and the Euler Error is lower at 0.675 compared to 0.791. In addition,

the MPJPE is also slightly lower at 52.23 compared to 54.45. However, the AUC PCK is slightly better for the model using *CRMixer* at 0.647 compared to 0.673. When comparing both architectures for 25 prediction frames, the results are similarly close, with the model based on *CSTMixer* being slightly better for the Geodesic Error, Euler Error, and MPJPE. For the AUC PCK value, the model using *CRMixer* layers is slightly better.

The results when predicting 25 frames are significantly worse than predicting only 10 frames. MPJPE increases by about 20 and AUC PCK increases by about 0.09. The geodesic error increases by about 0.045 and the Euler error increases by 0.285 for *CSTMixer* and 0.216 for *CRMixer*. The much worse results are expected since the model is not trained on longer predictions and longer sequences are generally more error-prone due to their length.

The visualization of the predicted sequences shows that the model is able to learn general joint positions and general movement patterns. However, there is often a small jump between the last input pose and the first predicted pose. This effect is more pronounced when using *CRMixer* layers.

	10 Frames		25 Frames	
	PCK	MPJPE	PCK	MPJPE
CSTMixer	0,826	24,06	0,699	48,9
CRMixer	0,816	25,31	0,699	48,27

Table 2. Table showing the results for the best performing models using *CSTMixer* or *CRMixer* when using data in three dimensional Euclidean format for 10 and 25 prediction frames. The results are achived on the test set of the Human3.6m dataset. More results for models using different parameters than the two models shown here are in the Appendix in Table 6.

The performance of the two model architectures is evaluated on the Human3.6m dataset with poses encoded in 3D Euclidean space. Table 2 shows the results for the best performing models.

The best performing *CSTMixer* model uses 128 as the feature length of each pose, has 5 *CSTMixer* layers, a temporal kernel size of 9 and a spatial kernel size of 51. The best performing *CRMixer* model has the same feature length and number of *CRMixer* layers, but a kernel size of (9,9). Other models with different kernel sizes give very similar but slightly worse results.

The results for predicting 10 frames are slightly worse when using a *CRMixer* with a MPJPE value of 25,31 compared to 24,06 and a AUC PCK value of 0,816 compared to 0,826. The results for predicting 25 frames are very similar to each other, with an AUC PCK of about 0.7. The MPJPE value is slightly worse when using *CSTMixer* layers at 48.9 compared to 48.18. Again, the values when predicting 25 frames are significantly worse than the values when predicting 10 frames, with a difference of about 25 in MPJPE and a difference of about 0.11 in the AUC PCK values. However the values when predicting 25 frames still

achieve better values compared to predicting 10 frames when using exponential map data format.

The visualization of predicted sequences shows that both models are able to learn general joint positions and to continue movements smoothly. However, the predicted poses are often close to the last input frame and do not exhibit as much movement compared to the ground truth sequence.

These results are significantly better than the results obtained using data in exponential map format. This may be because the model trained on 3D Euclidean data is specifically trained to achieve the lowest MPJPE, since that is the loss function used, or because 3D Euclidean data is inherently easier to learn and predict with these architectures.

	10 Frames		25 Frames	
	PCK	MPJPE	PCK	MPJPE
CSTMixer	0,481	81,78	0,363	124,12
CRMixer	0,506	77,86	0,41	111,41

Table 3. Results with the best performing model using a *Convolutional STMixer* or a *Convolutional Rectangular Mixer* on the AIS dataset when predicting 10 or 25 frames.

To assess the generalizability and robustness of the two models, they are additionally evaluated on the AIS data when generating 10 or 25 frames. The results are shown in Table 3. The performance when using *CSTMixer* layers is better for 10 and 25 frames than when using *CRMixer* layers. The performance in general is significantly worse than the performance of these two models on the Human3.6m test dataset.When generating 10 frames, the MPJPE is approximately 80 and the AUC PCK value is 0.5, which is significantly worse compared to the results on the Human3.6 dataset of about 24 and 0.826. The results when predicting 25 frames are also much worse, with MPJPE values above 111 and AUC PCK values below 0.41, compared to MPJPE values of about 49 and PCK values of 0.7. This indicates that the models are not able to generalize well on the AIS data.

	Exponential Map Data				3d Euklid Data	
Kernel	Geod	Euler	PCK	MPJPE	PCK	MPJPE
(9,1)	0,171	1,010	0,503	72,12	0,813	27,91
(1,127)	0,152	0,819	0,556	62,39	0,691	51,04

Table 4. Table showing the results of using *CRMixer* layers with a kernel in only temporal or spatial direction when predicting 10 frames.

To evaluate the influence of the temporal or spatial dimension when using *CRMixer* layers, two models are trained with a kernel of size (9,1) extending only

in the temporal direction or a kernel of size (1,127) extending only in the spatial direction. Table 4 shows the results when using exponential map or 3D Euclidean data format. When using data in exponential map format, the model using only the spatial dimension in its *CRMixer* layer achieves a better result with a MPJPE of 62.39 compared to 72.12. The other metrics also show better results. This indicates that the spatial dimension is particularly important when using data in exponential map format. The models trained on 3D Euclidean data using only the spatial dimension perform worse with an MPJPE of 51.04 compared to 27.91. This shows that the temporal dimension is more important when using 3D Euclidean data. The MPJPE value obtained using only the temporal dimension is close to the value obtained using both dimensions. This indicates that learning a general direction of motion of each joint is a large part of learning to predict a short frame sequence.

4.2 Discussion

Models have been successfully trained using the two different architectures. When using data in exponential map format, the models are able to learn general human poses and motion, but the results are worse compared to the results of the original implementation of the *MotionMixer* with a MPJPE 33.6 comapred to approximately 53. This suggests that convolutions are not not well suitad for predicting human poses. This may be as there is no inherent neighborhood in the pose representation that convolutions rely on.

When using data in 3D Euclidean format, the models perform much better. The choice between *CSTMixer* and *CRMixer* seems to have little impact, as their performance is nearly identical. Even models that use only kernels extending in the spatial direction are able to achieve a good result, indicating that learning a general motion pattern in each of the joint coordinates is sufficient to predict poses to a good degree. This suggests that learning to predict poses for a short duration in 3D Euclidean space is a less complex task compared to predicting angles.

The results for predicting 25 frames compared to 10 frames are much worse for all models. This is to be expected since the prediction is made over a longer range, which introduces more uncertainty into the prediction. Also the models are only trained to specifically predict 10 frames and not to use previous predictions as input.

When testing the generalizability on the AIS dataset, the models perform much worse. This may be due to the different bone lengths between the AIS data and the Human3.6m dataset. Joint positions could to be predicted where they would be in the Human3.6m dataset. As these models are optimized for the Human3.6m dataset with its specific bone lengths, different lengths are never observed and learned. Poses that deviate from these specific lengths are never seen and could therefore cause completely unexpected behavior. In addition, the AIS data seems to much more noisy compared to the Human3.6m dataset, which can make predictions less precise. Also the joints for the top of the head and the spine are interpoated between other joints in the AIS data. A better translation

from the AIS data to the data format of the Human3.6m dataset might lead to better results on the AIS data.

5 Conclusion

This work presents a human pose prediction system based on a convolutional *MotionMixer*. A *CRMixer* is presented that learns about the data with 2-dimensional convolutional kernels, and a *CSTMixer* that learns about temporal and spatial information separately with one-dimensional convolutions. The different architectures for learning spatial and temporal information are tested with different kernel sizes. The models are successfully trained on the Human3.6m dataset and show good results, especially when using the 3D Euclidean pose representation. Movement is continued smoothly, but to a lesser extent compared to the ground truth. However, these models are not able to generalize very effectively to the AIS dataset. General poses are usually recognized by the system, but they are often imprecise. Using poses in exponential map format leads to a worse result, where sometimes there is a jump between the last input frame and the first predicted frame, but general poses and motion patterns are still learned and predicted.

Further research is possible to improve the presented system. For example, different learning rates, different optimizers or other loss functions for training, or different kernel sizes and other parameters for the model can be explored. In addition, data augmentation such as rotating the poses on a horizontal axis, using reversed sequences or adding some noise to the poses could be used to improve the system's ability to generalize to other data and to be more robust to noisy data.

References

1. Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction, 2021.
2. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
3. Arij Bouazizi, Adrian Holzbock, Ulrich Kressel, Klaus Dietmayer, and Vasileios Belagiannis. Motionmixer: Mlp-based 3d human body pose forecasting, 2022.
4. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.
5. Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.
6. Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.
7. Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.
8. Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.
9. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
10. Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks, 2017.
11. Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
12. Chris Torkar, Saeed Yahyanejad, Horst Pichler, Michael W. Hofbaur, and Bernhard Rinner. Rnn-based human pose prediction for human-robot interaction. 2019.
13. Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014.
14. Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225, 2021.

6 Appendix

	Exponential Map Format				3D Euclidean Data	
Parameter	Geod	Euler	PCK	MPJPE	PCK	MPJPE
128_5_9_51	0,118	0,675	0,637	52,23	0,826	24,06
128_5_7_51	0,132	0,671	0,630	54,65	0,823	24,23
64_3_9_51	0,142	0,795	0,544	69,13	0,807	26,94
64_5_9_51	0,132	0,749	0,568	64,58	0,799	27,73
128_3_9_51	0,137	0,767	0,605	59,51	0,822	24,58
128_7_9_127	0,156	0,867	0,543	69,95	0,825	24,13
128_5_9_127	0,137	0,775	0,61	57,88	0,822	24,16
128_5_5_127	0,129	0,713	0,608	56,65	0,811	26,14
128_5_7_127	0,17	0,913	0,564	66,32	0,824	24,22
128_5_9_99	0,118	0,671	0,619	55,02	0,823	24,4
128_5_9_75	0,143	0,819	0,566	64,85	0,823	24,52
128_5_9_31	0,132	0,761	0,585	61,57	0,823	24,51
128_5_9_21	0,154	0,862	0,549	68,42	0,824	24,21
128_5_9_11	0,142	0,794	0,558	65,97	0,823	24,45
128_5_5_11	0,143	0,807	0,554	66,5	0,818	25,2
128_5_5_7	0,141	0,794	0,57	64,17	0,817	25,23
128_5_5_51	0,132	0,757	0,631	54,45	0,809	25,86
128_5_3_51	0,149	0,843	0,542	69,61	0,823	24,68

Table 5. The table shows the results for different hyperparameters for models using *CSTMixer* for 10 output frames. The results are shown for different metrics when the input vector is either in exponential map (angle) format or in three-dimensional Euclidean space. The metrics are the geodesic metric (Geod), the Euler angle metric (Euler), the AUC PCK score (PCK), and the MPJPE. The parameter string starts with the length features after the *Pose Embedding*, the second number is the number of *Convolutional STMixer* layers, the third and fourth numbers are the size of the kernel in each *Spatial Convolutional Mix* block and *Temporal Convolutional Mix* block respectively.

	Exponential Map Data				3D Euclidean Data	
Parameter	Geod	Euler	PCK	MPJPE	PCK	MPJPE
128_5_9_9	0,13	0,798	0,642	55,19	0,816	25,31
128_5_5_5	0,159	0,909	0,548	69,44	0,814	25,33
128_5_3_3	1,245	6,364	0,203	444,1	0,816	25,44
128_5_7_7	0,131	0,791	0,643	54,51	0,810	25,49
128_5_9_3	0,136	0,798	0,605	59,32	0,814	25,35
128_5_3_31	0,133	0,793	0,640	54,59	0,812	25,86
128_5_9_31	0,130	0,795	0,645	54,54	0,813	25,73
128_5_7_15	0,13	0,798	0,642	55,19	0,799	27,72
128_5_9_15	0,129	0,791	0,647	54,45	0,810	26,12
128_5_9_61	0,140	0,801	0,601	58,01	0,809	26,35
128_5_7_61	0,136	0,795	0,608	57,86	0,810	26,13
128_5_5_61	0,138	0,800	0,602	57,99	0,808	26,18
128_5_9_1	0,146	0,83	0,548	68,1	0,812	26,11
128_5_1_127	0,131	0,795	0,64	55,39	0,699	46,04

Table 6. The table shows the results for different hyperparameters for models using *RMixer* for 10 output frames. The results are shown for different metrics if the input vector is either in exponential map (Angle) format or in 3D Euclidean space. The metrics are the geodesic metric (Geod), Euler angle metric (Euler), the AUC PCK score (PCK) and MPJPE. The parameter string starts with the length features after the *Pose Embedding*, the second number is the number of *CRMixer* layers, the third and fourth numbers are the sizes of the kernel in temporal and spatial direction. Larger convolutional kernels like (9.99) were tested, but training took much longer and the runtime always crashed during training.