MSR-GCN: Multi-Scale Residual Graph Convolution Networks for Human Motion Prediction
— Supplementary Material —

Lingwei Dang[1], Yongwei Nie[1*], Chengjiang Long[2], Qing Zhang[3] and Guiqing Li[1]

[1]School of Computer Science and Engineering, South China University of Technology, China
[2]JD Finance America Corporation, USA
[3]School of Computer Science and Engineering, Sun Yat-sen University, China

Abstract

In this supplementary material, we provide more information that cannot be included in the paper due to the space limit.

1. Loss Function

We use ℓ_2 loss to optimize MSR-GCN. Let the j^{th} joint position in the t^{th} frame at s^{th} scale be $\hat{p}^s_{j,t}$, and the corresponding ground-truth be $p^s_{j,t}$, then the loss function for N training pose sequences each having J^s joints and T frames is written as

$$\mathcal{L}^s = \frac{1}{N \times J^s \times T} \sum_{n=1}^{N} \sum_{j=1}^{J^s} \sum_{t=1}^{T} \left\| \hat{p}^s_{j,t} - p^s_{j,t} \right\|_2. \quad (1)$$

The above loss is calculated at all S scales and added up to optimize the proposed model, that is,

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} \sum_{s=1}^{S} \lambda^s \mathcal{L}^s, \quad (2)$$

where $\mathcal{P}$ indicates network parameters, and λ denotes hyper parameters and we set them as 1 for all scales.

2. Model Structure

The detailed MSR-GCN model structure is shown in Table 1. As mentioned in the paper, our proposed approach is composed of three kinds of GCNs, called "Start GCNs", "Descending (D0-D3)/Ascending (A0-A3) GCNs", and "End GCNs (E0-E3)".

Table 1. Detailed architecture of MSR-GCN.

Module	Layers	Output Size	Operations
Start GCN	GCL	66 × 64	GCL, A(66 × 66), W(35 × 64)
	GCN	66 × 64	res-GCN with 2-layer GCLs A(66 × 66), W(64 × 64)
D0	GCNs	66 × 64	3 × res-GCN each has 2-layer GCLs A(66 × 66), W(64 × 64)
Downsampling 0	Linear1	36 × 64	linear transformation, W(66 × 36)
	Linear2	36 × 128	linear transformation,W(64 × 128)
D1	GCNs	36 × 128	3 × res-GCN each has 2-layer GCLs A(36 × 36), W(128 × 128)
Downsampling 1	Linear1	21 × 128	linear transformation, W(36 × 21)
	Linear2	21 × 256	linear transformation,W(128 × 256)
D2	GCNs	21 × 256	3 × res-GCN each has 2-layer GCLs A(21 × 21), W(256 × 256)
Downsampling 2	Linear1	12 × 256	linear transformation, W(21 × 12)
	Linear2	12 × 512	linear transformation,W(256 × 512)
D3	GCNs	12 × 512	3 × res-GCN each has 2-layer GCLs A(12 × 12), W(512 × 512)
A3	GCNs	12 × 512	3 × res-GCN each has 2-layer GCLs A(12 × 12), W(512 × 512)
Upsampling 2	Linear1	21 × 512	linear transformation, W(12 × 21)
	Linear2	21 × 256	linear transformation,W(512 × 256)
A2	GCNs	21 × 256	3 × res-GCN each has 2-layer GCLs A(21 × 21), W(256 × 256)
Upsampling 1	Linear1	36 × 256	linear transformation, W(21 × 36)
	Linear2	36 × 128	linear transformation,W(256 × 128)
A1	GCNs	36 × 128	3 × res-GCN each has 2-layer GCLs A(36 × 36), W(128 × 128)
Upsampling 0	Linear1	66 × 128	linear transformation, W(36 × 66)
	Linear2	66 × 64	linear transformation,W(128 × 64)
A0	GCNs	66 × 64	3 × res-GCN each has 2-layer GCLs A(66 × 66), W(64 × 64)
E0	GCN	66 × 64	res-GCN with 2-layer GCLs A(66 × 66), W(64 × 64)
	GCL	66 × 35	GCL, A(66 × 66), W(64 × 35)
E1	GCN	36 × 128	res-GCN with 2-layer GCLs A(36 × 36), W(128 × 128)
	GCL	36 × 35	GCL, A(36 × 36), W(128 × 35)
E2	GCN	21 × 256	res-GCN with 2-layer GCLs A(21 × 21), W(256 × 256)
	GCL	21 × 35	GCL, A(21 × 21), W(256 × 35)
E3	GCN	12 × 512	res-GCN with 2-layer GCLs A(12 × 12), W(512 × 512)
	GCL	12 × 35	GCL, A(12 × 12), W(512 × 35)

The most basic building block is the Graph Convolution Layer (GCL), which consists of a graph convolution layer, a batch normalization layer, a tanh activation layer, and a

*Corresponding author: nieyongwei@scut.edu.cn

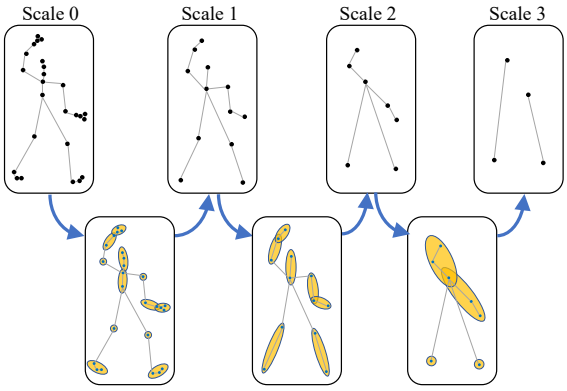

Figure 1. The default grouping manner of 25-12-7-4 for the CMU Mocap dataset.

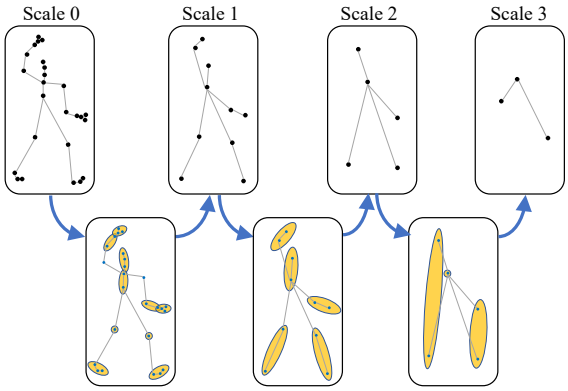

Figure 2. The manually specified 25-10-5-3 grouping manner for the CMU Mocap dataset.

dropout layer (with rate 0.1). A graph convolution layer has an adjacency matrix A and parameters W.

Each GCN is composed of 2 GCLs. The size of A and W of these GCLs are shown in the table. We use linear layers for downsampling and upsampling. The sizes of the parameters in these linear layers are also shown in the table. In the third column of the table, we give the output size of the corresponding layer. Please refer to the source code at https://github.com/Droliven/MSRGCN for more information.

3. Different Multi-Scale Grouping Manners

The default grouping manner for CMU can be found in Figure 1 in which there are 25 joints at the finest level and 12, 7, 4 joints in the subsequent coarser levels. We also trained MSR-GCN on CMU with other grouping manners, including three random grouping manners under the 25-12-7-4 manner, and the "manually specified 25-10-5-3" manner as shown in Figure 2. The performance of MSR-GCN under different grouping manners can be found in the paper.

Table 2. Comparison of average prediction error with Traj-GCN [33] using error bar

	H3.6M	CMU
Traj-GCN [33]	59.93 ± 0.91	40.56 ± 0.51
Ours	**58.37± 0.43**	**37.52 ± 0.48**

Table 3. Comparison with Traj-GCN at different forecast times.

Time (ms)	80	160	320	40	560	1000
Human3.6M	0.56	0.51	0.64	0.58	0.46	0.09
CMU	1.22	2.19	3.98	2.84	2.40	3.23

Table 4. Average prediction errors using the evaluation method of [33].

	H3.6M		CMU	
	short-term	long-term	short-term	long-term
Traj-GCN [33]	37.35	59.02	29.13	45.06
Ours	**36.36**	**57.84**	**24.81**	**40.81**

4. More Results

Comparison with Traj-GCN using error bar. We have trained our method and Traj-GCN [33] five times with random seeds in order to compare their performance more thoroughly. As shown in Table 2, the average prediction errors of our method are 58.37±0.43 and 37.52±0.48 on the datasets of Human3.6M and CMU. In comparison, [33] reports higher predictor errors and larger variances than our method, which are 59.93±0.91 on the Human3.6M and 40.56±0.50 on the CMU dataset respectively.

Comparison with Traj-GCN at different forecast times. We also compared MSR-GCN and Traj-GCN at different forecast times. As verified in Table 3, our method performs better than Traj-GCN in handling challenging long-term motion prediction.

Comparison using the evaluation method of [33]. In [33], the performance is evaluated on randomly selected 8 samples per action. The average prediction errors using the same evaluation method as [33] are shown in Table 4. As can be seen, MSR-GCN also outperforms Traj-GCN.