Supplementary Materials for ActFormer: A GAN-based Transformer towards General Action-Conditioned 3D Human Motion Generation

Liang Xu[2,3*] Ziyang Song[5*] Dongliang Wang[1,6] Jing Su[1] Zhicheng Fang[1] Chenjing Ding[1]
Weihao Gan[7] Yichao Yan[2] Xin Jin[3,4] Xiaokang Yang[2] Wenjun Zeng[3,4] Wei Wu[1,6]
[1]SenseTime Research [2]Shanghai Jiao Tong University [3]Eastern Institute of Technology, Ningbo
[4]Ningbo Institute of Digital Twin [5]The Hong Kong Polytechnic University [6]Shanghai AI Laboratory
[7]Mashang Consumer Finance Co., Ltd.

We provide the details about our network architecture (Appendix A), additional implementation details (Appendix B), additional qualitative results (Appendix C) and additional samples from our GTA Combat dataset (Appendix D).

A. Network architecture

Tab. 1 presents the network architecture we designed on the NTU-2P dataset. In the ActFormer generator, Gaussian Linear Error Units (GELU) [3] activations are used for Transformer encoders. In the GCN discriminator, Leaky Rectified Linear Units (LeakyReLU) [4] activations are adopted. This network architecture can also be applied to other datasets. The dimension of inputs/outputs may slightly vary due to changes in pose representations or the number of persons. On single-person action datasets, I-Former modules in the ActFormer generator are not needed.

I-Former and T-Former. Here we give a detailed illustration of the I-Former and T-Former modules. In both modules, given $P \cdot (T+1)$ tokens (corresponding to a T-frame, P-person motion sequence) as input, we represent the current token embedding by $F_t^p \in \mathbb{R}^{C_{in}}$, where $t \in [1, ..., T+1]$ and $p \in [1, ..., P]$. First, we apply linear transformations to each token to compute {query, key, value} vectors $q_t^p, k_t^p, v_t^p \in \mathbb{R}^d$ as,

$$q_t^p = W_q F_t^p, k_t^p = W_k F_t^p, v_t^p = W_v F_t^p, \tag{1}$$

where the learnable transform parameters $W_q, W_k, W_v \in \mathbb{R}^{d \times C_{in}}$ are shared among all tokens. Then we discuss subsequent steps in I-Former and T-Former, respectively.

In the I-Former module, we apply self-attention among persons in every single frame independently by query-key dot product and the weighted sum of values as below,

$$w_t^{p,p'} = q_t^p \cdot k_t^{p'}, \forall t \in [1, ..., T+1], \tag{2}$$

$$G_t^p = \sum_{p'} softmax_{p'}(\frac{w_t^{p,p'}}{\sqrt{d}}) v_t^{p'}. \tag{3}$$

Now we got the transformed token embedding $G_t^p \in \mathbb{R}^d$ by the I-Former.

In the T-Former, self-attention is performed among frames of each person separately. To be specific, the T-Former transforms the token embedding into $H_t^p \in \mathbb{R}^d$ as in the following,

$$w_{t,t'}^p = q_t^p \cdot k_{t'}^p, \forall p \in [1, ..., P], \tag{4}$$

$$H_t^p = \sum_{t'} softmax_{t'}(\frac{w_{t,t'}^p}{\sqrt{d}}) v_{t'}^p. \tag{5}$$

Interaction modeling in GCN discriminator. Here we illustrate in detail how concatenation models multi-person interactions in our GCN discriminator. We drop the temporal dimension T for simplicity and observe a K-node graph of the spatial human skeleton. Each node contains a vector $f^k \in \mathbb{R}^{P \cdot D}$ concatenated by P persons. We focus on a specific node with K_N neighbors in the skeleton, as shown in Fig. 1(a) ($K_N = 4$ including the center node itself, $P = 2, D = 3$ in this example). A spatial GraphConv. kernel w aggregates information from all the neighboring nodes and produce an output feature g for the center node, i.e.,

$$g = \sum_{k=1}^{K_N} \sum_{l=1}^{P \cdot D} w_l^k \cdot f_l^k. \tag{6}$$

If we split multiple persons' motion in the same node, i.e., transform $f^k \in \mathbb{R}^{P \cdot D}$ into $f^k \in \mathbb{R}^{P \times D}$, Eq. (6) can be

*Denotes equal contribution. Work done when Liang and Ziyang were at SenseTime. Corresponding authors: Dongliang Wang (wangdongliang@senseauto.com), Yichao Yan (yanyichao@sjtu.edu.cn) and Xin Jin (jinxin@eias.ac.cn).

Network	Architecture	Params.
Generator	(Input projection): Linear(Ch=(120, 200)) (Class embedding): Linear(Ch=(26, 200)) (I-Former): Transformer(Ch=200, n_heads=8) (T-Former): Transformer(Ch=200, n_heads=8) (I-Former): Transformer(Ch=200, n_heads=8) (T-Former): Transformer(Ch=200, n_heads=8) (Output projection): Linear(Ch=(200, 75))	1,359,675

Discriminator	GraphConv.(Ch=(150, 32), K=(2, 4), G=(25, 25))
	GraphConv.(Ch=(32, 64), K=(2, 4), G=(25, 11))
	GraphConv.(Ch=(64, 128), K=(2, 4), G=(11, 5))
	GraphConv.(Ch=(128, 256), K=(2, 4), G=(5, 5))
	GraphConv.(Ch=(256, 512), K=(5, 4), G=(5, 1))
	(Class embedding): Linear(in_ch=26, out_ch=512)
	(Output projection): Linear(in_ch=512, out_ch=1)
		5,174,593

Table 1. **Network architecture on NTU-2P dataset.** Params. is short for the *number of parameters*. The tuple for *Ch (channel)* denotes input/output channels of the layer. In GraphConv., the tuple for *K (kernel)* denotes spatial/temporal kernel sizes of the graph convolution layer. The tuple for *G (graph)* represents the number of nodes (joints) in the input/output spatial skeleton graph.

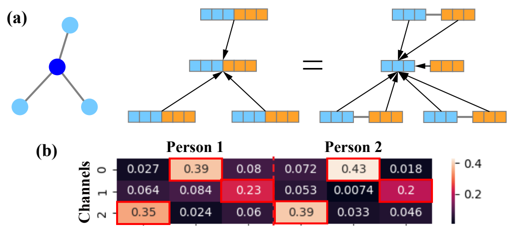

Figure 1. **Illustration about interaction modeling in GCN discriminator.** We make a theoretical analysis about (a) how concatenation enforces information aggregation from multiple persons. We also check (b) the weights of a trained discriminator to verify whether information from multiple persons is aggregated in practice.

rewritten as

$$g = \sum_{k=1}^{K_N} \sum_{l=1}^{P \cdot D} w_l^k \cdot f_l^k = \sum_{k=1}^{K_N} \sum_{\boldsymbol{p=1}}^{\boldsymbol{P}} \sum_{d=1}^{D} w_{pd}^k \cdot f_{pd}^k. \quad (7)$$

As visualized in Fig. 1(a), the output center node feature g aggregates information from all the K_N neighboring joints of all the P persons.

Besides theoretical analysis, we also check absolute values of trained weights from our GCN discriminator's 1st layer. As shown in Fig. 1(b), each output node feature indeed gets contributions from multiple persons. The information of different persons has been entangled since then, and human interaction modeling naturally exists in the following network layers.

B. Additional implementation details

Pose representations. On NTU RGB+D 120 and GTA Combat, local body poses are represented by normalized limb vectors instead of raw joint coordinates. On NTU-13 and BABEL, the continuous 6D representations [6] are employed to replace axis-angle joint rotations from SMPL body models.

Library credits. Our approach is implemented based on PyTorch [5]. For comparison to baseline methods, we use the official implementations of Action2Motion [1] and ACTOR [2].

Training. In the ActFormer's training, we adopt the Adam optimizer with betas (0, 0.999) and learning rate 0.0002 for both the generator and discriminator. The batch size is set to 64. During training, every time the discriminator is trained 4 times, the generator will be trained once. On each dataset, we train all models (including baseline methods) for 500 epochs.

In the training of the action recognition model, we adopt the SGD optimizer, with an initial learning rate of 0.1 decayed by 0.1 in the 10th and 50th epoch, respectively. The model is trained for 80 epochs on each dataset, and the batch size is set to 64.

Evaluation. We use each model to generate a sample set with 100 sequences per action category for evaluation. All motion sequences in our experiments hold 60 frames.

C. Additional qualitative results

Fig. 2 displays more generated motions by our approach. Compared to qualitative results in the main paper, we show more samples per action for multi-person actions to indicate that diversity also exists in our multi-person motion generation.

D. Additional GTA Combat samples

We provide more samples from our GTA Combat dataset. As shown in Fig. 3, by randomly combining interactive relationships and combat actions, the synthetic motions in GTA Combat present sufficient diversity. From dynamic clips, we see both combat actions of attackers and triggered reactions of attacked ones look natural.

References

[1] Action2motion official code implementation. `https://github.com/EricGuo5513/action-to-motion`.

[2] ACTOR official code implementation. `https://github.com/Mathux/ACTOR`.

[3] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[4] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

[6] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753. Computer Vision Foundation / IEEE, 2019.

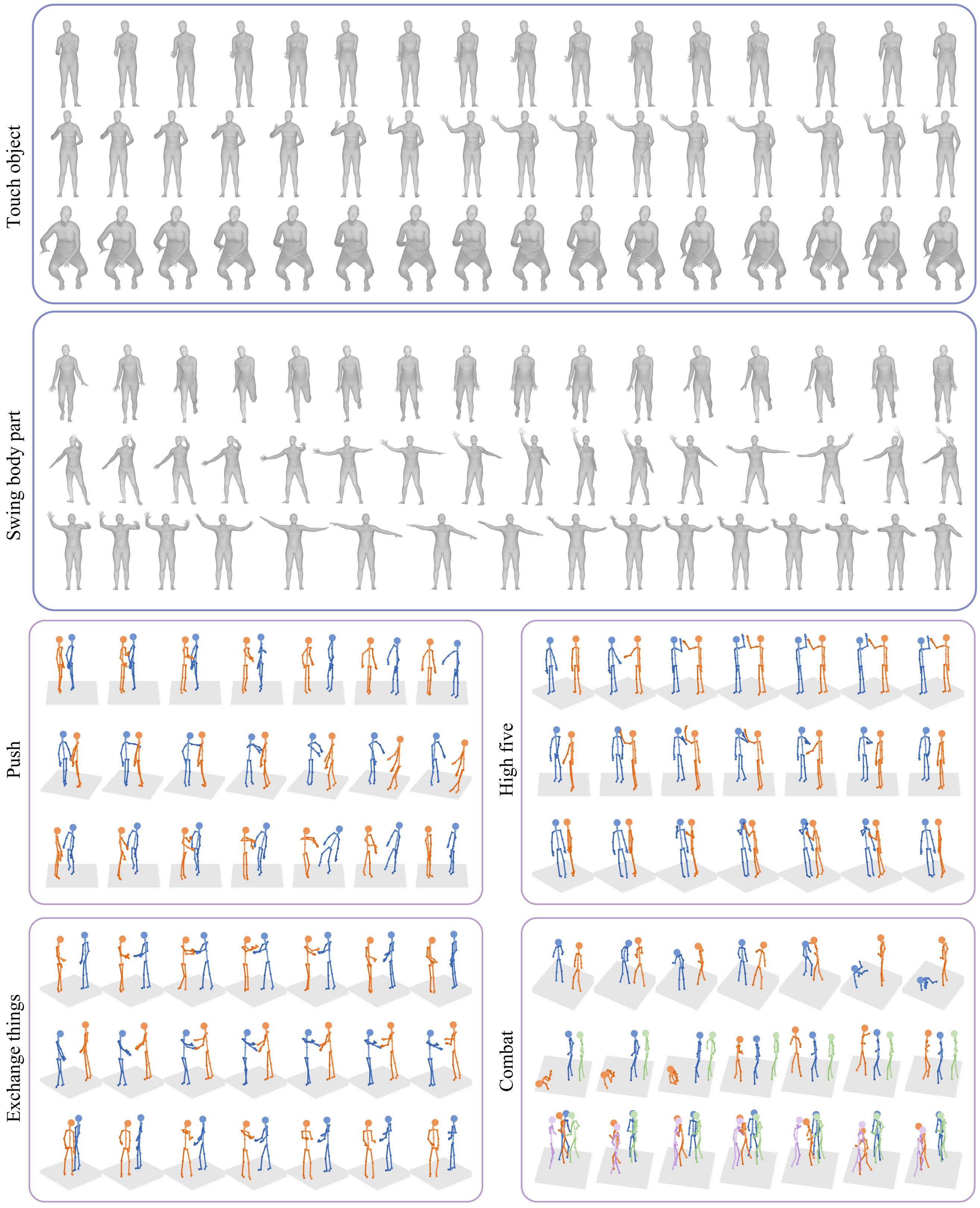

Figure 2. **Additional qualitative results.** We generate "Touch object" and "Swing body part" actions from BABEL, "Push" "High five" and "Exchange things" actions from NTU-2P, and "Combat" actions from GTA Combat.

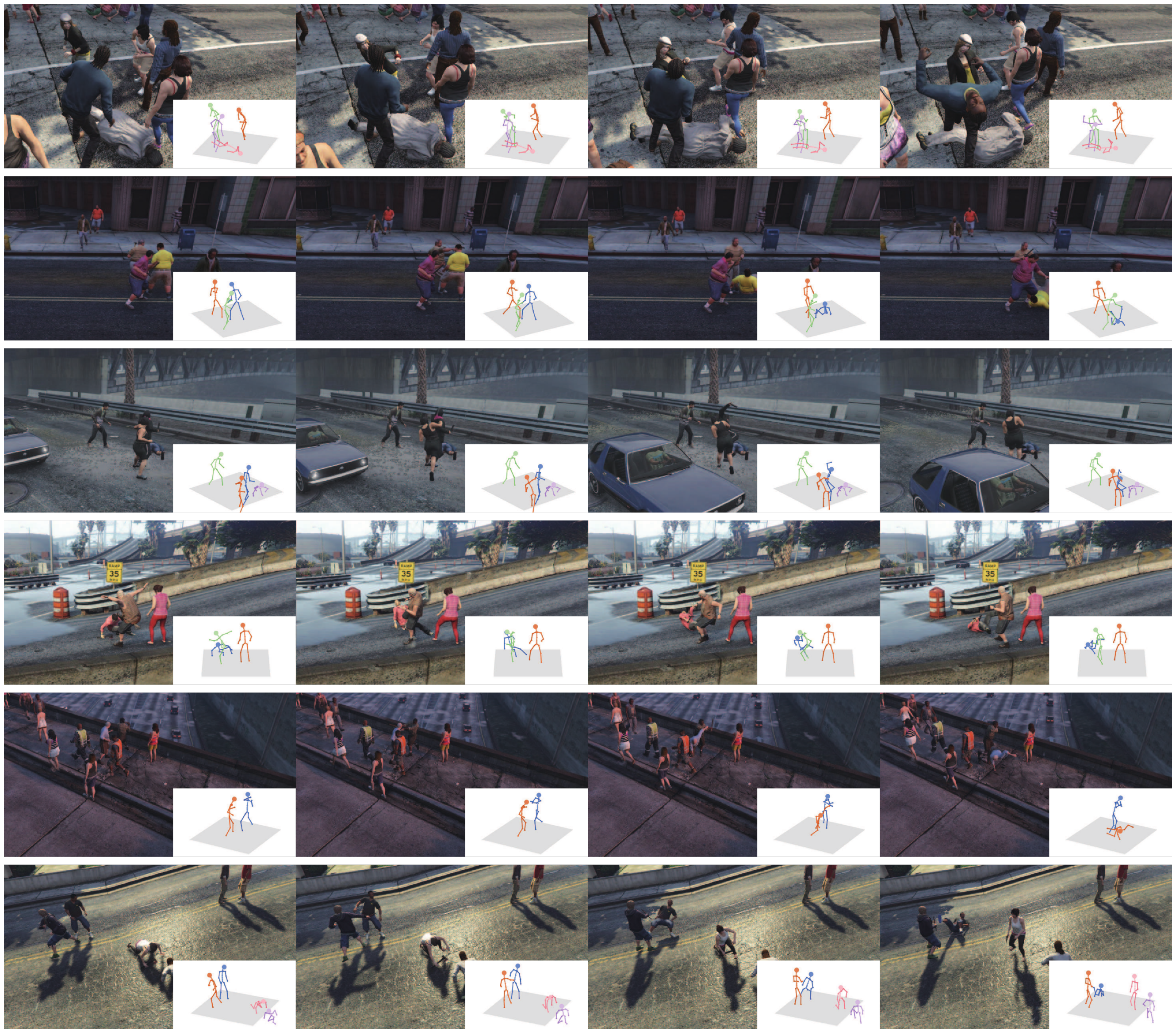

Figure 3. **Additional samples from GTA Combat dataset.**