LLMs are Good Action Recognizers

Supplementary Material

1. Additional Ablation Studies

We here conduct more ablation experiments on the X-Set protocol of the NTU RGB+D 120 dataset.

Impact of using different backbones for LLaMA. In the main experiments, we use the large language LLaMA with LLaMA-13B as its backbone. Here, to verify the generality of our framework, we also test using the smaller LLaMA-7B as the backbone. In Tab. 1, we report the overall accuracy, and the average runtime per action sample (when run in batch). As shown in Tab. 1, when a smaller backbone is used for the large language model, our framework can also achieve good performance. This shows the generality of our framework. Despite this, we also observe that LLaMA-13B with a larger model architecture can further improve the performance of our framework.

Table 1. Evaluation on using different backbones for LLaMA.

Method	Accuracy	Runtime(s)
LLaMA-7B	90.6	0.21
LLaMA-13B	91.5	0.39

Impact of the number of tokens U. In our framework, we set the number of tokens U in the codebook to be 512. Here we evaluate other choices of U, and report the results in Tab. 2. As shown, the model performance improves with U when U is smaller than 512, becomes stabilized after U reaches 512, and drops when U is set to be a very large number (16384). This might be because, while building the codebook with more tokens can lead the quantization process to be performed more accurately, involving too many tokens in the codebook may lead repeated features to be learned by different tokens and thus lead to training difficulties. Taking the above into consideration, we set U to 512 in our experiments.

Table 2. Evaluation on the number of tokens U.

Method	Accuracy
$U = 128$	90.1
$U = 256$	91.1
$U = 512$	91.5
$U = 1024$	91.5
$U = 16384$	90.4

Impact of the curvature c. In our framework, during constructing the hyperbolic codebook C_H, we set the curvature c for the Poincaré ball model to be 1. Here we also assess other choices of c ranging from 0.1 to 10. As shown in Tab. 3, our framework gets optimal performance when c is set to 1 or 5, and $c = 1$ is used in our experiments. Besides, with different choices of c from 0.1 to 10, our framework outperforms the previous state-of-the-art method consistently. This demonstrates the robustness of our framework to this hyperparameter.

Table 3. Evaluation on the curvature c.

Method	Accuracy
$c = 0.1$	90.8
$c = 0.5$	91.3
$c = 1$	91.5
$c = 5$	91.5
$c = 10$	91.4

Impact of the batch size B. In our framework, during the training process of the action-based VQ-VAE model, we set the batch size B to be 256. Here to evaluate the impact of the batch size B, we evaluate different choices of B and report the results in Tab. 4. As shown, the model performance increases with the batch size B consistently when B is smaller than 256, and becomes stabilized after B reaches 256. Thus, we set B to 256 in our experiments.

Table 4. Evaluation on the batch size B.

Method	Accuracy
$B = 64$	91.0
$B = 128$	91.4
$B = 256$	91.5
$B = 512$	91.5

Impact of regularizing the token usage frequency of "action sentences" to follow different distributions. In our framework, we aim to regularize the set of "action sentences" to follow Zipf's law. To achieve this, we incorporate L_{Zipf} into our total loss function to minimize the difference between the distribution D_{freq} representing the token usage frequency of "action sentences" and the Zipf distribution D_{Zipf} (**regularizing using the Zipf distribution**). Here to further evaluate the efficacy of L_{Zipf}, we test two variants. In the first variant (**no regularization**), we remove L_{Zipf} from the learning process. In the second variant (**regularizing using the uniform distribution**), instead of minimizing the difference between D_{freq} and D_{Zipf}, we formulate the loss function to minimize the difference between D_{freq} and the uniform distribution $D_{uniform}$. As shown in Tab. 5, our framework outperforms both variants. This further shows the efficacy of minimizing the difference between D_{freq} and D_{Zipf}, which regularizes the set of "action sentences" to follow Zipf's law.

Table 5. Evaluation on regularizing the token usage frequency of "action sentences" to follow different distributions.

Method	Accuracy
No regularization	89.9
Regularizing using the uniform distribution	90.6
Regularizing using the Zipf distribution	91.5

2. Additional Details about the Architecture of the Action-based VQ-VAE Model

With respect to the architecture of the action-based VQ-VAE model in our framework, inspired by previous VQ-VAE works [2, 5, 6], both the encoder and the decoder of our action-based VQ-VAE model consist of 1D convolution layers together with ReLU activation functions and the skip connection design. Besides, inspired by previous action recognition works [1, 4], we also involve the encoder of our model with graph convolution layers and 3D-CNN blocks to better represent the input action signals (i.e., the skeleton sequences). Furthermore, we involve the decoder of our model with up-sampling layers to facilitate action signal reconstruction. We also follow [2, 5] to use exponential moving average and codebook reset in the training process of our action-based VQ-VAE model.

3. Additional Details about $Corr(t^b)$

In our framework, we encourage the "action sentences" to follow the context-sensitivity bias and use more correlated tokens via $L_{context}$. In Sec. 3.1 in the main paper, during calculating $L_{context}$, we define a $Corr(\cdot)$ operation and take t^b as its input to calculate the number of correlated tokens used in discretizing $f^b_{1:W}$. Remind that inspired by [3], we regard each token c_u in the first half of the codebook (i.e., $u \in \{1, ..., \frac{U}{2}\}$) and the token $c_{u+\frac{U}{2}}$ in the second half of the codebook to be a pair of correlated tokens. Besides, t^b is a vector with length U, and the value of the u-th element of t^b represents the number of times token c_u is used in discretizing $f^b_{1:W}$.

We then discuss how we perform $Corr(t^b)$ in more detail as follows. Specifically, $Corr(t^b)$ is performed in the following three steps: (1) Firstly, for every element of t^b in its first half (i.e., $u \in \{1, ..., \frac{U}{2}\}$), we perform min pooling with size 2 over it and its pair element in the second half of t^b (e.g., a min pooling operation is then performed over the u-th element of t^b and the $(u + \frac{U}{2})$-th element of t^b). (2) We then store these $\frac{U}{2}$ min pooling outputs sequentially in a vector $t^b_{pooling}$. Note that by doing so, the value of the u-th element of $t^b_{pooling}$ represents the number of times both the token c_u and the token $c_{u+\frac{U}{2}}$ in the pair are used in discretizing $f^b_{1:W}$. (3) Finally, as the total number of correlated tokens used is two times to total number of pairs used, we calculate the total number of correlated tokens used in discretizing $f^b_{1:W}$ as $Corr(t^b) = 2 \times sum(t^b_{pooling})$.

4. Additional Details about the Evaluation on Unseen Activity Classes

In Sec. 4.4 in the main paper, to evaluate our framework on unseen activity classes, we use a new evaluation protocol. Specifically, under this protocol, 3 classes from the NTU RGB+D dataset are randomly selected to form the unseen class list during each time of evaluation, and this evaluation is performed for five times. Below, we list the unseen classes we randomly select in each time of evaluation: (1) ["brushing hair", "kicking something", "jump up"], (2) ["wear a shoe", "make ok sign", "shake fist"], (3) ["thumb up", "reach into pocket", "take off a hat/cap"], (4) ["bounce ball", "cutting nails", "point finger at the other person"], and (5) ["wear jacket", "apply cream on face", "put on headphone"].

5. Illustration on the Testing Process

In Fig. 1, we demonstrate the testing process of our framework. As shown, during the testing process of our framework, we first project the input action signal into its corresponding "action sentence" of discrete tokens. We then complete the instruction to the large language model by fill in the [tokens] part of the sentence based on the derived "action sentence". Finally, we pass the instruction to the large language model to perform action recognition.

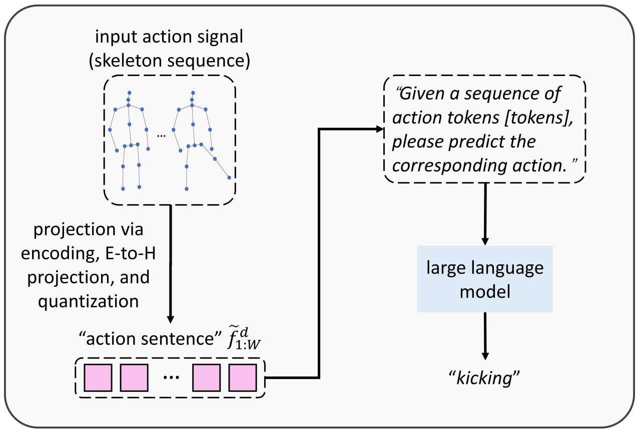
input action signal (skeleton sequence)
projection via encoding, E-to-H projection, and quantization
"action sentence" $\tilde{f}^d_{1:W}$
"Given a sequence of action tokens [tokens], please predict the corresponding action."
large language model
"kicking"

Figure 1. Illustration on the testing process of our framework.

References

[1] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2969–2978, 2022. 1

[2] Evonne Ng, Sanjay Subramanian, Dan Klein, Angjoo Kanazawa, Trevor Darrell, and Shiry Ginosar. Can language models learn to listen? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10083–10093, 2023. 1, 2

[3] Isabel Papadimitriou and Dan Jurafsky. Pretrain on just structure: Understanding linguistic inductive biases using transfer learning. *arXiv preprint arXiv:2304.13060*, 2023. 2

[4] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 1

[5] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. *arXiv preprint arXiv:2301.06052*, 2023. 1, 2

[6] Yaqi Zhang, Di Huang, Bin Liu, Shixiang Tang, Yan Lu, Lu Chen, Lei Bai, Qi Chu, Nenghai Yu, and Wanli Ouyang. Motiongpt: Finetuned llms are general-purpose motion generators. *arXiv preprint arXiv:2306.10900*, 2023. 1