

# HuMoCon: Concept Discovery for Human Motion Understanding

## Supplementary Material

### 6. Implementation Details

We resize all video frames to a resolution of  $320 \times 256$ . Limited by GPU memory, we only take 8 key frames for each video following MotionLLM [5]. And we represent the motion following Humanml3D [22].

During the encoder pre-training stage, we set the mask ratio  $\lambda$  to 0.75 and configure the loss weights  $\lambda^{dis}$ ,  $\lambda^{act}$  and  $\lambda^{align}$  to 0.3, 0.1, and 0.1, respectively. To stabilize the training of VQ-VAE, we apply the Exponential Moving Average (EMA) to update the codebook. The encoder is trained using one A100 GPU with a micro-batch size of 16, and gradients are accumulated over 8 steps before performing backpropagation.

For modality translation, we train the motion and video projection layers using a single A100 GPU with a batch size of 32. The LLMs are further trained with a batch size of 16. To align the output template of BABEL-QA [13], we fine-tune the LLM model on BABEL-QA training set using a batch size of 16. We also list the dataset details in Tab. 4 and Tab. 5.

### 7. Additional Visualizations

#### 7.1. Visualizations for Motion Understanding

Additional visualization results for motion understanding are provided in Fig. 9 and Fig. 10. These examples highlight our method’s ability to capture detailed motion information and perform accurate motion analyses.

#### 7.2. Visualizations for Video Understanding

We also present visualizations for video understanding in Fig. 6.

Table 4. The dataset details for motion data. Motion-XQA\* refers to the training dataset generated following the method of Motion-XQA [5].

|         | Encoder Pretraining |  | Modality Translation |           | Instruction Tuning |          |
|---------|---------------------|--|----------------------|-----------|--------------------|----------|
| Data    | Motion-X            |  | Motion-X caption     | Humanml3D | Motion-XQA         | BABEL-QA |
| # count | 81 K                |  | 20 K                 | 10 K      | 200 K              | 2 K      |

Table 5. The dataset details for video data. Motion-XQA\* refers to the training dataset generated following the method of Motion-XQA [5].

|         | Modality Translation |       | Instruction Tuning |               |
|---------|----------------------|-------|--------------------|---------------|
| Data    | Motion-X caption     | VATEX | Motion-XQA*        | Video-ChatGPT |
| # count | 20 K                 | 20 K  | 200 K              | 100 K         |

### 7.3. Qualitative Comparisons with Other Video Understanding Methods

We compared our algorithm with Video-LLaVA [80] and MotionLLM [5] to evaluate their ability to understand video content. As shown in the Fig. 6, our algorithm demonstrates a better understanding of the video’s content. Video-LLaVA, lacking a deeper comprehension of motion, it struggles to effectively distinguish between actions such as tiling and push-ups. Additionally, Video-LLaVA cannot identify the exact moment when the worker picks up the tile, preventing it from answering questions about where the tiles were picked up.

MotionLLM, on the other hand, leverages motion data to accurately differentiate the workers’ actions. However, due to insufficient alignment between motion and video data, it also fails to correctly answer the question regarding where the tile was picked up. In comparison, our algorithm effectively integrates motion and video data, enabling it to address both questions.

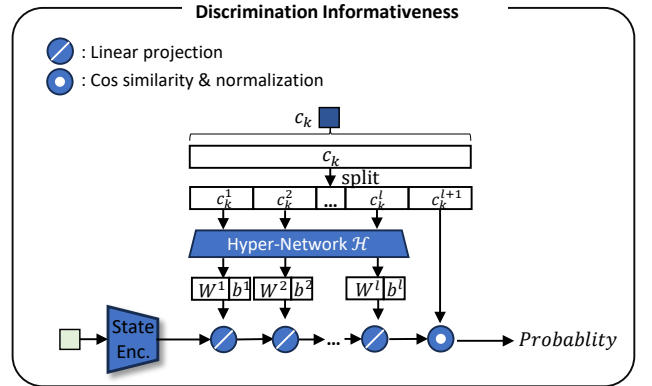


Figure 5. The structure of the hyper-network.

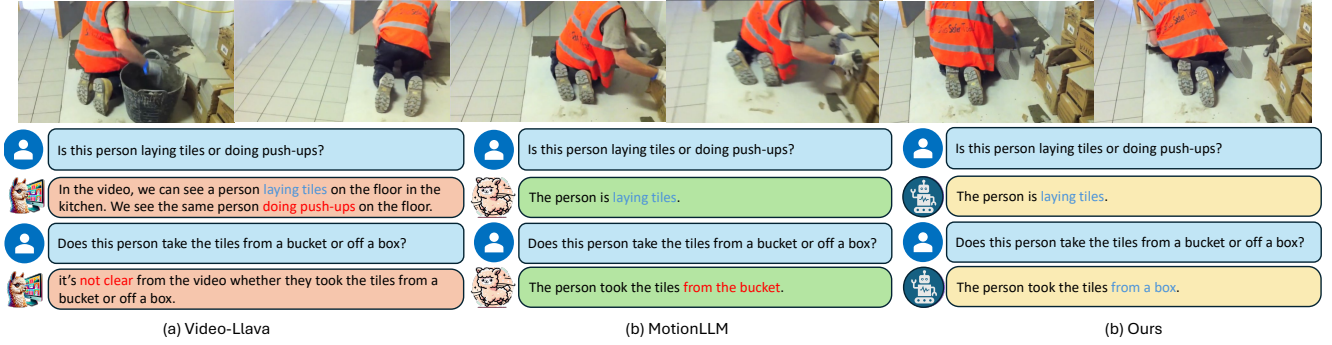


Figure 6. **Comparison between different video understanding methods.** We compare our method with Video-Llava and MotionLLM. We highlight the correct and wrong answers in blue and red, respectively.

Input: question, answer, prediction

LLM evaluation prompts:

Question | Ground truth | Prediction | Correctness

--- | --- | --- | ---

What does the person do? | sit | run | 0.0

What does the person do? | sit | sit | 1.0

What does the person do? | sit | sit down | 1.0

What does the person do? | sit | sit/walk | 0.5

Which part of the person touch the object? | right hand | right hand | 1.0

Which part of the person touch the object? | right hand | right arm | 0.5

Which part of the person touch the object? | right hand | left arm | 0.0

Figure 7. **Evaluation prompt for BABEL-QA benchmark.**

## 8. Evaluation Details

### 8.1. Evaluation on BABEL-QA Benchmark

For the BABEL-QA benchmark, we extend the evaluation protocol used in previous multi-modality LLMs evaluations [80]. The evaluation involves scoring the correctness of each Q&A pair generated by the LLMs. Detailed information about the evaluation prompt is provided in Fig. 7.

### 8.2. Evaluation on ActivityNet-QA Benchmark

On the ActivityNet-QA benchmark, we follow the approach of MotionLLM [5] to evaluate both the correctness of the predicted answers and assign a score ranging from 0 to 5, with 5 representing the highest level of semantic match. The details of the evaluation prompt are provided in Fig. 8.

## 9. Network Structure for Encoder Pretraining

### 9.1. Structure of Hyper-Networks

We utilize hyper-networks  $\mathcal{H}^u$  and  $\mathcal{H}^m$ , for velocity reconstruction, following InfoCon [46]. We split the discrete features  $D^u$  and  $D^m$  into  $K$  parts. We generate  $K - 1$  layer-network weights using the first  $K - 1$  feature parts, and the last feature part to calculate the cosine similarity with the output of a 3-layer-network, which is then used as the probability. We present the structure of the hyper-network in Fig. 5.

### 9.2. Structure of Network Components

We present the structure of the components of our network in Tab. 6.

### 9.3. Selection of modality

We set aside a portion of the data as a validation set and evaluated cross-entropy loss using video and motion as in-

Table 6. The network structure of encoder pretraining components

| Components                              | Architecture   |
|---|--|
| Motion Encoder $\mathcal{E}^m$          | (0): Linear(263, 256)  |
|   | (1): ReLU()  |
|   | (2): Linear(256, 512)  |
|   | (3): $4 \times \text{TransformerLayer}(512, \text{n\_head}=(8))$         |
|   | (4): Linear(512, 256)  |
|   | (5): ReLU()  |
| Motion Codebook $C^m$                   | (6): Linear(256, 128)  |
|   | (0): Parameter((512, 128), requires_grad=False)                          |
| Motion Projection Layer $\mathcal{P}^m$ | (0): Linear(128, 256)  |
| Motion Decoder $\mathcal{D}^m$          | (0): Linear(263, 256)  |
|   | (1): ReLU()  |
|   | (2): Linear(256, 512)  |
|   | (3): $4 \times \text{CasualTransformerLayer}(512, \text{n\_head}=(8))$   |
|   | (4): Linear(512, 256)  |
|   | (5): ReLU()  |
|   | (6): Linear(256, 512)  |
|   | (7): Linear(512, 256)  |
|   | (8): ReLU()  |
|   | (9): Linear(256, 263)  |
| Motion Velocity Decoder                 | (0): Linear(391, 256)  |
|   | (1): ReLU()  |
|   | (2): Linear(256, 512)  |
|   | (3): $4 \times \text{CasualTransformerLayer}(512, \text{n\_head}=(8))$   |
|   | (4): Linear(512, 256)  |
|   | (5): ReLU()  |
|   | (6): Linear(256, 512)  |
|   | (7): Linear(512, 256)  |
|   | (8): ReLU()  |
| Video Encoder $\mathcal{E}^u$           | (9): Linear(256, 263)  |
|   | (0): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))               |
|   | (1): Embedding(256, 768)   |
|   | (2): $12 \times \text{TransformerLayer}(768)$                            |
|   | (3): LayerNorm(768)  |
| Vision Codebook $C^u$                   | (4): TransformerLayer(768)   |
|   | (0): Parameter((512, 768), requires_grad=False)                          |
| Vision Projection Layer $\mathcal{P}^u$ | (0): Linear(768, 256)  |
| Video Decoder $\mathcal{D}^u$           | (0): Linear(768, 384)  |
|   | (1): $10 \times \text{CasualTransformerLayer}(384, \text{n\_head}=(16))$ |
|   | (2): Linear(384, 1536)   |
| Video Velocity Encoder $\mathcal{E}^u$  | (0): Conv2d(771, 768, kernel_size=(16, 16), stride=(16, 16))             |
|   | (1): Embedding(256, 768)   |
|   | (2): $12 \times \text{TransformerLayer}(768)$                            |
|   | (3): LayerNorm(768)  |
|   | (4): TransformerLayer(768)   |
| Vision Codebook $C^u$                   | (0): Parameter((512, 768), requires_grad=False)                          |
| Vision Projection Layer $\mathcal{P}^u$ | (0): Linear(768, 256)  |
| Video Decoder $\mathcal{D}^u$           | (0): Linear(768, 384)  |
|   | (1): $10 \times \text{CasualTransformerLayer}(384, \text{n\_head}=(16))$ |
|   | (2): Linear(384, 1536)   |

Input: question, answer, prediction

**LLM evaluation prompts:**

You are an intelligent chatbot designed for evaluating the correctness of generative outputs for question-answer pairs.

Your task is to compare the predicted answer with the correct answer and determine if they match meaningfully. Here's how you can accomplish the task:

— —

**##INSTRUCTIONS:**

- Focus on the meaningful match between the predicted answer and the correct answer.
- Consider synonyms or paraphrases as valid matches.
- Evaluate the correctness of the prediction compared to the answer.

Please evaluate the following video-based question-answer pair:

Question: {question}

Correct Answer: {answer}

Predicted Answer: {prediction}

Provide your evaluation only as a yes/no and score where the score is an integer value between 0 and 5, with 5 indicating the highest meaningful match.

Please generate the response in the form of a Python dictionary string with keys 'pred' and 'score', where value of 'pred' is a string of 'yes' or 'no' and value of 'score' is in INTEGER, not STRING.

DONOTPROVIDEANYOTHEROUTPUTTEXTTOEXPLANATION.OnlyprovidethePython dictionary string.

For example, your response should look like this: {'pred': 'yes', 'score': 4.8}.

Figure 8. Evaluation prompt for ActivityNet-QA benchmark.

puts. The results showed a loss of 0.88 for motion and 0.91 for video. The higher loss with video input is likely due to the fact that we are limited to using only 8 key frames (restricted by GPU memory; same as MotionLLM [5] and Video-LLaVa [40]). Therefore, in general, for long sequences, high-quality motion is preferred. For short videos without high-quality motion estimation, using video as input is more reliable due to information processing inequality.

#### 9.4. Ablation study on loss terms

. We conducted additional experiments for  $\mathcal{L}^{act}$  and  $\mathcal{L}^{dis}$  separately as shown in Table 7. Since the computation of  $\mathcal{L}^{dis}$  depends on  $\mathcal{L}^{act}$ , we adjusted its calculation in the absence of  $\mathcal{L}^{act}$  to directly take the input state as the input. The experimental results show that removing either loss impacts performance. However, the absence of  $\mathcal{L}^{dis}$  significantly hinders the network's ability to learn high-frequency information, leading to much less gain compared to removing  $\mathcal{L}^{act}$ .

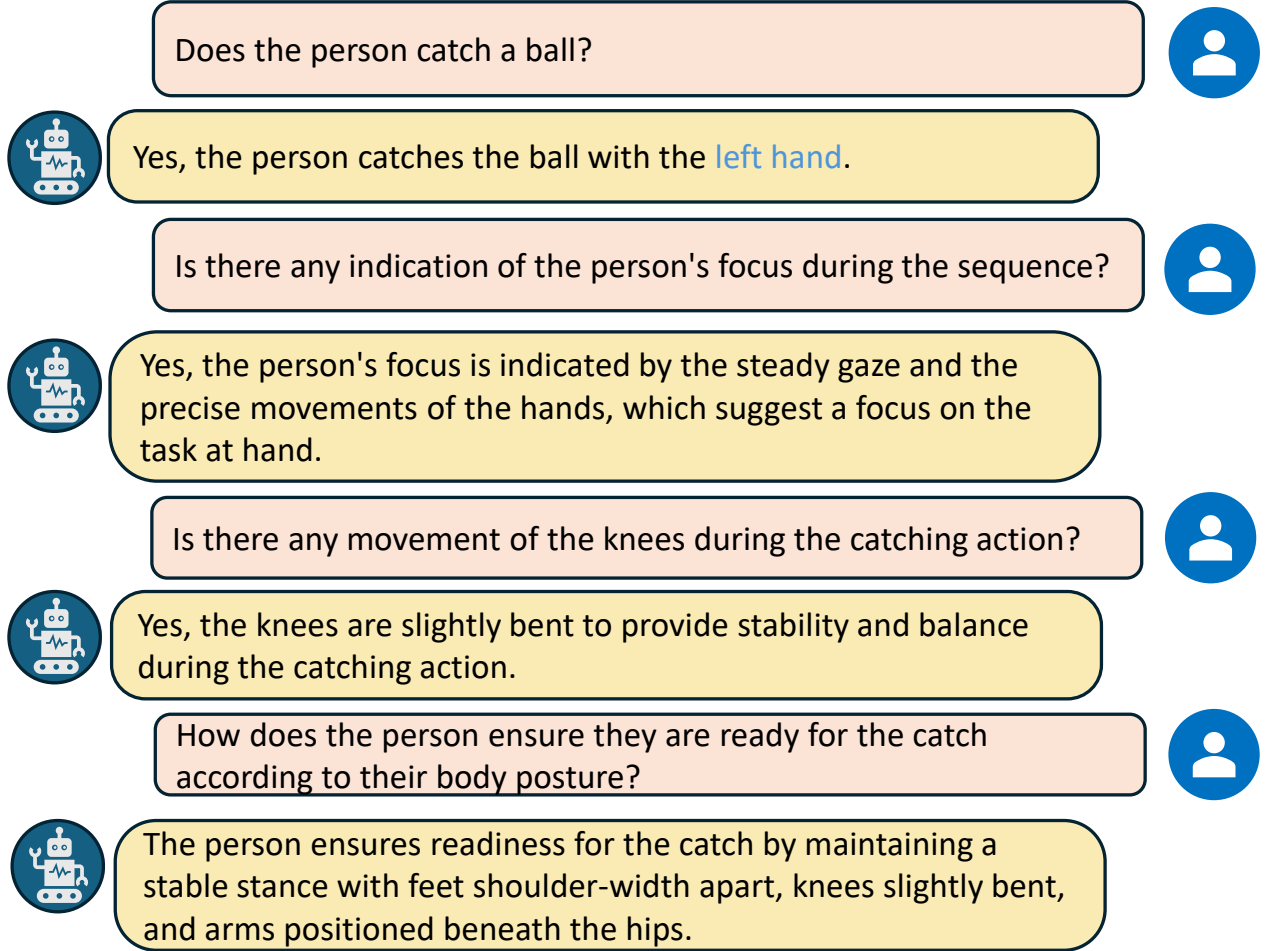
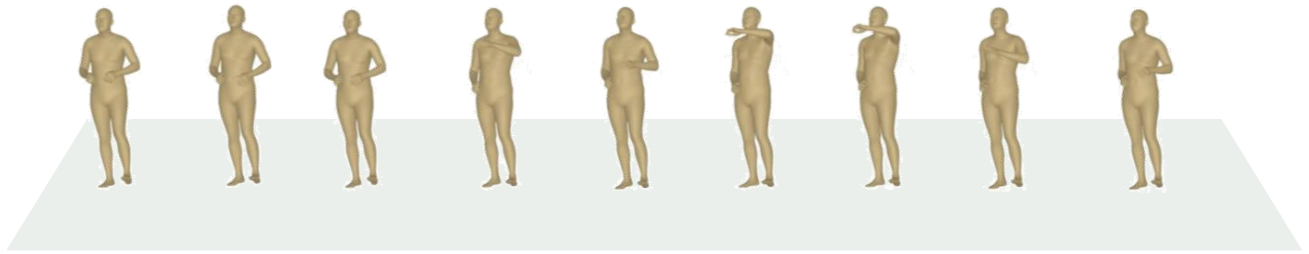
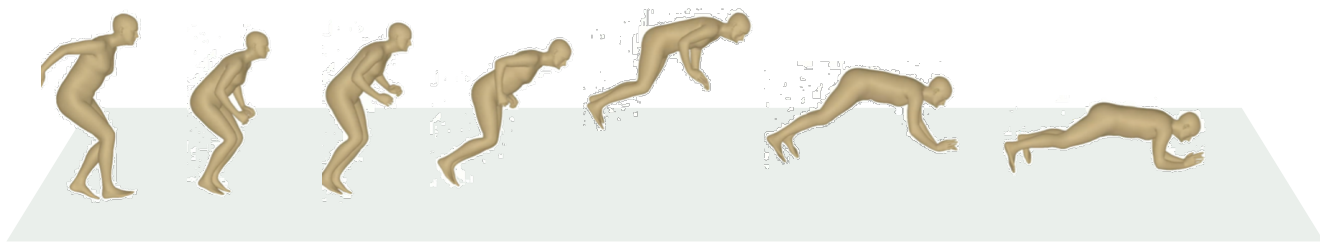


Figure 9. Visualizations for motion understanding.

Table 7. Additional ablation on specific loss terms.

| Model                          | Overall      | Query type   |              |              |
|--------------------------------|--------------|--------------|--------------|--------------|
|                                |              | Action       | Direction    | BodyPart     |
| Ours - w/o $\mathcal{L}^{act}$ | 0.563        | 0.605        | 0.478        | 0.465        |
| Ours - w/o $\mathcal{L}^{dis}$ | 0.629        | 0.683        | 0.457        | 0.577        |
| Ours                           | <b>0.711</b> | <b>0.809</b> | <b>0.697</b> | <b>0.623</b> |



Does the person jump or run?



The person jumps.

How does the body orientation change from the beginning to the point of impact in the karate front fall?



The body starts in a forward-leaning position, transitions into a horizontal alignment with the left leg raised, and then the body arches backward with the right leg raised to prepare for the impact.

How are the hands and knees used during the landing of the karate front fall?



During the landing, the hands are positioned under the hips for support, and the knees are bent to absorb the impact.

What is the final position of the individual's body upon landing on the mat?



Upon landing, the individual's body is in a controlled descent with the right knee bent to absorb the impact, the left leg trailing, and the arms positioned to aid in balance and stability.

Figure 10. Visualizations for motion understanding.