AML Project
Graph Network-based Simulator (GNS) for Human Pose Forecasting

Francesco Bonanni
Vincenzo Romito

Eleonora Turchetti
Theo Courbis

Abstract

The aim of this work is to compare the performance of an *auto-regressive* model against the *one-shot* model proposed by Sofianos et al. (2021). The auto-regressive model used is a fully Graph Convolutional Neural Network model inspired by the work of Sanchez-Gonzalez et al. (2020), in which they show that the model is able to learn the complex dynamic interactions of thousands of particles. We empirically prove that this neural network is also suitable to model the human pose forecasting problem. In particular we show that on a recent and large-scale benchmark Human3.6M (Ionescu et al., 2014) the auto-regressive model outperforms the one proposed in Sofianos et al. (2021) in short-term predictions by 14% on average, while in the long-term its performances gets worse by 5% on average. Moreover, instead of training multiple one-shot models for different prediction horizon lengths (as in (Sofianos et al., 2021)), we train a single model exploiting its auto-regressive nature. Our source code is available here.

1 Introduction

Forecasting future human poses is the task of modelling the complex structured-sequence of joint spatio-temporal dynamics of the human body. There are mainly two different strategies to forecast human motion. The *direct strategy* (e.g. Sofianos et al. (2021)) involves the utilization of one model that is capable of predicting the entire future trajectory in one shot. However, as pointed out by Zhou et al. (2023) the losses of the last time steps are heavier weighted than that of the beginning time steps in the objective function, which can limitate the learning process. The *recursive strategy* used in Sanchez-Gonzalez et al. (2020) involves the reuse of a one-step prediction model multiple times, in which the predictions of the previous time steps are used for the prediction of the following time step. One limitation of this model is that small prediction errors amplify significantly over time because of their loop structure for prediction. Here we will experiment with this recursive prediction approach. We propose to forecast human motion with an auto-regressive Graph Convolutional Network inspired by the work of Sanchez-Gonzalez et al. (2020) in which the authors propose a graph network-based simulator to perform physics simulations. A practical reason why we could prefer an auto-regressive instead of a one-shot model is that in the case of one-shot model proposed by Sofianos et al. (2021) they trained a different model for each prediction horizon. Since they want to evaluate the model for 8 different prediction frames, they have trained 8 different models (each one has roughly $90k$ parameters), for a total of $750k$ parameters. Even though in principle it is possible to combine a smaller number of one-shot models in order to predict several horizons, the task is straightforward within the auto-regressive framework, where a single (usually bigger) model is used for all predictions steps. Our best model has 1.5M parameters.

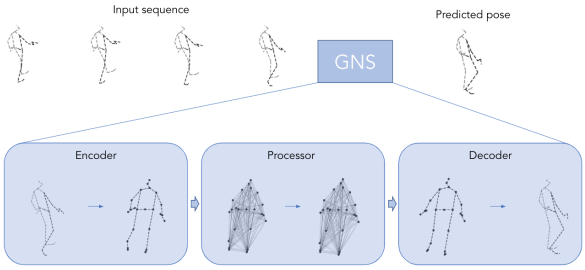

Figure 1: Model Architecture.

2 Related work

2.1 Temporal Modelling

Most recent work in human pose forecasting has leveraged Recurrent Neural Networks (RNN) (Jain et al., 2016), as well as recurrent variants. These

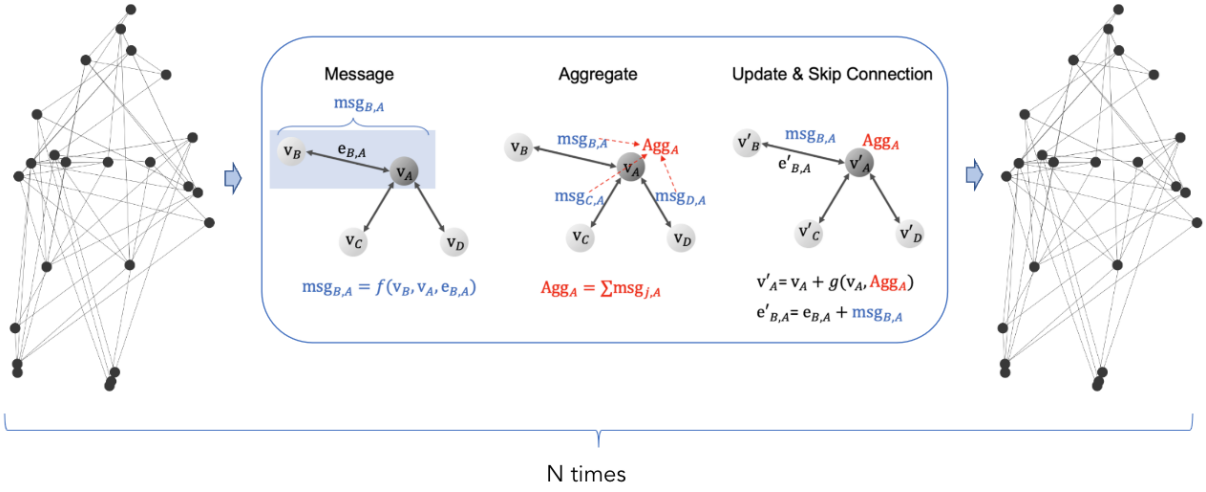

Figure 2: **Processor**. [image source]

techniques are flexible, but they have issues with long-term predictions such as inefficient training and poor long-term memory (Barsoum et al., 2017). Research has attempted to tackle this, by training with generative adversarial networks (Gui et al., 2018) and by imitation learning (Tang et al., 2018a). Emerging trends have adopted (self-)attention to model time (Mao et al., 2019), which also applies to model spatial relations (Tang et al., 2018b). One of the best performing model is STS-GCN (Sofianos et al., 2021): it models the human pose dynamics only with a graph convolutional network (GCN), including the temporal evolution and the spatial joint interaction within a single-graph framework, which allows the cross-talk of motion and spatial correlations.

2.2 Representation of human body

Nearly all literature adopts 3D coordinates or angles. Martinez et al. (2017) has noted that encoding residuals of coordinates, thus velocity, may be beneficial. Graphs are a natural choice to represent the body. These have mostly been hand designed, mainly leveraging the natural structure of the kinematic tree (Jain et al., 2016), and encoded via Graph Convolutional Networks (GCN) (Kipf and Welling, 2016). Most recently, research has explored all joints linked together and learnt graph edges (Mao et al., 2019).

3 Proposed method explained

The joints are 3D vectors $\vec{r}_j^{(t)}$ representing the position of joint j at time t. The motion history is a sequence S of T matrices $X^{(t)}$ for frames $t = 1, \ldots, T$. Each $X^{(t)}$ is an $n \times 3$ matrix where n is the total number of joints. The aim is to predict $X^{(T+1)}$.

3.1 Model Architecture

The model architecture is depicted in Figure 1.

Encoder. The encoding process is described in Figure 6. A graph G is built such that each node v_i contains the sequence of velocities computed using finite differences for $t = 1, \ldots, T-1$. Each edge $e_{i,j}$ contains the relative displacement between node v_i and node v_j at time T and the absolute distance $||v_i - v_j||$ at time T.

Processor. Once we construct the graph G in Figure 6, we connect two vertices if their euclidean distance is lower than a *connectivity radius* r_c. This value is an hyperparameter and we will try different values. We will also consider a radius $r_c = \inf$ which is equivalent to a fully connected graph. Then, a message-passing scheme is performed to the input graph using two MLP f and g. The first one is used for message features while g is used for updating the nodes features following the process described in Figure 2. This process is repeated N times and we will refer to N as the *number of message passing steps*. The processor is described in Figure 2.

Decoder. After updating nodes and edges features, we simply apply another MLP from the nodes features to the predicted positions for each

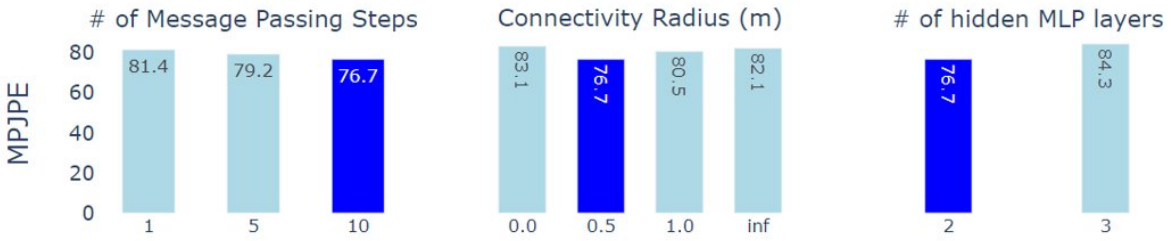

Figure 3: Hyperparameters tuning: the best performances are obtained using 10 *message passing steps*, a *connectivity radius* of $0.5m$ and 2 *hidden MLP layers*.

joint and we obtain $X^{(T+1)}$ (to be precise, we output the velocity and then we compute the next positions using finite differences).

Once we have $X^{(T+1)}$, we can use this one-step prediction model multiple times in an auto-regressive fashion: the predictions of the previous time steps are used for the prediction of the following time step.

4 Dataset and metrics

Dataset. The dataset used is *Human3.6M* (Ionescu et al., 2014), one of the largest motion capture datasets widely used in human pose forecasting, which consists of 3.6 million human poses and corresponding images captured by a high-speed motion capture system. The dataset is organized into 15 actions (e.g. Walking, Smoking, Greeting) performed by 7 different actors represented as skeletons of 32 joints, but we only use 22, the relevant ones. In order to carry out a consistent comparison with the results obtained by Sofianos et al. (2021), we used the same dataset split: subject 11 (S11) for validation, subject 5 (S5) for testing, and all the others (S1, S6, S7, S8, S9) for training.

Metric. In order to benchmark against the work of Sofianos et al. (2021), we used the same metric MPJPE (*Mean Per Joint Position Error*), typically used for 3D Pose Estimation, which corresponds to the mean eucledian distance between ground truths and predictions for all the joints.

Implementation details. The optimizer is Adam. The learning rate is set to 0.01, decayed by a factor of 0.1 every 5 epochs after the 20-*th* epoch and the batch size is 256 (this is the same setting used by Sofianos et al. (2021)). All our experiments are trained for 50 epochs on an NVIDIA Tesla T4 GPU.

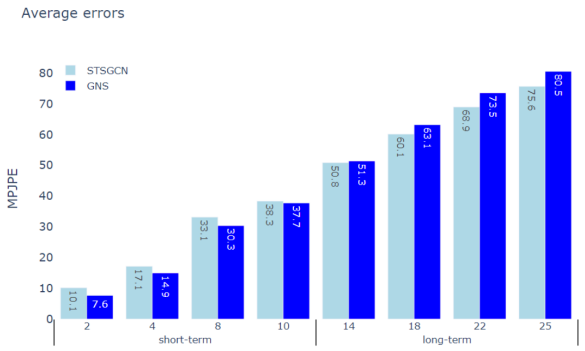

Figure 4: Average MPJPE errors in mm for short and long-term prediction.

5 Experimental results and benchmark

First of all, we ran the codes provided by the two papers to verify their correctness and reproducibility. As for code by Sofianos et al. (2021), we managed to get the same results, while as for code by Sanchez-Gonzalez et al. (2020), we ran it for fewer epochs (they used millions of epochs), still getting acceptable results. Then, starting from the implementation[1] of Sanchez-Gonzalez et al. (2020), written in Tensorflow, we converted[2] it in PyTorch and we checked qualitatively (by looking at the simulations) that we obtained similar results. Then we adapted their model to our problem, by removing all the unnecessary features (boundaries features, external fields and particle types). Moreover, we only predict velocities instead of accelerations. Then we performed a quantitative evaluation of the model developed by Sanchez-Gonzalez et al. (2020) adapted by us to human pose forecasting, against the model of Sofianos et al. (2021), both for short-term (< 500 msec) and long-term (> 500

[1] https://github.com/deepmind/deepmind-research/tree/master/learning_to_simulate

[2] taking inspiration from https://github.com/geoelements/gns

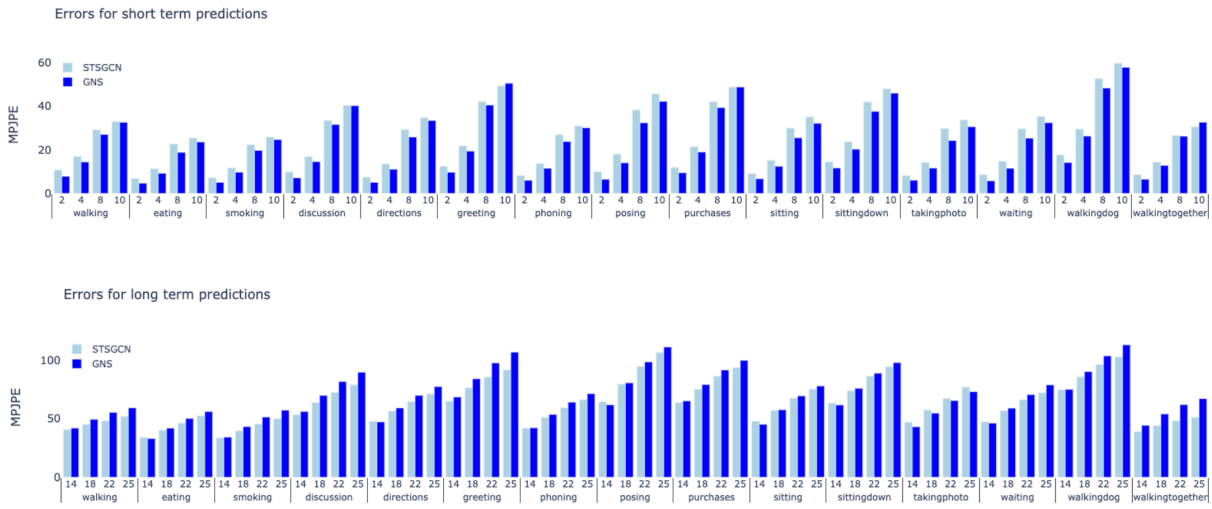

Figure 5: MPJPE errors in mm for short and long-term prediction for all actions.

msec) predictions. Both models take 10 frames (400 msec) as input, and predict future poses for the next 2-10 frames (80-400 msec) in the short-term case and for 14-25 frames (560-1000 msec) in the long-term case. A hyperparameters tuning was performed on the *number of message passing steps*, *connectivity radius* and *number of hidden MLP layers*. As we can see in Figure 3, the best parameters are respectively 10, 0.5 and 2. We also found that the connectivity radius has an important role for obtaining a more balanced accuracy along the horizon of frame predictions or a higher performance on short term predictions: high connectivity radius yield low MPJPE on short-term but high MPJPE on long-term sequences. The hyperparameter tuning was performed on the validation set without using noise. Training the model for 50 epochs with the best configuration and using the optimizer described in previous section, on the test set we obtain the average results over all actions shown in Figure 4 with the results obtained by Sofianos et al. (2021). In Figure 5 we report the errors obtained for each actions. In Table 1 and Table 2 we summarize all the results obtained. Looking at Figure 4 we can see that there are two differen behaviour: in the short-term regime our model consistently outperforms the STSGCN model, and by looking at Figure 5 we can see that this holds for almost each action. Considering the average MPJPE errors, the improvement of the model ranges from 3% in the case of 10 time frames, up to 27% improvement for the case of 2 time frames, with an average improvements of 14%. However, we can see that the performance deteriorates in the long-term prediction except for almost half of the actions in the 14-frames predictions. The worsening of the model, considering the average of the errors, ranges from a maximum of 9% in the case of 1000-frames predictions to a minimum of 0.3% in the case of 560-frames predictions. The average deterioration of the model is 5%. As expected, the more the prediction horizon increases, the worse the performance gets. We think this is due to the accumulation of errors. Other experimental results are that directly predicting positions does not yield better performances. Same for the input window length: performances are stable does not yield better results after 10.

5.1 Adding Noise To Mitigate Long-Term Prediction Errors

As explained by Sanchez-Gonzalez et al. (2020), their auto-regressive architecture suffers from error accumulation over long-term predictions. The model is trained on ground-truth one-step data, and so they are never presented with input data corrupted by this sort of accumulated noise. So, in testing phase, when a long-term predictions is generated by feeding the model with its own noisy previous predictions as input, the fact that its inputs are outside the training distribution may lead it to make larger errors, and thus rapidly accumulate further errors. To mitigate this problem, at training phase, they corrupt the input velocities of the model with some noise. They say that the best type of noise to simulate the accumulation of long-term prediction errors is *random walk noise*: noise is drawn according to $\mathcal{N}(0, \sigma_v)$ for each input state, added it to the noise of the previous state

in sequence as in a random random walk, adjusting input positions. We have implemented the same type of noise using various standard deviations σ_v $(3 \cdot 10^{-4}, 6.7 \cdot 10^{-4}, 3 \cdot 10^{-5})$, but in any case the performance deteriorates, both in short-term and long-term predictions, revealing that errors accumulation in the prediction cannot be modeled using a random walk.

6 Conclusions and future work

Our main finding is that a GNN with an auto-regressive strategy it is preferable in shot-term predictions with respect to the STS-GCN model but it suffers from error accumulation in the long-term. Moreover, we trained a single (bigger) model to perform all the tasks (short-term and long-term) while the authors of STS-GCN train different models for each task, resulting in a more compact model. To overcome the issue of long-term errors accumulation we tried corrupting the inputs with a random walk noise but it did not work as expected. We believe it would be useful to further analyze and obtain statistics about the prediction errors in order to build a better model for the noise input.

Role of each member

Francesco Bonanni

- conversion[3] of the code used by Sanchez-Gonzalez et al. (2020) from Tensorflow to Pytorch.
- adaptation of the same code to our problem: deletion of unnecessary features such as boundaries and external fields and integration of the same code with the functions written by Sofianos et al. (2021).
- adaptation of the already existing visualization functions and implementation of code for creating all gifs, plots, figures and tables (except hyperparameter tuning).
- report: *Abstract*, *Introduction* and *Proposed method explained*

Eleonora Turchetti

- hyperparameter tuning, implementation of the random-walk noise and experimental tests with noise
- report: *Dataset and metrics*, *Experimental results and benchmark*

Vincenzo Romito and Theo Courbis

- hyperparameter tuning and plot of the best configuration
- report: *Related work* and *Conclusions*

[3] taking inspiration from https://github.com/geoelements/gns

References

Emad Barsoum, John Kender, and Zicheng Liu. 2017. HP-GAN: probabilistic 3d human motion prediction via GAN.

Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and Jose M. F. Moura. 2018. Adversarial geometry-aware human motion prediction.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. 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.

Ashesh Jain, Amir Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-rnn: Deep learning on spatio-temporal graphs.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. 2019. Learning trajectory dependencies for human motion prediction.

Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. 2020. Learning to simulate complex physics with graph networks.

Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. 2021. Space-time-separable graph convolutional network for pose forecasting.

Yongyi Tang, Lin Ma, Wei Liu, and Wei-Shi Zheng. 2018a. Long-term human motion prediction by modeling motion context and enhancing motion dynamic.

Yongyi Tang, Lin Ma, Wei Liu, and Wei-Shi Zheng. 2018b. Long-term human motion prediction by modeling motion context and enhancing motion dynamics.

Hao Zhou, Dongchun Ren, Xu Yang, Mingyu Fan, and Hai Huang. 2023. Csr: Cascade conditional variational auto encoder with socially-aware regression for pedestrian trajectory prediction. *Pattern Recognition*, 133:109030.

	Walking				Eating				Smoking				Discussion			
msec	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
STSCGN	10.7	16.9	29.1	32.9	6.8	11.3	22.6	25.4	7.2	11.6	22.3	25.8	9.8	16.8	33.4	40.2
Ours	**7.8**	**14.3**	**27.0**	**32.5**	**4.6**	**9.1**	**18.7**	**23.5**	**5.0**	**9.7**	**19.6**	**24.6**	**7.0**	**14.5**	**31.5**	**40.1**

	Directions				Greeting				Phoning				Posing				Purchases				Sitting			
msec	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
STSCGN	7.4	13.5	29.2	34.7	12.4	21.8	42.1	49.2	8.2	13.7	26.9	30.9	9.9	18.0	38.2	45.6	11.9	21.3	42.0	**48.7**	9.1	15.1	29.9	35.0
Ours	**5.0**	**11.0**	**25.8**	**33.3**	**9.6**	**19.3**	**40.4**	**50.4**	**5.9**	**11.4**	**23.7**	**30.0**	**6.4**	**13.9**	**32.3**	**42.1**	**9.4**	**18.9**	**39.2**	**48.7**	**6.6**	**12.4**	**25.4**	**32.0**

	Sitting Down				Taking Photo				Waiting				Walking Dog				Walking Together				Average			
msec	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
STSCGN	14.4	23.7	41.9	47.9	8.2	14.2	29.7	33.6	8.6	14.7	29.6	35.2	17.6	29.4	52.6	59.6	8.6	14.3	26.5	30.5	10.1	17.1	33.1	38.3
Ours	**11.5**	**20.2**	**37.5**	**45.9**	**6.0**	**11.5**	**24.2**	**30.5**	**5.6**	**11.4**	**25.2**	**32.3**	**14.1**	**26.2**	**48.2**	**57.7**	**6.4**	**12.8**	**26.1**	**32.5**	**7.4**	**14.4**	**29.7**	**37.1**

Table 1: MPJPE error in mm for short-term prediction of 3D joint positions on Human3.6M.

	Walking				Eating				Smoking				Discussion			
msec	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000
STSCGN	**40.6**	**45.0**	**48.0**	**51.8**	33.9	**40.2**	**46.2**	**52.4**	**33.6**	**39.6**	**45.4**	**50.0**	**53.4**	**63.6**	**72.3**	**78.8**
Ours	41.9	49.4	55.3	59.2	**32.9**	41.8	50.1	56.0	34.2	43.1	51.3	57.2	56.1	69.8	81.7	89.5

	Directions				Greeting				Phoning				Posing				Purchases				Sitting			
msec	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000
STSCGN	47.6	**56.5**	**64.5**	**71.0**	**64.8**	**76.3**	**85.5**	**91.6**	**41.8**	**51.1**	**59.3**	**66.1**	64.3	**79.3**	**94.5**	**106.4**	**63.7**	**74.9**	**86.2**	**93.5**	47.7	**57.0**	**67.4**	**75.2**
Ours	**47.1**	59.0	69.8	77.3	68.5	84.0	97.5	106.7	42.1	53.5	64.0	71.3	**61.6**	80.5	98.4	111.1	65.1	79.1	91.4	99.7	**45.0**	57.5	69.4	77.8

	Sitting Down				Taking Photo				Waiting				Walking Dog				Walking Together				Average			
msec	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000	560	720	880	1000
STSCGN	63.3	**73.9**	**86.2**	**94.3**	47.0	**57.4**	**67.2**	**76.9**	47.3	**56.8**	**66.1**	**72.0**	**74.7**	**85.7**	**96.2**	**102.6**	**38.9**	**44.0**	**48.2**	**51.1**	**50.8**	**60.1**	**68.9**	**75.6**
Ours	**61.6**	75.8	88.7	97.8	**43.0**	54.7	65.4	73.0	**46.1**	58.8	70.5	78.7	74.9	90.1	103.5	113.0	44.2	54.0	61.9	67.0	51.0	63.4	74.6	82.3

Table 2: MPJPE error in mm for long-term prediction of 3D joint positions on Human3.6M.

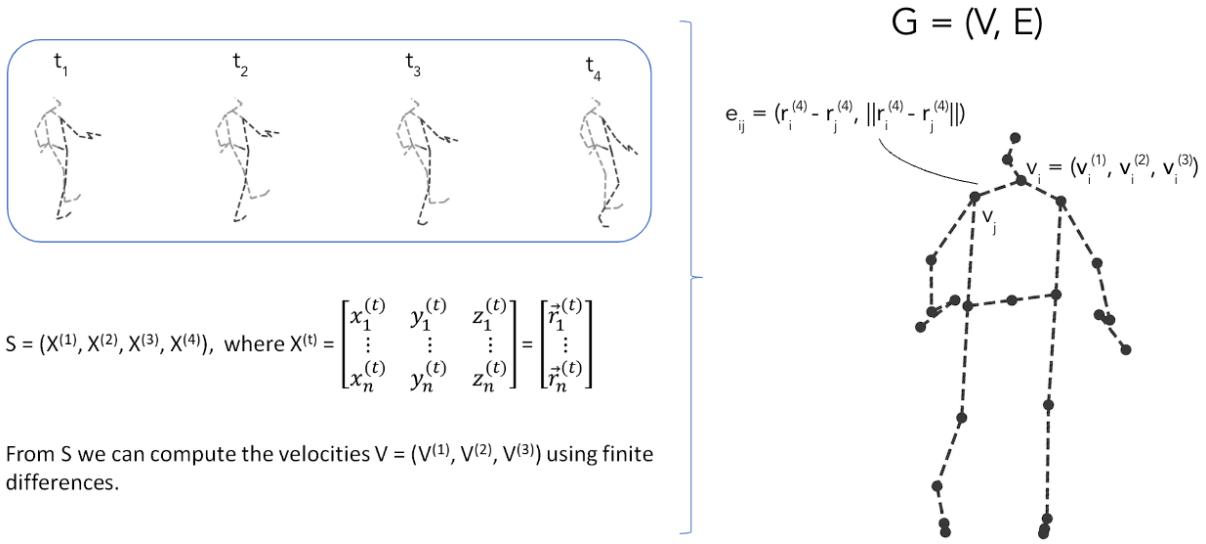

Figure 6: **Encoder**. The joints are 3D vectors $\vec{r}_j^{(t)}$ representing the position of joint j at time t. The motion history is a sequence S of T matrices $X^{(t)}$ for frames $t = 1, \ldots, T$. Each $X^{(t)}$ is an $n \times 3$ matrix where n is the total number of joints. The aim is to predict $X^{(T+1)}$. The information about the sequence is encoded in a graph G in the following way. Each node v_i contains the sequence of velocities computed using finite differences for $t = 1, \ldots, T-1$. Each edge $e_{i,j}$ contains the relative displacement between node v_i and node v_j at time T and the absolute distance $||v_i - v_j||$ at time T.