Advanced Machine Learning
3D Human Pose Forecasting

Theodoros Sofianos (1867968) Katsiaryna Zavadskaya (1847985)

June 2020

Abstract

Human motion prediction is one of the common computer vision problems, which have been typically approached using Recurrent Neural Networks (RNN). Inspired by some recent works, we aim at avoiding using RNNs and focus on CNNs and GCCNs for predicting sequential data in a single pass. Our aim is to create one unique model, capable of recognizing future human body poses, for different actions, subjects and time windows. We will evaluate our model on the classic Human3.6M dataset [2], and comparing all of our results with the current state of the art [3].

Introduction

The problem of human motion prediction has received a lot of attention from the research community in the recent years, because of it's various applications in modern science. We, as humans, can predict another person's movement, since we have an idea of the human body physical limitations and the way average person could perform an intentional movement. In order to create a system which is able to forecast human motion, the model learns from given 3D body joints coordinates for several time frames and predicts deterministically future body poses for the later time frames in terms of body joint 3D coordinates. The previous works on this topic can be separated into two big categories, the recurrent and the convolutional ones. Recurrent neural networks is a standard way to predict sequential data, however, they come with a bunch of problems, such as error accumulation and relatively big number of parameters. Inspired by more recent works that propose the use of CNNs, we implement a network, consisting of CNNs only. This comes with some advantages. Firstly, we perform the predictions in a single pass, thus 'winning' in running time and avoiding error accumulation. More concretely, we utilize the Graph Convolution Networks, which receive much interest from the research community in the recent years.

Related work

Recurrent architectures: The first attempts to approach the task of human motion prediction are for the most part recurrent ones. We studied some interesting papers, such as [1], however the use of RNNs is receiving some criticism in the recent years. Our aim is to develop a scalable and efficient architecture for our task at hand, so we avoided using RNN and focus on some more recent methods.

Graph CNNs: Graph Convolutions perform the convolution operation on a graph data, by forwarding the graph's adjacency matrix, after the convolution. The normalization of the adjacency matrix is essential, according to the authors [6]. A classic approach when using GCNN for human motion prediction is to represent the adjacency matrix as the kinematic tree [6]. This idea provides a solid basis for future works, but it has some limitations. There is not a standard way to represent the human motion through time with some adjacency matrix. In our case, the adjacency matrix represents connections between body joints, however it is expressed in terms of neighbouring body joints, instead of the kinematic tree.

Proposed method

Model input

Several steps were taken to implement model correctly. In order to train model we need two types of input: sequences of body joint coordinates and adjacency matrix.

Coordinates sequences: Each of the time frames is a matrix (V, D), where V is the body joints and D is the dimension of the coordinates of each body joint, which is equal to 3 in our case. Then we stack together multiple time frames and receive matrix (T, V, D), where T is the number of time frames. This way we receive a stacked sequence of the human body coordinates. Our decision was to use only 17 moving body joints instead of 32 available in the dataset. Moreover, during

training, we always use $T = 10$.

Adjacency matrix: The other input for the model is an adjacency matrix. For the each time frame adjacency matrix A represents presence or absence of a direct physical connection between body joints. Then we symmetrically normalize adjacency matrix A for each frame by multiplying it with diagonal node degree matrix. Lets denote normalized adjacency matrix as $\hat{A} = D^{-0.5} \times A \times D^{-0.5}$. Later for each input sequence, we stack the adjacency matrices of its T individual frames, thus creating the time dimension. Each body joint is connected to the same body joint for the T frames. Since the connectivity of the human body is always the same, we have used the same spatio-temporal adjacency matrix for all the sequences, which eases computations.

The 3D inputs are multiplied with normalized adjacency matrix during the forward pass, right after the convolution operation. The authors of [5] introduced a kernel function inside the adjacency matrix. In our case, we can not use their kernel function because of the differences in the nature of our tasks. Thus, we have decided to use one unique adjacency matrix, that describes the spatio-temporal connections between body joints and is sequence-indifferent. The advantage of our method is clearly the running time, since we do not have to compute a separate graph for each sequence.

Network architecture

The architecture we use was inspired by [5], but changed with the correspondence to our task. Most importantly, we have introduced a one-layered CNN that handles the embedding dimension,in order to get deterministic,rather than probabilistic, predictions for the future human poses.

Our model consists of the three networks:

1. Spatio-temporal Graph CNN (STG-CNN)
2. 1-Layer CNN
3. Time-Extrapolator CNN (TE-CNN)

Where the middle 1-Layer CNN is not part of original network idea.

Spatio-temporal Graph CNN performs spatio-temporal convolution operations on the graph representation of joints coordinates by mapping the original 3D body joints positions into an embedding of higher dimension. 1-Layer CNN is needed for taking us from the embedding dimension of the coordinates back to the original 3D dimension. Time-Extrapolator CNN maps the time dimension of the input sequence to the one of the output sequence. The layers of the STG-CNN and TE-CNN are connected with shortcuts and the convolution operations maintain the original dimensions.

When it comes to the output of the model, we get same as the input – sequences of the body joints coordinates, however number of the time frames of the input not needed to be equal to the time window of the output. The detailed view of the full model can be seen in the Figure (1).

Dataset and Benchmark

Following [3], we train our network on the Human3.6M dataset, which includes 3.6 million of 3D human poses and corresponding images, where 11 subjects perform 17 activities: discussion, smoking, taking photo, talking on the phone, etc. For a fair comparison, we adopt the dataset coordinates preprocessed by Martinez et al in [4]. Thus finally, we have 7 subjects in total. We use S11 for validation and S5 for testing, the rest are used for training. The number of actions performed by each of the subjects were reduced to 15.

In order to prepare data, we downsampled it from original 50 Hz to 25 Hz. Then we performed data split into subsets, the proportions can be seen the Table (3). This is done so we get comparable results with our baseline model.

Since, the bodies of different people are unequal in height and width, we decided to normalize input coordinates. To do that, we tried two types of normalization:

Mean-Std Normalization: We computed mean and std statistics of the training subset and applied them to validation and test subsets before passing them to the model.

Average body length Normalization: A body length of a subject was easily computed by exploiting graph structure of the human body and is equal to the sum of the all the graph edges lengths. The average was found among 5 training subjects and applied to validation and test subsets.

Empirically, we found out that the Mean-Std Normalization was working best for our task, however, improved final results only slightly, comparing to Average body length Normalization and also to not normalized coordinates.

As a benchmark we used [3] and Mean Per Joint Position Error function, measured in millimeters.

Experimental results

The best model

After number of experiments with deepness of the network, size of embedding dimension and types of optimizer, we found our best performing network with 5069 learnable parameters.
The architecture of the model is following:

- 3 layers of STG-CNN
- 1 layer of CNN
- 4 layers of TX-CNN

Hyperparameters:

- Embedding dimension = 20
- Optimizer = Adam (with default 0.001 learning rate)
- Number of training epochs = 50
- Batch size = 128

The final architecture schema is presented on the Figure (3), together with the loss (2).

Results

Since our aim is to create one unique system, during the learning process we measured performance on only one input and output time frame: T_in = T_out = 10. For this time window we outperformed current state-of-the-art [3]. However, the model is capable to make predictions not only for T_out = 10 (400 miliseconds), but for varying time window:

1. Short-term: 80, 160, 320, 400 milliseconds
2. Long-term: 560, 1000 milliseconds

The comparison of our results with the state-of-the-art is visible in the Tables (1) and (2) for short-term and long-term predictions, respectively.

Conclusions and Future work

In conclusion, we saw that a simple implementation of GCNN and CNNs performed great on predicting the future human body pose, for a specific time-frame window. We were curious to see if the same architecture can work for varying time windows, but our results showed that its not the case, at least for very different time windows, such as for 2 and 25 frames. Since our network is designed to have a relatively small number of parameters, it is unsure whether our network has the capacity to work for different time windows. Since we have also seen that the data normalization does not improve the model greatly, a way to increase the model capacity could be to use sequence-specific adjacency matrices, or even make the adjacency matrix learnable, so that additional info is passed at each input to our model. Something that stood out for us is that most of our visualizations of the predictions do not include 'illegal' body poses, ie poses that can not be performed because of the human anatomy and also the length of the body limbs is maintained, although we have not intervened to avoid these issues. We have noticed that our model, after some number of epochs, starts to predict more 'realistic' body poses, something that is not necessarily depicted into the loss. We believe that if the model is designed carefully, there is no need to add manually restrictions for illegal body poses and lengths of human limbs, since the model could learn them on its own. Thus, the focus should be just on minimizing the loss.

References

[1] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[2] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[3] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9489–9497, 2019.

[4] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[5] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for

human trajectory prediction. *arXiv preprint arXiv:2002.11927*, 2020.

[6] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Figures and Tables

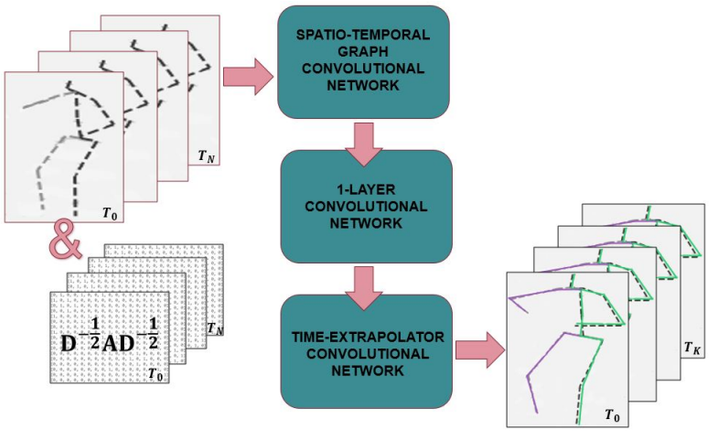

Figure 1: Schema of the model

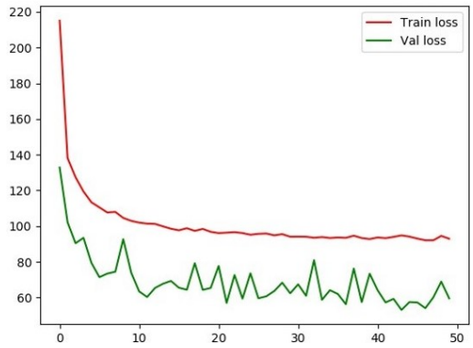

Figure 2: Loss of the best model

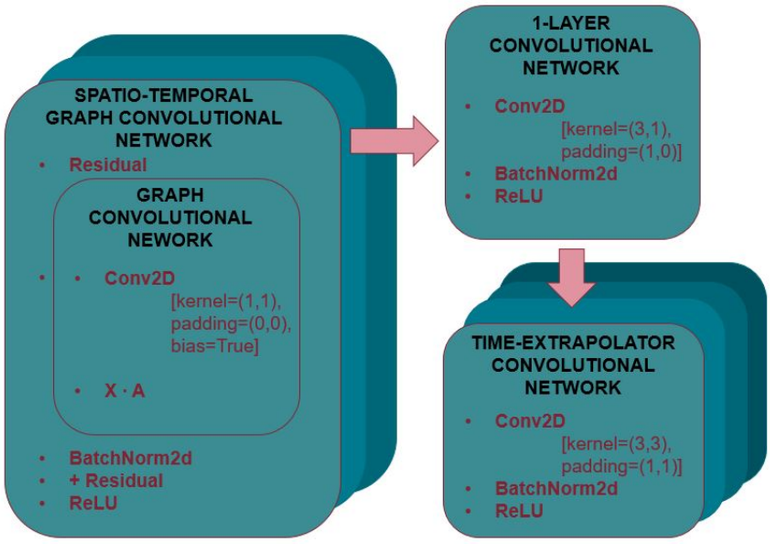

Figure 3: Network architecture

	Walking				Eating				Smoking				Discussion			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
Baseline 3D	8.9	15.7	29.2	33.4	8.8	18.9	39.4	47.2	7.8	14.9	25.3	28.7	9.8	22.1	39.6	44.1
Ours	**43.9**	**60.6**	**62.0**	**72.5**	**39.8**	**38.6**	**40.9**	**49.3**	**40.9**	**41.7**	**43.8**	**47.6**	**44.3**	**50.0**	**57.1**	**55.2**

	Directions				Greeting				Phoning				Posing				Purchases				Sitting			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
Baseline 3D	12.6	24.4	48.2	58.4	14.5	30.5	74.2	89.0	11.5	20.2	37.9	43.2	9.4	23.9	66.2	82.9	19.6	38.5	64.4	72.2	10.7	24.6	50.6	62.0
Ours	**42.5**	**42.5**	**47.1**	**43.3**	**45.2**	**56.7**	**70.6**	**78.9**	**40.8**	**42.7**	**45.6**	**52.8**	**46.2**	**44.4**	**58.0**	**58.5**	**42.8**	**45.7**	**57.2**	**60.8**	**38.7**	**34.7**	**40.8**	**46.1**

	Sitting Down				Taking Photo				Waiting				Walking Dog				Walking Together				Average			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
Baseline 3D	11.4	27.6	56.4	67.6	6.8	15.2	38.2	49.6	9.5	22.0	57.5	73.9	32.2	58.0	102.2	122.7	8.9	18.4	35.3	44.3	12.1	25.0	51.0	61.3
Ours	**39.3**	**48.5**	**55.3**	**59.5**	**52.9**	**47.5**	**62.2**	**58.6**	**43.5**	**42.7**	**50.2**	**52.8**	**47.2**	**65.9**	**78.1**	**94.0**	**47.2**	**56.7**	**63.6**	**67.0**	**43.1**	**47.3**	**54.4**	**57.9**

Table 1: Short-term prediction compared with state-of-the-art

	Walking		Eating		Smoking		Discussion		Average	
milliseconds	560	1000	560	1000	560	1000	560	1000	560	1000
Baseline 3D	42.3	51.3	56.5	68.6	32.2	60.5	70.5	103.5	50.4	71.0
Ours	**94.1**	**110.3**	**58.1**	**82.7**	**61.2**	**89.8**	**83.6**	**116.8**	**74.25**	**103.3**

Table 2: Long-term prediction compared with state-of-the-art

Subset	Number of frames
Train	370.276
Validation	56.419
Test	99.079
Total	525.774

Table 3: Data Train, Validation and Test subsets