Bidirectional Cross-Modal Knowledge Exploration for Video Recognition with Pre-trained Vision-Language Models

Supplementary Material

Wenhao Wu[1,2] Xiaohan Wang[3] Haipeng Luo[4] Jingdong Wang[2] Yi Yang[3] Wanli Ouyang[5,1]
[1]The University of Sydney [2]Baidu Inc. [3]Zhejiang University
[4]University of Chinese Academy of Sciences [5]Shanghai AI Laboratory
whwu.ucas@gmail.com

In this appendix, we provide additional details and results for our approach. Specifically, §A contains further *details* on the training process (§A.1), attributes branch (§A.2), zero-shot evaluation (§A.3), statistics of video datasets (§A.4), visual encoder architectures (§A.5), and Distributed InfoNCE (§A.6). In §B, we present additional *results*, including comparisons on UCF-101 and HMDB-51 (§B.1) and more visualizations (§B.2).

A. Implementation Details

A.1. Training details

Regular Video Recognition. We present our approach for regular video recognition in Table A.1, sharing the same training recipe for all video datasets, including Kinetics-400, ActivityNet, Charades, HMDB-51, and UCF-101.

Few-shot Video Recognition. We repeat the samples to maintain the same number of iterations as the regular counterpart. For instance, if the model is trained on Kinetics-400 with around 900 iterations per epoch for the general setting, we repeat the sample to maintain the same number of iterations for few-shot settings. We train few-shot models for 2 epochs on Kinetics-400 and 10 epochs on other video datasets, *i.e.*, ActivityNet, HMDB-51, and UCF-101, while keeping other settings the same as in Table A.1.

Zero-shot Video Recognition. We use the Kinetics-400 pre-trained models to perform cross-dataset recognition without additional training on other datasets such as ActivityNet, HMDB-51, UCF-101, and Kinetics-600.

A.2. Attributes Branch

To improve the quality of auxiliary attributes, we pre-generate them using CLIP ViT-L/14 with 8 frames. We employ the text encoder architecture of CLIP ViT-B/32 as our attribute encoder. To integrate the *Attributes branch* with the *Video branch*, we set λ to 0.6 for the *Video branch* with ViT-B and λ to 0.8 for the *Video branch* with ViT-L.

Setting	Value
Training Hyperparameter	
Batch size	256
Vocabulary size	49408
Training epochs	30 (ViT-B), 20 (ViT-L)
Optimizer	AdamW
Learning rate (Base)	5e-5, cosine
Learning rate (CLIP layers)	5e-6, cosine
Weight decay	0.2
Linear warm-up epochs	5
Adam β_1,β_2	0.9, 0.999
Augmentation	
Resize	RandomSizedCrop
Crop size	224 (Default)
Random Flip	0.5
Random Gray scale	0.2

Table A.1. Default training recipe for video recognition.

A.3. Evaluation Protocols for Zero-shot Recognition

We employ our Kinetics-400 pre-trained models to evaluate on other datasets. For UCF-101, HMDB-51, and ActivityNet, we adopt two major evaluation protocols as described in [1]:

1. **Half-Classes Evaluation:** To ensure comparability with previous works, we randomly select half of the test dataset's classes - 50 for UCF, 25 for HMDB, and 100 for ActivityNet - and evaluate on the selected subset. We repeat this process ten times and average the results for each test dataset. We refer to this setting as UCF*, HMDB* and ActivityNet*.

2. **Full-Classes Evaluation:** This evaluation setting involves directly evaluating on the full dataset to return more realistic accuracy scores.

Model	Embedding dimension	Input resolution	Vision Transformer layers	Vision Transformer width	Vision Transformer heads	Text Transformer layers	Text Transformer width	Text Transformer heads
ViT-B/32	512	224	12	768	12	12	512	8
ViT-B/16	512	224	12	768	12	12	512	8
ViT-L/14	768	224	24	1024	16	12	768	12
ViT-L/14-336px	768	336	24	1024	16	12	768	12

Table A.2. CLIP-ViT hyperparameters

For Kinetics-600, we follow [3] to choose the 220 new categories outside of Kinetics-400 in Kinetics-600 for evaluation. We use the three splits provided by [3] and sample 160 categories for evaluation from the 220 categories in Kinetics-600 for each split. We report the mean accuracy of the three splits as the final accuracy.

A.4. Statistics of Video Datasets

We describe the video datasets used in our experiments:
Kinetics-400 is a large-scale video dataset that includes 240,000 training videos and 20,000 validation videos across 400 different human action categories. Each video in the dataset is a 10-second clip of an action moment, annotated from raw YouTube videos.
Kinetics-600 is an extension of Kinetics-400, consisting of approximately 480,000 videos from 600 action categories. The videos are divided into 390,000 for training, 30,000 for validation, and 60,000 for testing. In this paper, we use its test set for zero-shot evaluation.
UCF-101 is an action recognition dataset that contains 13,320 videos from 101 realistic action categories, collected from YouTube.
HMDB-51 is a collection of realistic videos from various sources, including movies and web videos. The dataset comprises 7,000 video clips from 51 action categories.
ActivityNet-v1.3 is a large-scale untrimmed video benchmark that contains 19,994 untrimmed videos of 5 to 10 minutes from 200 activity categories.
Charades is a video dataset designed for action recognition and localization tasks. It contains over 10,000 short video clips of people performing daily activities, and consists of 157 action categories.

A.5. Encoder Architectures

In this paper, we provide the complete architecture details of the visual encoder and textual encoders. The CLIP-ViT architectures are shown in Table A.2.

A.6. Distributed InfoNCE

Instead of Data-Parallel Training (DP), which is single-process, multi-thread, and only works on a single machine, Distributed Data-Parallel Training (DDP) is a widely adopted single-program multiple-data training paradigm for single- and multi-machine training. Due to GIL contention across threads, per-iteration replicated model, and additional overhead introduced by scattering inputs and gathering outputs, DP is usually slower than DDP even on a single machine. Hence, we develop the Distributed InfoNCE based on DDP for large batch size and fast training.

The core of the Distributed InfoNCE implementation is batch gathering, which enables us to calculate the NM$\times$NM similarity matrix across M GPUs for InfoNCE loss. Without batch gathering, each GPU only computes a local N$\times$N matrix where N$\ll$NM. This means that the cosine similarity and the InfoNCE loss would only be calculated for the pairs within a single GPU, and their gradients would be later averaged and synced. That's obviously not what we want.

The batch gathering technique allows each GPU to hold N vision features and perform a matrix product with NM text features, resulting in an N$\times$NM matrix. This computation is distributed (*i.e.*, sharded) across M GPUs, and we have calculated NM$\times$NM similarities across the GPUs in total. The loss we employ is symmetric, and the same process is applied *w.r.t.* text inputs. Algorithm 1 provides an example pseudocode to help understand the process.

B. More Results

B.1. Comparisons on UCF-101 and HMDB-51

In this section, we evaluate the performance of our method on the UCF-101 and HMDB-51 datasets to demonstrate its capacity for generalization to smaller datasets. We finetune our models on these two datasets using the pretrained ViT-L model on Kinetics-400 and report the accuracy on split one. We use 16 frames as inputs and train for 30 epochs. Table A.3 shows that our model has strong transferability, achieving a mean class accuracy of 98.8% on UCF-101 and 83.1% on HMDB-51.

B.2. More Qualitative Results

We present additional visualizations of the *Temporal Saliency* generated by our Video Concept Spotting mechanism in Figure A.1. In Figure A.2, we also showcase more visualizations of the *Generated Attributes* produced by our Video-Attribute Association mechanism using two different lexicons.

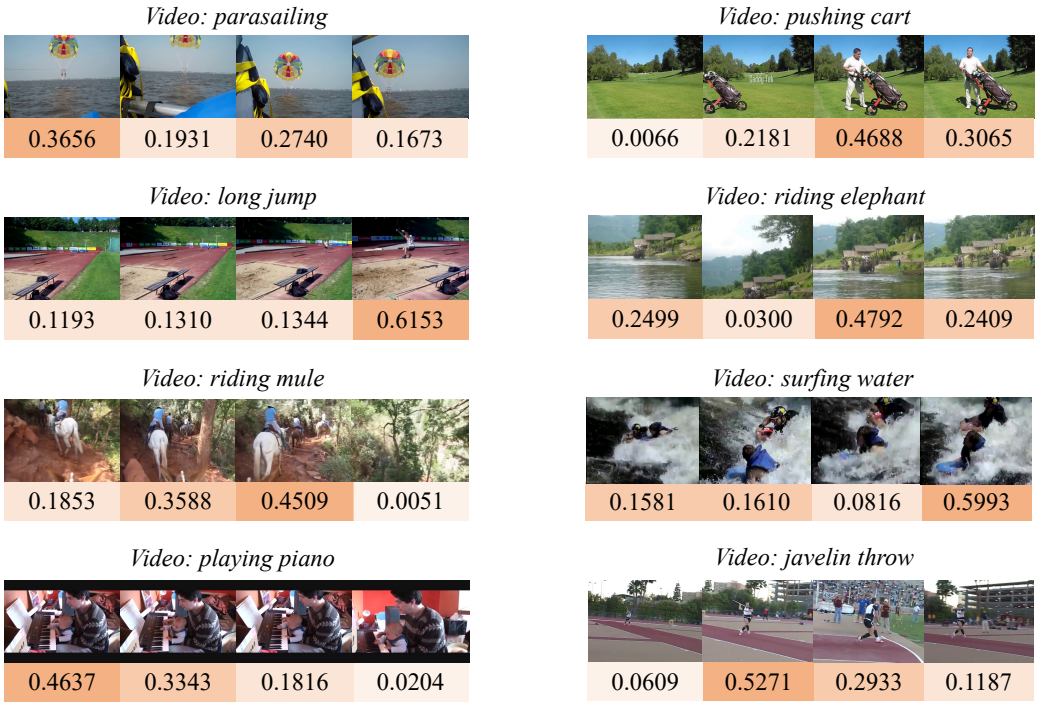
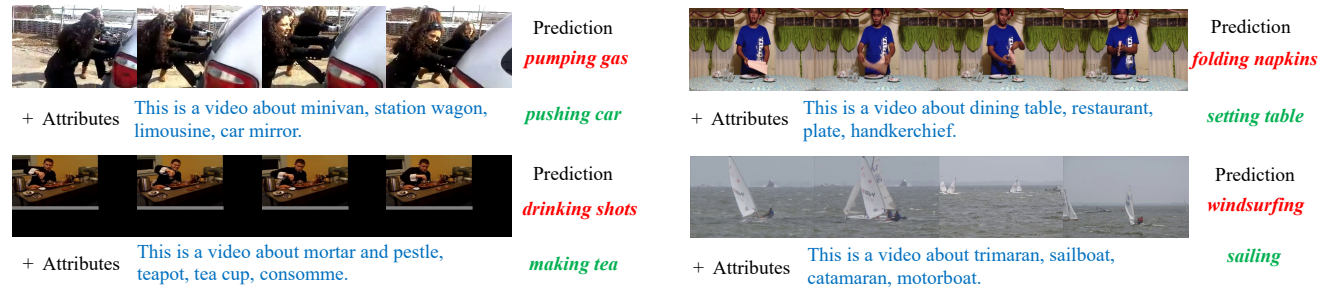


Figure A.1. Visualization of temporal saliency from our **Video Concept Spotting** mechanism. Please zoom in for best view.

(a) Generated attributes from Kinetics-400 lexicon.



(b) Generated attributes from ImageNet-1K lexicon.

Figure A.2. Visualization of the attribute sentence generated by the **Video-Attribute Association** mechanism that corrected the original incorrect prediction to the correct one.

Algorithm 1 Numpy-like Pseudocode of Distributed InfoNCE for our *Video branch*

```
# category_encoder: encoder network for category input
# video_encoder: encoder network for video input
# V: minibatch of video inputs
# T: minibatch of category inputs
# N: the local batch size of each GPU, e.g.,16
# M: the number of GPUs, e.g.,8
# N * M: the global batch size for multi-gpu training, e.g.,128

# extract feature representations of each modality
local_vision_features = video_encoder(V) # shape: [N, embed_dim]
local_text_features = category_encoder(T) # shape: [N, embed_dim]

# normalization
local_vision_features = l2_normalize(local_vision_features, axis=1)
local_text_features = l2_normalize(local_text_features, axis=1)

# batch_gather is a function gathering and concatenating the tensors across GPUs.
all_vision_features = batch_gather(local_vision_features) # shape: [N * M, embed_dim]
all_text_features = batch_gather(local_text_features) # shape: [N * M, embed_dim]

# scaled pairwise cosine similarities
# shape = [N, N * M]
logits_per_vision = logit_scale * local_vision_features @ all_text_features.t()
# shape = [N, N * M]
logits_per_text = logit_scale * local_text_features @ all_vision_features.t()

# The logits are then used as inputs for N*M-way (e.g., 128-way) classification,
# resulting in a loss value corresponding to N inputs in each GPU.
# Then Distributed Data Parallel mechanism takes care of averaging these across GPUs,
# which becomes equivalent to calculating the loss over NMxNM (e.g.,128x128) similarities.
```

Method	UCF-101	HMDB-51
ARTNet [7]	94.3%	70.9%
I3D [2]	95.6%	74.8%
R(2+1)D [6]	96.8%	74.5%
S3D-G [10]	96.8%	75.9%
TSM [5]	95.9%	73.5%
STM [4]	96.2%	72.2%
MVFNet [9]	96.6%	75.7%
TDN [8]	97.4%	76.4%
Ours ViT-L	**98.8%**	82.2%
Ours ViT-L (336↑)	98.6%	**83.1%**

Table A.3. Top-1 accuracy on UCF-101 and HMDB-51 achieved by different methods which are transferred from their **Kinetics Pre-trained** models with RGB modality.

References

[1] Biagio Brattoli, Joseph Tighe, Fedor Zhdanov, Pietro Perona, and Krzysztof Chalupka. Rethinking zero-shot video classification: End-to-end training for realistic applications. In *CVPR*, pages 4613–4623, 2020. 1

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 4

[3] Shizhe Chen and Dong Huang. Elaborative rehearsal for zero-shot action recognition. In *ICCV*, pages 13638–13647, 2021. 2

[4] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, pages 2000–2009, 2019. 4

[5] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 4

[6] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 4

[7] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *CVPR*, 2018. 4

[8] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *CVPR*, pages 1895–1904, 2021. 4

[9] Wenhao Wu, Dongliang He, Tianwei Lin, Fu Li, Chuang Gan, and Errui Ding. Mvfnet: Multi-view fusion network for efficient video recognition. In *AAAI*, 2021. 4

[10] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 4