Knowledge Distillation Based on Large Video Retrieval Models

12310520 Rui Yuhan 12310437 Qiao Shihan

***Abstract*—In the domain of autonomous driving (AD), deploying effective video retrieval systems faces a critical bottleneck: the high computational demand of large multimodal models versus the strict resource constraints of edge devices. This report focuses on lightweighting strategies to resolve this conflict. We explore two complementary pathways to achieve efficient video retrieval: Knowledge Distillation (KD) and Post-Training Quantization (PTQ). First, leveraging the TeachCLIP framework, we distill complex knowledge from heavy teacher models into lightweight student networks. We propose an acceleration strategy via "Offline Feature Extraction" (improving training throughput by 9.7%) and investigate architectural lightweighting by replacing the visual encoder with ConvNeXt. Second, we apply GPTQ-based post-training quantization to the massive VAST foundation model. By compressing the model to INT4 precision, we achieve a 2.80× reduction in global model size and a significant 4.33× compression of the vision encoder. These efforts collectively demonstrate viable paths toward constructing lightweight, high-performance video retrieval systems suitable for on-vehicle deployment.**

***Index Terms*—Autonomous Driving, Video Retrieval, Lightweight Deep Learning, Knowledge Distillation, Model Quantization, TeachCLIP, VAST.**

I. Introduction

Efficient video retrieval is critical for autonomous driving safety validation and data mining, yet deploying powerful Multimodal Large Language Models (MLLMs) on resource-constrained vehicles remains a significant bottleneck. To address this, we focus on **model lightweighting** strategies that bridge the gap between high-level semantic understanding and on-board hardware limitations.

This report explores two optimization paradigms. First, we leverage Knowledge Distillation (KD) via the TeachCLIP framework [1], transferring fine-grained knowledge from heavy teachers to lightweight students. We enhance this pipeline with an "Offline Feature Extraction" strategy for faster training and investigate architectural optimizations using ConvNeXt encoders. Second, we employ Post-Training Quantization (PTQ) on the VAST foundation model. By quantizing weights to INT4 precision using MS-Swift, we drastically reduce memory footprint while retaining retrieval capabilities.

The subsequent sections detail our methods: Section II reviews KD; Section III presents our TeachCLIP optimizations; Section IV discusses VAST quantization; and Section V concludes with performance insights.

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. [2], 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 [3]. 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) [4] employs a dual-stream teacher-student framework that transfers knowledge from a large teacher model to a lightweight student model through multi-level distillation.

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:

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

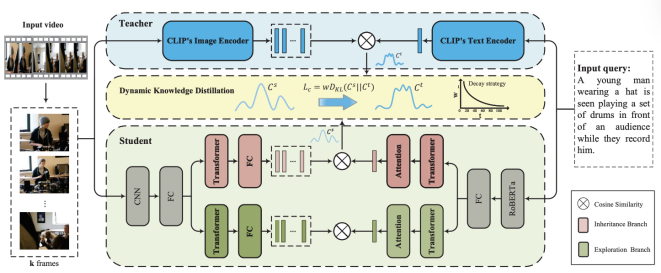


Fig. 1. The framework of DLDKD.

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:

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

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:

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

where B is the batch size, v_i is the video embedding, t_i^+ is the positive text query, $\text{sim}(\cdot,\cdot)$ 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:

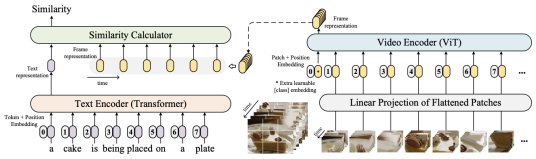
$$\lambda_I^{(t)} = \lambda_I^{(0)} \cdot \exp(-\gamma_I \cdot t), \quad \lambda_E^{(t)} = \lambda_E^{(0)} \cdot (1 + \gamma_E \cdot t) \tag{8}$$

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.

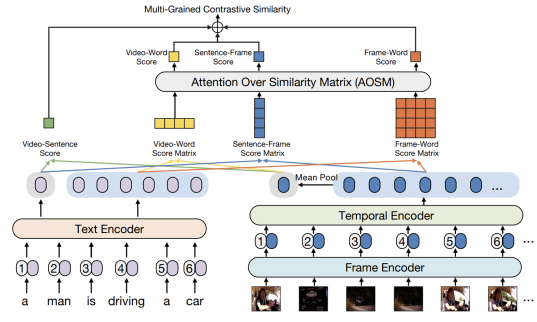
2) Feature Pre-Extraction Strategy: DLDKD employs feature pre-extraction to accelerate training: teacher model features are pre-computed and cached for all training samples before distillation training, eliminating repeated forward passes through the teacher model during training. In the subsequent modifications 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 [5]) and heavy fine-grained models (e.g., X-CLIP [6]). 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.



(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 → Patch → ViT, Word → Embedding

→ 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.

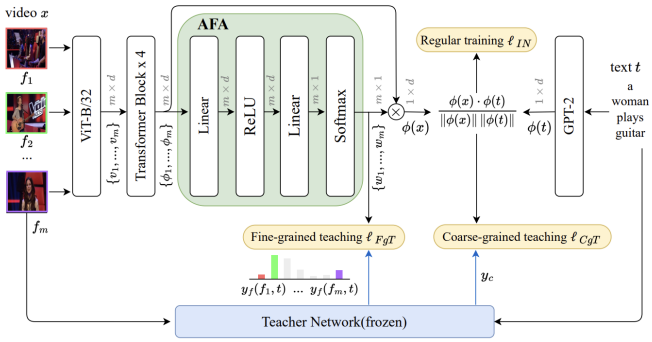


Fig. 3. The framework of TeachCLIP.

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 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 → ReLU → Linear → 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. Feature Pre-computation Acceleration

We precompute and cache frame-level visual features offline to eliminate the expensive video encoding overhead from the training loop. Specifically, each video is sampled into a fixed number of frames (F), and every frame is processed by the **teacher model (X-CLIP)**'s visual encoder to obtain high-level semantic embeddings (e.g., 512-D per frame for ViT-B/32). The resulting feature tensor $\mathbf{V} \in \mathbb{R}^{F\times d}$ is saved alongside a binary frame mask $\mathbf{m} \in \{0,1\}^F$ to handle varying video lengths.

During distillation training, the student directly loads the pre-computed pair $(\mathbf{V}, \mathbf{m})$, bypassing the need to re-encode raw pixels. This strategy significantly enhances throughput and reproducibility. As shown in Table II, this optimization yields a measurable speedup in training time per step.

TABLE II
TRAINING EFFICIENCY WITH PRECOMPUTED TEACHER FEATURES

Setting	Time/step (s)	Speedup
Online Extraction (Baseline)	1.91	–
Offline Pre-computation	**1.74**	**+9.7%**

B. On Student Model

We investigated replacing the Transformer-based visual encoder (ViT) in the student model with ConvNeXt, leveraging the open-source implementation and pre-trained weights from OpenCLIP [7], which provides reproducible training of CLIP models across diverse architectures. The objective was to explore whether CNN inductive biases, such as translation invariance and locality, could offer advantages in visual feature extraction or efficiency compared to the standard ViT. However, as detailed below, realizing these potential benefits in the video domain proved non-trivial.

1) Exploration under OpenCLIP Framework on COCO Dataset: We first validated fine-tuning strategies on COCO using ConvNeXtV2-Tiny. As shown in Table III, full fine-tuning caused overfitting, whereas a "controlled gradual unfreezing" strategy (unfreezing text at epoch 3, visual at epoch 5) achieved the best balance (I2T R@1: 9.55%).

TABLE III
COMPARISON OF CONVNEXTV2-TINY STRATEGIES ON COCO

Strategy	I2T R@1	Observation
No Freeze (Full Fine-tuning)	8.12%	Overfitting, worst performance
Unfreeze 3-5 (Gradual)	**9.55%**	**Best balance, effective constraint**
Change Text Tower + Unfreeze	9.07%	Intermediate performance

Extending this to ConvNeXt-Base with attentional pooling ('absattn'), we tested four strategies (Table IV). While distillation with a frozen student provided strong regularization, the optimal performance (R@1: 13.11%) was achieved by removing distillation in favor of a Head-level Alignment loss ('headalign') combined with early, small-scale unfreezing. This demonstrates that precise alignment losses with unfreezing schedules can outperform standard distillation.

TABLE IV
ITERATIVE OPTIMIZATION OF CONVNEXT-BASE ON COCO

Strategy	I2T R@1	Note
Baseline (3-stage Unfreeze)	12.34%	Standard gradual unfreezing
Distill + Frozen	13.10%	Strong regularization via teacher
Distill + Late Unfreeze	12.98%	Unfreezing hurts distillation constraint
Head Align + Early Unfreeze	**13.11%**	**Best strategy, replaces distillation**

2) Preliminary Implementation on TeachCLIP with MSRVTT Dataset: We integrated the ConvNeXt-Base encoder into TeachCLIP for video retrieval on MSRVTT. However, direct migration yielded suboptimal results (Table V, R@1 $\approx$ 2.6–3.4%), even with locked encoders. This indicates that simple encoder substitution is insufficient; effective adaptation requires specialized temporal modeling (e.g., 'absattn') to capture video dynamics, marking a key direction for future optimization.

TABLE V
PRELIMINARY RESULTS OF CONVNEXT-BASED STUDENT ON MSRVTT

Setting	Text-to-Video R@1
MSRVTT-9k (10 epochs)	3.4%
MSRVTT-7k (10 epochs)	2.5%
MSRVTT-7k (16 epochs, Locked Encoder)	2.6%

C. On Teacher Model

To utilize the computationally expensive InternVideo2.5 [8] as a teacher without the overhead of online inference, we implemented an "Offline Feature Extraction - Online Loading" scheme:

a) 1. Offline Feature Pre-computation: To avoid expensive repeated inference, we pre-compute and store InternVideo2.5 features (global, text, frame-level) as a local knowledge base.

b) 2. Data Pipeline Index Alignment: We implemented unique Sample IDs to strictly align shuffled student inputs with the corresponding offline teacher features.

c) 3. Training Logic Adaptation: The training loop was refactored to skip teacher inference, instead utilizing an 'Offline Feature Loader' to feed pre-computed similarity matrices and weights directly into the distillation loss.

1) Results on MSRVTT: We evaluated the performance of the student model distilled from different teachers on the MSRVTT dataset. As shown in Table VI, although InternVideo2.5 demonstrates superior performance as a teacher, the student model distilled from it (TeachCLIP w/ InternVideo2.5) currently underperforms the baseline distilled from X-CLIP. This performance drop is likely due to suboptimal hyperparameter settings (e.g., temperature, loss weights) when aligning with the new teacher's distribution, indicating that further tuning and optimization are required in future work.

TABLE VI
COMPARISON OF DIFFERENT TEACHERS ON MSRVTT

Model	R@1	R@5	R@10
X-CLIP	49.3	75.8	84.8
TeachCLIP (Teacher: X-CLIP)	46.8	74.9	82.9
InternVideo2.5 (Teacher)	55.9	78.3	85.1
TeachCLIP (Teacher: InternVideo2.5)	42.8	70.2	80.6

IV. POST-TRAINING QUANTIZATION OF VAST MODEL USING MS-SWIFT

In this section, we apply post-training quantization methods introduced in MS-Swift [9] to compress the VAST model [10] 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) [9] implements a GPTQ-based quantization framework for post-training quantization. The framework employs GPTQ with INT4 precision, using block-wise sequential quantization with Hessian-based optimization to minimize reconstruction error. Key configuration parameters include: group size of 128 for grouped quantization, symmetric quantization (sym=true), and sequential quantization (true_sequential=true) to preserve inter-layer dependencies. The framework supports module-level selective quantization, enabling quantization to be applied only to specific components (e.g., vision encoder blocks) while preserving full precision for other modules such as text encoders or modality-specific branches in multimodal models.

B. VAST Architecture for Video-Text Retrieval

VAST (Vision-Audio-Subtitle-Text) [10] is an omni-modality foundation model. The VAST framework architecture is illustrated in Figure 4.

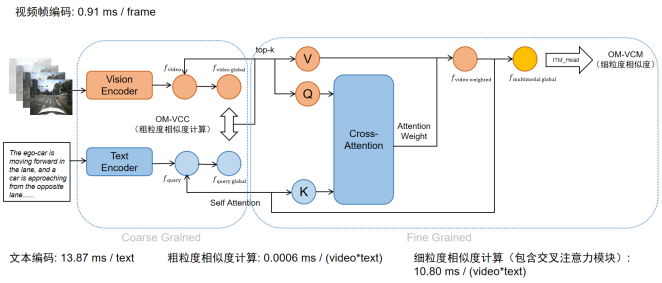


Fig. 4. The framework of VAST.

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 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 apply GPTQ-based post-training quantization to compress the fine-tuned VAST model from FP16 to INT4 precision, achieving approximately 4× model size reduction while preserving retrieval performance.

Quantization Setup: The quantization is performed on the fine-tuned VAST model using MS-Swift's GPTQ implementation. We configure the quantization to target INT4 precision with the following key parameters: group size of 128 for grouped quantization, symmetric quantization (sym=true), and sequential quantization (`true_sequential`=true) to maintain layer dependencies. The quantization is selectively applied only to the vision encoder blocks (`vision_encoder.visual.blocks`), preserving the full precision of other components including the text encoder and audio branches.

Calibration Data: We use 200 calibration samples in JSONL format, each containing text queries and corresponding image frame sequences. During quantization, these samples are forward-passed through the model to collect activation statistics for each layer. The collected activations are used to compute Hessian matrices, which guide the quantization parameter optimization to minimize reconstruction error.

Quantization Process: The quantization proceeds block-by-block through 40 vision encoder blocks. For each block, four linear modules are quantized sequentially: attention QKV projection (`attn.qkv`), attention output projection (`attn.proj`), and MLP layers (`mlp.fc1`, `mlp.fc2`). The process involves collecting activations from calibration data, computing Hessian matrices for weight sensitivity, performing grouped quantization (group size 128) to convert FP16 weights to INT4, and applying error compensation. Weights are transformed from FP16 (16 bits) to INT4 (4 bits) with per-group scaling factors, achieving approximately 4× compression with 4 INT4 values packed into 1 INT16 for storage.

The quantized VAST model maintains competitive retrieval performance with significantly reduced model size and memory footprint, enabling more efficient deployment for real-time video-text retrieval applications.

D. Quantization Results and Comparison

Table VII presents a comprehensive comparison between the original FP16 model and the quantized INT4 model, demonstrating the effectiveness of the GPTQ-based quantization approach.

TABLE VII
COMPARISON OF MODEL STATISTICS BEFORE AND AFTER QUANTIZATION

Metric	FP16	INT4	Ratio
File Size	5.20 GB	1.86 GB	2.80×
Total Parameters	1,396.66M	522.85M	2.67×
Vision Encoder	1,136.44M	262.62M	4.33×
Total Weight Size	5.33 GB	1.99 GB	2.67×

The quantization results demonstrate effective compression: the vision encoder achieved 4.33× compression, closely matching the theoretical 4× ratio for INT4 quantization. The overall model size was reduced from 5.20 GB to 1.86 GB (2.80×), while the total compression ratio (2.67×) is lower due to other components remaining at full FP32 precision as part of the selective quantization strategy.

Table VIII presents the retrieval performance comparison on the suscape test set (autonomous driving scenario dataset) before and after quantization.

TABLE VIII
RETRIEVAL PERFORMANCE COMPARISON ON SUSCAPE TEST SET

Metric	FP16	INT4
Recall@1	64.4	37.3
Recall@5	88.7	55.9
Recall@10	97.2	65.2
Average Recall	83.4	52.8

The quantized model shows expected performance degradation compared to the full-precision model. Despite the performance drop, the quantized model maintains reasonable retrieval capabilities while achieving significant model size reduction.

V. CONCLUSION

In this work, we explored efficient approaches for video-text retrieval in autonomous driving scenarios, focusing on Knowledge Distillation and Post-Training Quantization to address the trade-off between accuracy and computational cost.

In the domain of Knowledge Distillation, we successfully optimized the training efficiency of the TeachCLIP framework by implementing an offline feature pre-computation strategy, achieving a 9.7% speedup. We further investigated the flexibility of the framework by replacing the student encoder with ConvNeXt and the teacher with InternVideo2.5. Our results highlight that while stronger teachers and inductive biases have potential, direct substitution requires careful hyperparameter tuning and specialized temporal modeling to fully realize performance gains.

For Model Quantization, we applied GPTQ INT4 quantization to the VAST model using MS-Swift. We achieved a significant reduction in model size (from 5.20 GB to 1.86 GB), demonstrating the feasibility of deploying large foundation models on constrained hardware. However, the observed drop in Recall@1 (from 64.4% to 37.3%) indicates that aggressive quantization to 4-bit precision requires advanced mitigation strategies, such as Quantization-Aware Training (QAT) or mixed-precision techniques, to preserve fine-grained retrieval capabilities.

Future work will focus on two directions: (1) refining the distillation process for InternVideo2.5 by aligning feature spaces and loss weights more effectively, and (2) exploring recovery strategies for quantized models to minimize accuracy degradation while maintaining efficiency. Ultimately, these efforts contribute to building scalable and responsive video retrieval systems for the autonomous driving industry.

REFERENCES

[1] K. Tian, R. Zhao, Z. Xin, B. Lan, and X. Li, "Holistic features are almost sufficient for text-to-video retrieval," in *CVPR*, 2024.

[2] 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

[3] 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

[4] 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.

[5] H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li, "CLIP4Clip: An empirical study of clip for end to end video clip retrieval," *arXiv preprint arXiv:2104.08860*, 2021.

[6] Y. Ma, G. Xu, X. Sun, M. Yan, J. Zhang, and R. Ji, "X-CLIP:: End-to-end multi-grained contrastive learning for video-text retrieval," *arXiv preprint arXiv:2207.07285*, 2022.

[7] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, "Openclip," Jul. 2021, if you use this software, please cite it as below. [Online]. Available: https://doi.org/10.5281/zenodo.5143773

[8] Y. Wang, X. Li, Z. Yan, Y. He, J. Yu, X. Zeng, C. Wang, C. Ma, H. Huang, J. Gao, M. Dou, K. Chen, W. Wang, Y. Qiao, Y. Wang, and L. Wang, "Internvideo2.5: Empowering video mllms with long and rich context modeling," *arXiv preprint arXiv:2501.12386*, 2025.

[9] 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

[10] 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