Video Retrieval of Autonomous Driving Scenarios Based on Large Models

12310520 Rui Yuhan 12310437 Qiao Shihan

***Abstract*—Video retrieval in autonomous driving scenarios demands models to comprehend complex visual environments and align them with detailed textual descriptions. While multimodal large language models (MLLMs) have demonstrated potential in vision-language tasks, their application to domain-specific video retrieval remains underexplored. Furthermore, the scarcity of publicly available high-quality annotated video data in the autonomous driving domain makes it challenging for models to accurately learn video content semantics from limited data. Additionally, significant differences exist between autonomous driving videos and those in general video datasets, leading to suboptimal performance of generic large models on autonomous driving datasets.**

To address these challenges, we evaluate the performance of two distinct pretrained MLLMs on a carefully curated autonomous driving dataset. This dataset features high-quality captions that have undergone multi-round optimization, combining human annotation expertise with advanced automated labeling techniques, while also serving as a benchmark platform for exploring more efficient annotation strategies. Experimental results demonstrate that our approach effectively retrieves relevant video segments based on textual queries, highlighting the potential of MLLMs to enhance retrieval accuracy and scalability.

***Index Terms*—Autonomous Driving Video Retrieval, Multimodal Large Language Models (MLLMs), Video Understanding, Automated Annotation Optimization**

I. Introduction

II. Two knowledge distillation methods used

A. Research Background and Motivation of Knowledge Distillation

With the rapid advancement of large-scale foundation models, Vision-Language Models (VLMs), and Large Language Models (LLMs), deploying these models on edge devices faces significant challenges due to high computational and memory requirements. Knowledge Distillation (KD), first proposed by Hinton et al. [1], addresses this issue by training a lightweight student network under the supervision of a complex teacher model. Unlike compression methods that modify network structures, KD transfers knowledge by having the student mimic the teacher's output distribution as soft labels, enabling effective model compression while maintaining performance. Additionally, KD facilitates knowledge transfer from source tasks to target tasks with limited labeled data, making it particularly valuable for domain-specific applications such as autonomous driving video retrieval.

1) Mainstream Knowledge Distillation Approaches: Knowledge distillation methods can be categorized based on the source of knowledge being distilled [2]. The three primary categories are: **logit-based distillation**, which transfers final predictions using softmax with temperature to create soft labels; **feature-based distillation**, which transfers intermediate layer representations and attention maps, providing richer information than logits alone; and **similarity-based distillation**, which transfers structural knowledge through pairwise similarities between features, channels, or instances. Additionally, distillation can be organized by training schemes: **offline distillation** (pre-trained teacher with frozen weights), **online distillation** (simultaneous training), and **self-distillation** (same network transferring knowledge across layers or stages). Other notable approaches include **attention-based distillation**for transferring focus patterns, **contrastive distillation** leveraging contrastive learning principles, and **cross-modal distillation** for transferring knowledge between different modalities, which is particularly relevant for multimodal tasks like video retrieval.

B. DLDKD

DLDKD (Dual Learning with Dynamic Knowledge Distillation) [3] employs a dual-stream teacher-student framework that transfers knowledge from a large teacher model to a lightweight student model through multi-level distillation.

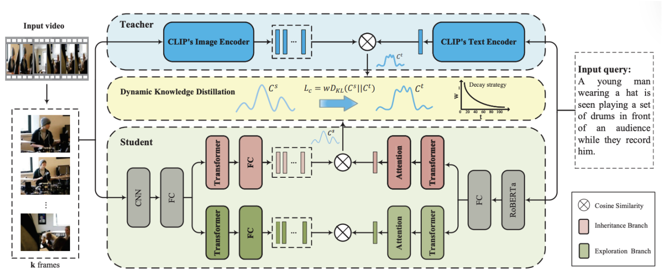


Fig. 1. The framework of DLDKD.

DLDKD performs distillation at multiple levels: feature-level distillation transfers intermediate representations, logit-level distillation aligns output distributions, and attention-level distillation preserves attention patterns.

1) Loss Functions: The DLDKD framework employs a dual learning objective that combines knowledge from the teacher model with task-specific learning through two student branches. The total loss function is defined as:

$$\mathcal{L}_{total} = \lambda_I \mathcal{L}_I + \lambda_E \mathcal{L}_E + \mathcal{L}_{task} \tag{1}$$

where λ_I and λ_E are dynamic weights that adjust the contribution of the inheritance branch and exploration branch losses, respectively.

Inheritance Student Branch Loss ($\mathcal{L}_I$): The inheritance branch learns to absorb knowledge from the teacher model through multi-level distillation. The loss combines feature-level, logit-level, and attention-level distillation:

$$\mathcal{L}_I = \alpha_f \mathcal{L}_{feat} + \alpha_l \mathcal{L}_{logit} + \alpha_a \mathcal{L}_{attn} \tag{2}$$

where α_f, α_l, and α_a are weights for feature-level, logit-level, and attention-level distillation losses, respectively.

The feature-level distillation loss aligns intermediate representations between teacher and student:

$$\mathcal{L}_{feat} = \frac{1}{N} \sum_{i=1}^{N} \| F_t^{(i)} - F_s^{(i)} \|_2^2 \tag{3}$$

where $F_t^{(i)}$ and $F_s^{(i)}$ denote the feature representations at layer i from the teacher and student models, respectively, and N is the number of layers used for feature distillation.

The logit-level distillation loss uses KL divergence to align output distributions:

$$\mathcal{L}_{logit} = \mathrm{KL}(\mathrm{softmax}(z_t/\tau) \| \mathrm{softmax}(z_s/\tau)) \tag{4}$$

where z_t and z_s are the logits from teacher and student models, and τ is the temperature parameter for softening the distributions.

The attention-level distillation loss preserves the teacher's attention patterns:

$$\mathcal{L}_{attn} = \frac{1}{L} \sum_{l=1}^{L} \| A_t^{(l)} - A_s^{(l)} \|_F^2 \tag{5}$$

where $A_t^{(l)}$ and $A_s^{(l)}$ are attention maps at layer l from teacher and student models, L is the number of attention layers, and $\| \cdot \|_F$ denotes the Frobenius norm.

Exploration Student Branch Loss ($\mathcal{L}_E$): The exploration branch focuses on task-specific learning to capture domain-specific patterns that may not be fully represented in the teacher model:

$$\mathcal{L}_E = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\mathrm{sim}(v_i, t_i^+)/\tau_E)}{\sum_{j=1}^{B} \exp(\mathrm{sim}(v_i, t_j)/\tau_E)} \tag{6}$$

where B is the batch size, v_i is the video embedding, t_i^+ is the positive text query, $\mathrm{sim}(\cdot, \cdot)$ computes cosine similarity, and τ_E is the temperature parameter for the exploration branch.

Task-Specific Loss ($\mathcal{L}_{task}$): The task loss ensures that the final model performs well on the video-text retrieval task:

$$\mathcal{L}_{task} = \mathcal{L}_{retrieval} = -\log P(v^+|t) - \log P(t^+|v) \tag{7}$$

where $P(v^+|t)$ and $P(t^+|v)$ represent the probabilities of retrieving the correct video given text query and vice versa.

Dynamic Weight Adjustment: The weights λ_I and λ_E are dynamically adjusted during training based on the performance of each branch. This allows the framework to automatically balance between teacher knowledge inheritance and task-specific exploration:

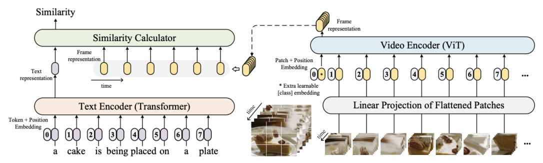
$$\lambda_I^{(t)} = \lambda_I^{(0)} \cdot \exp(-\gamma_I \cdot t), \quad \lambda_E^{(t)} = \lambda_E^{(0)} \cdot (1 + \gamma_E \cdot t) \tag{8}$$

where t is the training epoch, $\lambda_I^{(0)}$ and $\lambda_E^{(0)}$ are initial weights, and γ_I, γ_E are decay/growth rates that control the dynamic adjustment schedule.

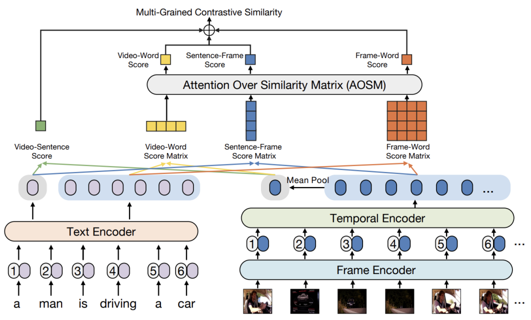
2) Feature Pre-Extraction Strategy: DLDKD employs feature pre-extraction to accelerate training: teacher model features are pre-computed and cached for all training samples before distillation training, eliminating repeated forward passes through the teacher model during training. In the subsequent modifications to TeachCLIP, we will incorporate this feature pre-extraction strategy to shorten the training time.

C. TeachCLIP

1) Inspiration: Existing VTR models generally fall into two categories: lightweight global feature models (e.g., CLIP4Clip) and heavy fine-grained models (e.g., X-CLIP). The former is computationally efficient but often lacks frame-level details, resulting in suboptimal retrieval accuracy. The latter leverages frame-level interactions to enhance performance but incurs high computational costs.


Similarity
Similarity Calculator
Frame representation
Text representation
time
Text Encoder (Transformer)
Token + Position Embedding
a cake is being placed on a plate
Frame representation
Video Encoder (ViT)
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches
time

(a) CLIP4Clip Framework



(b) X-CLIP Framework

Fig. 2. Comparison of CLIP4Clip and X-CLIP frameworks. Both use similar bottom-up encoding but differ in upper-layer feature usage.

In terms of vision and text encoding, both CLIP4Clip and X-CLIP share nearly identical bottom-level encoding mechanisms (Frame $\rightarrow$ Patch $\rightarrow$ ViT, Word $\rightarrow$ Embedding $\rightarrow$ Transformer). The primary distinction lies in how they utilize these features in the upper layers: CLIP4Clip performs pooling to retain only global video vectors, whereas X-CLIP preserves fine-grained features and performs multi-granularity alignment.

TeachCLIP aims to bridge this gap by distilling the fine-grained alignment capability of heavy models into a lightweight student model.

2) Framework: The TeachCLIP framework is designed to transfer the fine-grained knowledge from a teacher model to a student model while maintaining high inference efficiency.

Student Model: The student model is built upon CLIP4Clip. It samples the video into frames, processes each frame through the CLIP ViT to obtain frame features, and enhances them via a temporal Transformer. A key innovation

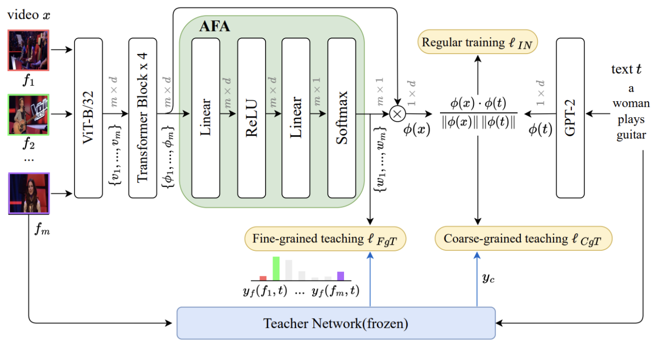

video x
f_1
f_2
f_m
ViT-B/32
Transformer Block x 4
AFA
Linear
ReLU
Linear
Softmax
Regular training ℓ_{IN}
$\frac{\phi(x)\cdot\phi(t)}{\|\phi(x)\|\,\|\phi(t)\|}$
$\phi(x)$
$\phi(t)$
GPT-2
text t
a woman plays guitar
Fine-grained teaching ℓ_{FgT}
Coarse-grained teaching ℓ_{CgT}
$y_f(f_1,t)$ ... $y_f(f_m,t)$
y_c
Teacher Network(frozen)

Fig. 3. The framework of TeachCLIP.

is replacing the original mean pooling with Attentional Frame-Feature Aggregation (AFA). AFA generates frame weights $\{w_i\}$ to compute a weighted sum of frame features:

$$\phi(x) = \sum_{i=1}^{m} w_i \cdot \phi_i \tag{9}$$

The AFA module consists of a lightweight structure (Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ Softmax) and introduces negligible parameters.

Teacher Model: The teacher model provides two types of supervision:

- **Video-level soft labels (y_c):** The teacher provides the distribution of video-text correlations to guide the student's ranking.
- **Frame-level soft labels (y_f):** The teacher provides the distribution of frame-text correlations to guide the student's frame weight allocation in AFA.

Learning Objectives: The training involves three losses:

1) *Frame-level Distillation (ℓ_{FgT}):* Optimizes the AFA weights to match the teacher's frame-text similarity distribution. The loss function is defined as:

$$\ell_{FgT} = -\frac{1}{b}\sum_{i=1}^{b}\sum_{k=1}^{m} y_f(f_{i,k}, t_i)\,\log w_{i,k} \tag{10}$$

where b is the batch size, m is the number of frames, and y_f represents the frame-text similarity from the teacher.

2) *Video-level Distillation (ℓ_{CgT}):* Aligns the student's video-text similarity matrix with the teacher's multi-grained similarity matrix using the Pearson distance d_p:

$$\begin{aligned}\ell_{CgT} = &\frac{1}{b}\sum_{i} d_p(\sigma(B_{i,\cdot}), \sigma(y_c(v_i,\cdot))) \\ &+\frac{1}{b}\sum_{j} d_p(\sigma(B_{\cdot,j}), \sigma(y_c(\cdot,t_j)))\end{aligned} \tag{11}$$

where σ denotes the softmax function and B is the similarity matrix of the student.

3) *Contrastive Learning (ℓ_{IN}):* Standard maximization of similarity for positive pairs and minimization for negative pairs, using symmetric InfoNCE loss:

$$\ell_{IN} = \frac{1}{2}\left[\ell_{NCE}^{v\to t} + \ell_{NCE}^{t\to v}\right] \tag{12}$$

where

$$\ell_{NCE}^{v\to t} = -\frac{1}{b}\sum_{i=1}^{b}\log\frac{\exp(B_{ii}/\tau)}{\sum_{j=1}^{b}\exp(B_{ij}/\tau)} \tag{13}$$

and $\ell_{NCE}^{t\to v}$ is defined symmetrically.

The total loss is $\ell = \ell_{CgT} + \ell_{FgT} + \ell_{IN}$. During inference, the teacher model is discarded, and only the lightweight student model with AFA is used.

3) *TeachCLIP vs X-CLIP Parameters and FLOPs Comparison:* Table I presents the comparison of parameters and FLOPs between TeachCLIP and the teacher model (X-CLIP).

TABLE I
COMPARISON OF PARAMETERS AND FLOPS

Model	Parameters	FLOPs (Inference)	Description
TeachCLIP	≈ 200M	53.65G (12 frames)	Student model, based on CLIP4Clip + AFA, maintains lightweight inference
X-CLIP (Teacher)	≈ 220M	145G (8 frames) / 287G (16 frames)	Teacher model, introduces multi-grained contrastive similarity, higher inference cost

III. Alterations on TeachCLIP

A. On student model

B. On teacher model

IV. Post-Training Quantization of VAST Model using MS-Swift

In this section, we apply post-training quantization methods introduced in MS-Swift [4] to compress the VAST model [5] for video-text retrieval tasks. Post-training quantization is a model compression technique that reduces the precision of model weights and activations after the model has been fully trained, enabling efficient deployment on resource-constrained devices without requiring retraining from scratch.

A. MS-Swift Quantization Architecture and Methodology

MS-Swift (ModelScope Swift) [4] implements a GPTQ-based quantization framework for post-training quantization of large-scale models. While the documentation mentions uniform, dynamic, and static quantization with INT8 precision, the actual implementation focuses on GPTQ (GPT Quantization) with INT4 precision, providing a more specialized and efficient quantization approach.

The key components of MS-Swift's quantization implementation include:

GPTQ Quantization Method: The framework employs GPTQ, a post-training quantization method that quantizes model weights to INT4 precision. GPTQ uses a layer-wise quantization strategy that minimizes reconstruction error by optimizing quantization parameters. The method involves grouping weights and applying quantization sequentially, which allows for better accuracy preservation compared to uniform quantization approaches.

GPTQ Parameters: The quantization process is configured through several key parameters:

- **group_size:** Controls the grouping strategy for weight quantization, determining how weights are partitioned for quantization
- **sym:** Specifies whether to use symmetric quantization (sym=true) or asymmetric quantization (sym=false)
- **true_sequential:** When enabled, ensures that quantization is performed sequentially across layers, maintaining dependencies between layers during the quantization process

Module-Level Selective Quantization: Rather than applying uniform quantization across the entire model, MS-Swift supports selective quantization at the module level. This allows different modules or layers to be quantized with different configurations or to remain unquantized, based on their sensitivity to quantization error and their importance to model performance.

Custom Adapters and Runtime Patches: The framework provides mechanisms for custom adapters and runtime patches, enabling fine-grained control over the quantization process and allowing for model-specific optimizations. These adapters can be used to handle special cases or to apply quantization only to specific components of the model.

Multimodal Model Handling: For multimodal models like VAST, the framework includes special handling mechanisms. Specifically, certain branches (such as audio branches) can be selectively excluded from quantization or masked during the quantization process, ensuring that modality-specific components maintain their full precision when necessary.

Grouped and Sequential Quantization Strategies: The implementation supports both grouped quantization, where weights are quantized in groups (controlled by group_size), and sequential quantization (true_sequential), where layers are quantized one after another to maintain inter-layer dependencies. These strategies are critical for maintaining model accuracy during INT4 quantization.

In practice, MS-Swift's quantization implementation is more specialized toward GPTQ-based modular quantization rather than a generic quantization framework, making it particularly suitable for large language models and multimodal foundation models where INT4 precision can achieve significant compression while preserving model performance.

B. VAST Architecture for Video-Text Retrieval

VAST (Vision-Audio-Subtitle-Text) [5] is an omni-modality foundation model. The VAST framework architecture is illustrated in Figure 4.

For video-text retrieval, VAST employs dual encoders: a video encoder that processes frames through a ViT architecture with temporal modeling, and a text encoder based on transformer architecture. The model learns cross-modal alignment in a shared embedding space, using contrastive learning to maximize similarity between matching video-text pairs.

The VAST model achieves strong performance in video-text retrieval after fine-tuning on domain-specific data. Given the model's size, there is room for further lightweight optimization

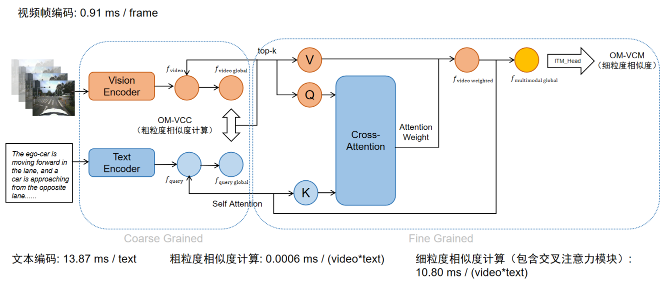


Fig. 4. The framework of VAST.

to better adapt to real-time applications. We therefore apply post-training quantization to obtain smaller model weights while maintaining competitive performance.

C. Post-Training Quantization Experimental Process

We perform post-training quantization on the fine-tuned VAST model weights using MS-Swift's quantization framework. The experimental process consists of the following steps:

Model Preparation: We start with a VAST model that has been fine-tuned on our target video-text retrieval dataset. The fine-tuned model serves as the full-precision baseline, providing reference accuracy metrics for comparison with the quantized version.

Calibration Dataset Preparation: A representative subset of the training dataset is selected for calibration. This calibration set typically consists of 100-1000 samples that cover the diversity of the full dataset. The calibration data is used to compute quantization parameters without requiring gradient computation or backpropagation.

Quantization Configuration: We configure the quantization settings using MS-Swift's API, specifying:

- Target precision: INT8 for both weights and activations
- Quantization scheme: Static quantization with calibration
- Layer-wise precision: Maintaining FP32 precision for sensitive layers (e.g., layer normalization, final projection layers)
- Calibration method: Min-max calibration or percentile-based calibration

Quantization Execution: The quantization process is performed in two phases:

1) *Calibration Phase:* The calibration dataset is passed through the model in inference mode. Activation statistics are collected for each layer to determine optimal quantization scales and zero-points.
2) *Conversion Phase:* The full-precision model weights are quantized to INT8 using the calibrated parameters, and the model graph is transformed to use quantized operations.

Evaluation: The quantized model is evaluated on the test set to assess the impact of quantization on retrieval performance. Key metrics include retrieval accuracy (Recall@K for K=1, 5, 10), mean reciprocal rank (MRR), and inference latency. Additionally, we measure the model size reduction and memory footprint to quantify the compression benefits.

Performance Optimization: If the quantized model shows significant accuracy degradation, we apply fine-tuning strategies such as quantization-aware fine-tuning (QAT) on the quantized model, or adjust the quantization configuration to use mixed-precision quantization for critical layers.

The quantized VAST model, achieved through MS-Swift's post-training quantization, enables efficient deployment while maintaining competitive retrieval performance, making it suitable for real-time video-text retrieval applications in autonomous driving scenarios.

V. Conclusion

Autonomous driving video retrieval presents unique challenges due to the complexity of real-world driving scenarios and the scarcity of high-quality annotated datasets. In this work, we explored the potential of multimodal large language models (MLLMs) to bridge the gap between visual perception and textual understanding in autonomous driving contexts. By leveraging a carefully curated dataset with optimized annotations, combining human expertise and automated refinement using Video-LLama2, we demonstrated that pretrained MLLMs can be effectively adapted for domain-specific video retrieval tasks.

Our experiments with two state-of-the-art MLLMs (Vast and Clip-Vip) revealed that fine-tuning on high-quality annotated data significantly improves retrieval accuracy, validating the importance of robust dataset construction. Additionally, our work establishes a benchmark for future research in autonomous driving video understanding, providing a foundation for developing more interpretable and scalable retrieval systems.

Moving forward, we anticipate that further advancements in MLLMs, combined with more sophisticated annotation pipelines, will enhance the generalization capabilities of autonomous driving systems. Future research could explore dynamic annotation strategies, real-time retrieval optimization, and the integration of multi-sensor data (e.g., LiDAR and radar) to further improve model performance in complex driving environments.

Ultimately, this study highlights the critical role of high-quality data and domain-specific adaptation in advancing autonomous driving technologies, paving the way for safer and more intelligent transportation systems.

References

[1] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015, nIPS 2014 Deep Learning Workshop. [Online]. Available: https://arxiv.org/abs/1503.02531

[2] A. M. Mansourian, R. Ahmadi, M. Ghafouri, A. M. Babaei, E. B. Golezani, Z. Y. Ghamchi, V. Ramezanian, A. Taherian, K. Dinashi, A. Miri, and S. Kasaei, "A comprehensive survey on knowledge distillation," *Transactions on Machine Learning Research*, 2025, published at TMLR, arXiv:2503.12067. [Online]. Available: https://arxiv.org/abs/2503.12067

[3] J. Dong, M. Zhang, Z. Zhang, X. Chen, D. Liu, X. Qu, X. Wang, and B. Liu, "Dual learning with dynamic knowledge distillation for partially relevant video retrieval," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 11 302–11 312.

[4] Y. Zhao, J. Huang, J. Hu, X. Wang, Y. Mao, D. Zhang, H. Zhang, Z. Jiang, Z. Wu, B. Ai, A. Wang, W. Zhou, and Y. Chen, "Swift: A scalable lightweight infrastructure for fine-tuning," 2024. [Online]. Available: https://arxiv.org/abs/2408.05517

[5] S. Chen, H. Li, Q. Wang, Z. Zhao, M. Sun, X. Zhu, and J. Liu, "Vast: A vision-audio-subtitle-text omni-modality foundation model and dataset," *Advances in Neural Information Processing Systems*, vol. 36, 2023, neurIPS 2023. [Online]. Available: https://arxiv.org/abs/2305.18500