CHALMERS
UNIVERSITY OF TECHNOLOGY

Scalable Vision–Language Machine Learning for Semantic Retrieval of Autonomous Driving Logs

Efficient Embedding Learning for Large-Scale Driving Scene Retrieval

Master's thesis in Systems, Control and Mechatronics & Engineering Mathematics Computational Science

MOHAMMAD ALBARHAM, LINUS BERGGREN

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Scalable Vision–Language Machine Learning for Semantic Retrieval of Autonomous Driving Logs

MOHAMMAD ALBARHAM
LINUS BERGGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Scalable Vision–Language Machine Learning for Semantic Retrieval of Autonomous Driving Logs
Efficient Embedding Learning for Large-Scale Driving Scene Retrieval
MOHAMMAD ALBARHAM
LINUS BERGGREN

Industrial Supervisors: Ori Maoz and Altug Altetmek, Aptiv
Academic Supervisor and Examiner: Lars Hammarstrand, Chalmers University of Technology

Master's Thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Multi-view autonomous driving visualization illustrating a surround-camera perception system for semantic retrieval of driving logs.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Scalable Vision–Language Machine Learning for Semantic Retrieval of Autonomous Driving Logs
Efficient Embedding Learning for Large-Scale Driving Scene Retrieval
MOHAMMAD ALBARHAM
LINUS BERGGREN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis studies scalable semantic retrieval of autonomous driving multi-view videos recorded with synchronized multi-camera systems. Using a subset of 3,502 multi-view driving videos from NVIDIA's PhysicalAI Autonomous Vehicles dataset, the work investigates text-to-multi-view video retrieval using natural-language queries and learned cross-modal embeddings. Because the dataset does not contain paired textual descriptions, the proposed pipeline generates pseudo ground-truth captions from sampled video frames using a pretrained vision-language model and extracts frozen text and visual embeddings with jina-clip-v2. These generated captions provide the supervision used for training and evaluation without requiring manual annotation. Lightweight trainable alignment heads are then used to map text and video representations into a shared embedding space, while multi-view representations are constructed through view-level and temporal aggregation. The results quantify the difference between single-view (front camera) and multi-view (front and surrounding cameras) retrieval representations. Extending the representation from a single front-facing camera to six synchronized camera views increases Recall@5 from 71% to 85% and Recall@10 from 81% to 93%, indicating improved separation of ground-truth matches in the learned embedding space. In contrast, the LLM-based semantic similarity score changes only marginally, from 77 to 79, suggesting that both retrieval settings often retrieve semantically related driving scenarios. The experiments further show that temporal sampling can be reduced considerably with only minor changes in retrieval performance, indicating substantial redundancy in densely sampled driving video. Since the supervision is derived from automatically generated captions rather than human-annotated descriptions, the retrieval results should be interpreted with that limitation in mind. Overall, the thesis demonstrates that frozen pretrained encoders combined with lightweight fusion and alignment modules provide a computationally scalable approach for semantic retrieval of large-scale autonomous driving multi-view videos.

Keywords: Multi-view video retrieval, Text-to-multi-view video retrieval, Multi-camera driving logs, Temporal video sampling.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Ori Maoz, for his guidance and support throughout this thesis. His extensive knowledge, insightful feedback, and discussions during our weekly meetings consistently challenged us to think more critically and refine both the research and its presentation.

We also thank Lars Hammarstrand for his valuable input and support, which helped ensure a thorough and rigorous evaluation of the thesis.

Finally, we thank Altug Altetmek for his support within the company and for assisting with practical matters throughout the project, contributing to a productive working environment. For Mohammad Albarham, this work is part of his research work at Chalmers University of Technology and he was funded by the Swedish Institute scholarship during his studies.

Mohammad Albarham and Linus Berggren, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order.

ADAS	Advanced Driver Assistance Systems
CLIP	Contrastive Language-Image Pre-training
BEV	Bird's Eye View
CNN	Convolutional Neural Network
FPS	Frames Per Second
GELU	Gaussian Error Linear Unit
GT	Ground Truth
InfoNCE	Information Noise-Contrastive Estimation
LiDAR	Light Detection and Ranging
LLM	Large Language Model
MLP	Multilayer Perceptron
MRR	Mean Reciprocal Rank
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
ML	Machine Learning
NLP	Natural Language Processing
RGB	Red, Green, Blue
RQ1	Research Question 1
RQ2	Research Question 2
VLM	Vision-Language Model

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables used throughout this thesis.

Indices

i	Camera / view index
t	Temporal index (frame or timestep)
j	Retrieval candidate index
c	Semantic category index used by the LLM judge
K	Retrieval cutoff in Recall@K

Sets

V	Sequence of sampled video frames
$\mathcal{C}$	Set of semantic categories used by the LLM judge
$\{z_j\}_{j=1}^{M}$	Set of embedding vectors for database driving logs

Parameters

τ	Temperature parameter used in contrastive loss and attention pooling
h	Hidden dimension of the attention MLP
d	Embedding dimension
d_v	Video embedding dimension
d_{in}	Input embedding dimension
d_{out}	Output embedding dimension
M	Number of database embeddings / retrieval candidates

Functions

A_{view}	View-level aggregation function
A_{time}	Temporal aggregation function
$\text{sim}(\cdot, \cdot)$	Similarity function (cosine similarity)

Variables

v_t	Video frame at time t
$h_{i,t}$	Frame-level embedding for view i at time t
u_t	Aggregated embedding at time t
z	Log-level video embedding
z_v, z_t	Aligned video and text embeddings
e_v, e_t	Pre-alignment video and text embeddings
q	Query embedding
s_i	Attention score for embedding i
s_c	Semantic similarity score for category c
S	Normalized semantic similarity score (0–100)
α_i	Attention weight for embedding i
α_t	Temporal attention weight
$\beta_{i,t}$	View-level attention weight

Contents

List of Figures

List of Tables

1

Introduction

Advanced Driver Assistance Systems (ADAS) are transforming modern vehicles through safety-critical features such as automatic emergency braking, intelligent speed assistance, and lane keeping assist. Developing and validating these systems requires large-scale driving datasets that capture the diversity of real-world traffic conditions, ranging from routine highway driving to rare safety-critical events. Automotive companies therefore collect thousands of hours of driving data across diverse environments and traffic scenarios. As these datasets continue to grow, efficiently organizing and retrieving relevant driving scenarios becomes a major engineering challenge.

In modern ADAS platforms, driving data is collected using multiple synchronized cameras mounted around the vehicle to provide full surround-view coverage. This multi-camera architecture is motivated by the fact that a single front-facing camera captures only part of the surrounding environment, whereas multiple camera views can provide complementary perspectives of the scene. By observing the environment from different angles, multi-view systems have the potential to capture information that may not be visible from a single viewpoint, resulting in a more comprehensive representation of the driving environment [1].

As these datasets increase in scale and complexity, retrieving specific driving scenarios becomes increasingly difficult. Driving recordings are high-dimensional, temporally redundant, and semantically dense, making manual inspection infeasible at scale. Yet efficient retrieval is central to many engineering workflows, including corner-case discovery, dataset curation, and post-event analysis. What is needed is an automated retrieval system capable of interpreting high-level semantic queries and locating relevant driving scenarios within large-scale databases.

Cross-modal retrieval has advanced significantly through large-scale vision-language pretraining. Models such as Contrastive Language-Image Pre-training (CLIP) [2] learn shared embedding spaces in which semantically related visual data and text are mapped close together, enabling retrieval based on semantic similarity rather than keyword matching. While such approaches have shown strong performance in text-video retrieval, they are primarily designed for single-view video from general-domain datasets. Some recent work has explored retrieval in autonomous driving using Bird's Eye View (BEV) representations [3], though these methods require specialized 3D reconstruction pipelines and dedicated BEV encoders. Efficiently leveraging synchronized multi-view driving recordings directly at the log level remains comparatively underexplored, creating a gap between how driving data is collected and how current retrieval systems are designed.

1.1 Problem formulation

Validation engineers need to retrieve relevant driving logs from large multi-camera archives using free-form textual descriptions. In this thesis, the problem is formulated as learning log-level video-language embeddings that support efficient semantic retrieval of complete driving logs. More specifically, each retrieval item is a multi-view driving log represented by one shared embedding that captures its semantic content across time and across views. The objective is to learn an embedding space in which a textual query and its corresponding multi-view driving log are placed close to each other, while unrelated logs are placed farther apart. This formulation enables scalable retrieval over large driving archives while remaining compatible with synchronized multi-view video data and natural language queries.

This problem formulation is investigated through the following research questions:

- **RQ1:** How large is the retrieval performance gap between single-view and multi-view visual representations for log-level driving scene retrieval under otherwise matched conditions?
- **RQ2:** What minimum temporal sampling density per camera stream is required to achieve retrieval performance close to a denser reference?

1.2 Limitations

This thesis focuses on log-level semantic retrieval for multi-view ADAS video recordings using aggregated representations constructed from sparsely sampled frames and clips. The work does not perform fine-grained temporal localization, clip-level retrieval, or event grounding within individual logs. Rather than addressing detailed 3D scene understanding, tracking, or end-to-end foundation model training, the focus is on evaluating lightweight aggregation strategies for efficient retrieval of relevant driving scenarios. Furthermore, experiments are conducted on a single driving dataset using retrieval-based evaluation metrics, and conclusions are therefore limited to this setting.

1.3 Contributions

The main contributions of this thesis are as follows:

- **Multi-view embedding fusion for driving scene retrieval.** Instead of demonstrating that multi-view representations outperform single-view ones by design, we quantify and analyze the magnitude of this gap under controlled conditions. Extending from a single front-facing camera to six synchronized views improves embedding space structure substantially, with Recall@5 increasing from 71% to 85% and Recall@10 from 81% to 93%, while the LLM-based semantic similarity score improves only marginally. This indicates that

multi-view fusion primarily improves discrimination among semantically similar driving logs rather than expanding semantic coverage.

- **Scalable pseudo ground-truth generation via sparse temporal sampling.** We show that reducing the number of sampled frames per camera from 11 to 4 results in only minor changes in Recall@K while lowering the relative compute cost to approximately 36% of a dense reference. Interpreted as a measure of embedding-space consistency, this finding reveals substantial temporal redundancy in driving video and supports the use of sparse sampling for scalable log-level representation learning.
- **Effective retrieval with frozen pretrained encoders and lightweight alignment.** We demonstrate that frozen pretrained encoders (jina-clip-v2) combined with lightweight trainable projection heads and attention-based pooling modules achieve strong semantic retrieval performance without end-to-end fine-tuning of large backbones. The approach is computationally efficient and generalizes to a dataset of 3,502 multi-camera driving logs drawn from 12 countries.

1.4 Thesis outline

Chapter 2 introduces the background and research context for semantic retrieval in multi-view driving data. Chapter 3 presents the theoretical foundations underlying the work, including embedding spaces, aggregation methods, and cross-modal alignment. Chapter 4 describes the proposed methodology, dataset, embedding pipeline, and retrieval framework. Chapter 5 presents the experimental results together with a discussion of their implications, followed by the conclusion and future work in Chapter 6.

2

Background

2.1 Driving data at scale

The development and validation of ADAS systems require large, diverse collections of recorded driving data. Modern large-scale datasets provide synchronized multi-sensor recordings across thousands of hours, capturing a broad range of environments, traffic densities, weather conditions, and geographic regions [4]. As these collections grow in size and coverage, systematic access to their content becomes increasingly important. Tasks such as identifying failure cases, gathering training data for specific scenarios, and reviewing system behavior after incidents all require the ability to locate semantically relevant recordings within large archives. This is a task that cannot be performed manually at scale.

Driving data is most commonly collected using multiple synchronized cameras mounted around the vehicle to provide full surround-view coverage. A typical configuration includes a front-facing telephoto camera, wide-angle lateral cameras, and rear-facing cameras, together covering the full 360-degree field of view. This multi-camera design reflects a fundamental requirement for reliable perception: objects and events that fall outside the frontal field of view, that are partially occluded, or that are only briefly visible from one angle can still be captured reliably by another camera in the array [1]. As a result, multi-view driving recordings contain significantly richer spatial information than any single camera recording alone.

2.2 Vision-language models and cross-modal retrieval

Vision-language models learn joint representations of images and natural language, enabling cross-modal tasks such as image captioning, visual question answering, and semantic retrieval. Among these, CLIP [2] has been particularly influential. CLIP is trained on large collections of image-text pairs using a contrastive objective that aligns matching pairs in a shared embedding space while separating non-matching ones. The result is a space in which semantically related visual content and text descriptions are placed close together, enabling retrieval by similarity search: a text query can retrieve the most semantically relevant images simply by finding the nearest embeddings.

This shared-space formulation has become the standard paradigm for text-image and text-video retrieval. It decouples query encoding from database encoding, allowing a database of visual embeddings to be indexed offline and queried efficiently

at inference time. The approach scales naturally to large archives and requires no explicit keyword annotation, making it well suited to unstructured multimedia collections such as large driving datasets. To further extend this paradigm across diverse modalities, recent frameworks such as LanguageBind [5] have explored n-modality alignment. LanguageBind employs language as a central binding modality to align video, audio, and depth representations into a unified semantic space, showcasing how multi-modal data can be structured using semantic language guidance.

2.3 Text-video retrieval

Building on vision-language pretraining, a substantial body of work has extended cross-modal retrieval from static images to video. The common approach encodes a video as a sequence of sampled frames, extracts per-frame features using a pretrained image encoder, and aggregates them temporally through mean pooling or a learned temporal module into a single video-level embedding that can be aligned with a text query.

Most text-video retrieval methods are developed and benchmarked on general-domain datasets such as MSR-VTT and ActivityNet Captions, where each example consists of one video clip paired with one or more text descriptions depicting a single activity. Wang et al. [6] show that providing multiple text descriptions per video during training improves retrieval robustness by reducing sensitivity to annotation noise and better capturing the semantic variability of video content. Dai et al. [7] propose a re-ranking strategy using multi-grained cross-attention on top of frozen image encoders, achieving strong retrieval performance without retraining the visual backbone, demonstrating that lightweight alignment modules can effectively complement fixed pretrained features. More recent work explores using vision-language models (VLMs) to automatically generate textual descriptions for video content, enabling retrieval over archives where manual annotation is infeasible [8]. While these methods primarily rely on cascading static image representations over time, native video-language foundation models, such as InternVideo2 [9], scale up visual-semantic pre-training using large-scale video-text data. These models produce native clip-level embeddings that capture complex spatio-temporal relationships directly, though they often entail substantial computational overhead compared to frame-aggregated alternatives.

A shared limitation across these methods is their single-view assumption: each retrieval item is treated as one video stream, and the possibility of multiple synchronized cameras capturing the same scene simultaneously is not considered.

2.4 Retrieval for autonomous driving

Semantic retrieval of driving scenarios has received growing interest as dataset scales have increased. TrafficLens [10] addresses multi-camera traffic video analysis by applying a VLM sequentially across multiple camera feeds, using the output of each camera as context for the next to generate a unified textual description of the full

scene. This demonstrates that language-based summarization across multiple synchronized cameras is feasible and can reduce processing time significantly compared to independent per-camera processing. Other text-based retrieval efforts, such as CARIM [11], propose caption-based autonomous driving scene retrieval schemes that frame the task as text-to-text matching, utilizing large generative models to describe driving events and thus enabling flexible semantic search.

BEV-TSR [3] proposes a text-to-scene retrieval framework for autonomous driving that operates in Bird's Eye View space. The approach fuses multiple surrounding cameras into a unified BEV representation using a dedicated encoder, then aligns this representation with text using a shared cross-modal embedding module. Evaluated on the nuScenes-Retrieval benchmark, the method achieves strong top-1 retrieval accuracy and demonstrates that spatial fusion across cameras improves driving scene retrieval. However, the reliance on a dedicated BEV encoder introduces architectural complexity and a dependency on depth estimation or LiDAR inputs that are not always available.

In parallel with multimodal architectures, self-supervised learning has established powerful baseline representations for autonomous driving. While general-purpose vision models like DINOv2 [12] produce robust visual features that require no manual labels and transfer seamlessly to various downstream perception tasks, driving-specific self-supervised approaches, such as S3PT [13], explicitly leverage scene semantics and spatial-temporal coordinates to optimize feature representations tailored for the long-tailed distributions typical of automotive environments.

Finally, Zhang et al. [14] address the related problem of multi-event video-text retrieval, where a single video contains several distinct events and must be matched against fine-grained text queries. Their work shows that the standard assumption of one-to-one video-text correspondence breaks down when video content is compositionally rich, motivating retrieval methods that can handle more complex semantic structure.

2.5 Research gap

The reviewed methods reveal a clear gap: existing text-video retrieval methods are predominantly designed for single-view, general-domain video and do not account for the complementary spatial structure of synchronized multi-camera recordings. While BEV-TSR demonstrates the value of multi-camera fusion for driving scene retrieval, it requires a dedicated BEV encoder and depth estimation pipeline, adding considerable complexity and restricting applicability to settings with full 3D sensor support. No existing work addresses log-level retrieval from synchronized multi-view RGB recordings using lightweight, encoder-agnostic aggregation that can be applied without committing to a specific 3D spatial representation.

This thesis addresses this gap by proposing a retrieval pipeline that fuses per-camera embeddings from synchronized multi-view recordings into a single log-level representation using attention-based aggregation, built on top of frozen pretrained encoders.

The approach requires only RGB video streams, avoids BEV construction entirely, and is evaluated on a real-world dataset of 3,502 multi-camera driving logs spanning 12 countries.

3

Theory

This chapter outlines the key theoretical ideas that the thesis builds on. It describes how visual and textual data are represented as embeddings, how embeddings are combined across time and camera views, and how different modalities are aligned for retrieval. The focus is on concepts related to text–video retrieval, such as vision–language models, pooling strategies, and contrastive objectives, using pretrained encoders and lightweight alignment mechanisms.

3.1 Vision-language models

Vision-language models jointly process visual inputs (images or video frames) and natural language, enabling tasks such as image captioning, visual question answering, and cross-modal retrieval. In this thesis, VLMs are used in two complementary ways: first, to embed visual and textual inputs into a shared representation space for retrieval using lightweight models such as jina-clip-v2, and second, to automatically generate textual descriptions used as pseudo ground-truth using large models like Qwen 2.5 VL [15].

A common retrieval architecture is the dual-encoder formulation, where separate encoders are used for vision and text:

$$z_v = f_v(x), \qquad z_t = f_t(t),$$

where x denotes an image (or video frame) and t denotes a text description. The encoders map both modalities into a shared embedding space in which semantically matching pairs are placed close together, while unrelated pairs are separated. Similarity is typically measured using cosine similarity, forming the basis for embedding-based retrieval systems.

In addition to retrieval, generative VLMs can be used to automatically produce textual descriptions of visual content. In this work, such a model is used to generate captions for driving videos, enabling scalable construction of paired text-video training examples without manual annotation. This is particularly important in autonomous driving settings, where manually labeling large volumes of multi-camera video data would be expensive.

A key limitation of automatically generated captions is that they are inherently noisy. Generative models may hallucinate objects that are not present or omit small or partially occluded but semantically important objects [16]. These issues are especially relevant in driving scenes, where subtle events may still be safety critical. To reduce noise, structured prompts and constrained output formats are

used to encourage factual and task relevant descriptions. Nevertheless, the generated captions should be regarded as pseudo-labels rather than true ground truth [17].

3.2 Embedding spaces

Modern retrieval systems convert raw visual and textual data into compact vectors called embeddings. These embeddings represent the meaning of the input in a vector space, where semantically similar content lies close together. Embedding models therefore learn how to map high-dimensional inputs such as images, video frames, or natural language into this space in a way that preserves meaningful relationships. For cross-modal tasks like video–text retrieval, both video and text must be represented in embedding spaces that are either already compatible or can be aligned during training. This ensures that a video and its corresponding textual description end up close to each other in the shared space. The following sections explain the theory behind how they represent visual and linguistic information.

3.2.1 Text embeddings

Text embeddings represent words, sentences, or full captions as vectors in a fixed-dimensional space. This makes it possible to compare pieces of text using standard vector operations. Formally, a text embedding model defines a function

$$f_t : T \to \mathbb{R}^d, \tag{3.1}$$

where T denotes a text input (for example a word or a sentence), and $\mathbb{R}^d$ is a latent space with fixed dimension d. The goal is that texts with similar meaning are mapped to nearby vectors.

In practice, many embedding models are trained with contrastive learning objectives, where matching or related text pairs are pulled closer in the embedding space, while unrelated pairs are pushed apart [18]. After encoding, similarity between two texts T_1 and T_2 is commonly measured using cosine similarity:

$$\text{sim}(T_1, T_2) \;=\; \frac{f_t(T_1) \cdot f_t(T_2)}{\|f_t(T_1)\| \, \|f_t(T_2)\|}. \tag{3.2}$$

By measuring the angle between vectors rather than their magnitudes, cosine similarity captures the directional relationship between embeddings. This is particularly suitable for $L2$-normalized embeddings, where all vectors have unit length, ensuring that similarity scores reflect semantic alignment rather than differences in magnitude.

It is also helpful to distinguish between two main families of text embeddings: static and contextual embeddings [19]. Static embedding models assign one fixed vector to each word type. This means the word always gets the same vector, even when its meaning changes depending on the sentence. As a result, static embeddings can be limited for words with multiple meanings.

Contextual embedding models address this limitation by producing embeddings that depend on the surrounding words [19]. In other words, the same word can receive different vectors depending on the context it appears in. Modern transformer-based models create contextual embeddings using self-attention, which allows the model to relate each word to all other words in the input [20]. For sentence-level embeddings, token representations are typically combined into a single vector using a simple pooling method, such as mean pooling or using a special pooled token (depending on the model).

3.2.2 Video embeddings

Video embedding models represent a video as a fixed-length vector that describes its visual content over time. The goal is to capture both what appears in the frames and how the scene changes across time. Formally, a video embedding model defines a function

$$g : V \to \mathbb{R}^d, \tag{3.3}$$

where V is a sequence of video frames and $\mathbb{R}^d$ is a fixed-dimensional embedding space.

A common approach is to build the video representation in two steps. First, a visual encoder extracts information from each frame. Second, a temporal module aggregates frame-level embeddings into one video-level embedding [21]. This separation makes the problem easier, since frame encoders focus on spatial content while the temporal module focuses on combining information across time.

Let the video be written as a sequence of frames $V = (v_1, v_2, \ldots, v_T)$, where v_t is the frame at time step t. A visual encoder is applied independently to each frame:

$$f_v : v_t \to \mathbb{R}^{d_v}. \tag{3.4}$$

For each frame, it produces a frame-level embedding

$$h_t = f_v(v_t), \qquad t = 1, \ldots, T. \tag{3.5}$$

These embeddings capture appearance-related information such as objects, road structure, and overall scene context. Encoders trained with contrastive vision-language objectives often produce embeddings that are semantically meaningful and robust to changes in viewpoint and lighting. However, because each frame is processed independently, the frame embeddings do not contain temporal information by themselves.

To obtain a single embedding for the full video, the sequence of frame embeddings $H = (h_1, h_2, \ldots, h_T)$ is aggregated into a video-level embedding. This can be described using an aggregation function

$$A : (\mathbb{R}^{d_v})^T \to \mathbb{R}^{d_v}, \tag{3.6}$$

and the final video embedding is

$$z = A(H). \tag{3.7}$$

Simple choices for A include mean pooling across time, while more flexible approaches include attention-based pooling, where the model learns to assign higher weight to more informative frames.

3.3 Embedding aggregation

In embedding based retrieval systems, each retrieval item must be represented by a single fixed-dimensional embedding that can be efficiently stored, indexed, and compared. In this thesis, the retrieval item is a complete driving log consisting of multiple camera views recorded over time. Since the number of frames and camera streams can vary between logs, the extracted frame embeddings must be combined into a single compact log level representation through aggregation.

Assume a driving log contains N synchronized camera views and T sampled time steps. For each camera view $i \in \{1, \dots, N\}$ and time step $t \in \{1, \dots, T\}$, a frame level embedding is extracted:

$$h_{i,t} \in \mathbb{R}^{d_v}.$$

The goal is to combine these embeddings into a single log-level embedding

$$z \in \mathbb{R}^{d_v},$$

which can be used for retrieval.

3.3.1 Fixed pooling methods

One way to aggregate embeddings is through fixed pooling operations such as mean pooling or max pooling. Given a sequence of embeddings $(u_1, \dots, u_T)$, where $u_t \in \mathbb{R}^{d_v}$, mean pooling computes the average embedding:

$$z = \frac{1}{T} \sum_{t=1}^{T} u_t.$$

Mean pooling is simple and computationally efficient, and works well when nearby frames contain similar semantic information. This is often the case in driving recordings, where consecutive frames can be highly redundant.

Another common approach is max pooling, which selects the maximum activation in each embedding dimension:

$$z_j = \max_{t \in \{1,\dots,T\}} (u_t)_j, \qquad j = 1, \dots, d_v,$$

where $(u_t)_j$ denotes the jth component of embedding u_t. Unlike mean pooling, max pooling preserves the strongest response observed across the sequence, which can help emphasize important features that appear only briefly.

Fixed pooling methods are simple and efficient, but they either treat all embeddings equally or rely only on the strongest activation. As a result, they may not fully capture the varying importance of different frames or camera views.

3.3.2 Attention-based aggregation

A more flexible alternative is attention-based aggregation, where the model learns how much weight should be assigned to each embedding. Given a sequence of embeddings $\{u_t\}_{t=1}^{T}$, where $u_t \in \mathbb{R}^{d_v}$ and T is the number of embeddings in the sequence, attention pooling computes a weight α_t for each embedding and the pooled representation z as

$$\alpha_t = \frac{\exp(s_t/\tau)}{\sum_{k=1}^{T} \exp(s_k/\tau)}, \qquad z = \sum_{t=1}^{T} \alpha_t u_t, \tag{3.8}$$

where s_t is a learned importance score for embedding u_t, typically computed by a small multilayer perceptron (MLP),

$$s_t = \mathrm{MLP}(u_t), \tag{3.9}$$

and τ is a temperature parameter controlling the sharpness of the attention distribution.

Unlike fixed pooling methods, attention pooling allows the model to focus on the most informative frames or views while reducing the influence of redundant or less relevant observations. This is particularly useful in driving data, where important events may only occur briefly or only be visible from certain camera perspectives.

3.3.3 Hierarchical multi-view aggregation

In multi-camera driving systems, each camera captures only part of the surrounding scene. Objects or events that are unclear or occluded in one view may be clearly visible in another, making it important to combine information across both views and time.

A practical strategy is hierarchical aggregation. First, embeddings are aggregated across camera views at each time step:

$$u_t = A_{\text{view}}(h_{1,t}, \ldots, h_{N,t}),$$

producing one embedding u_t for each time step, where $h_{N,t}$ denotes the embedding of camera view N at time step t. The resulting sequence $(u_1, \ldots, u_T)$ is then aggregated across time:

$$z = A_{\text{time}}(u_1, \ldots, u_T).$$

Both fixed pooling methods and attention-based aggregation can be used at the view-level and temporal level. This hierarchical approach produces a single log-level embedding while preserving complementary information from multiple synchronized camera streams. The specific aggregation strategies evaluated in this thesis are described in Chapter 4.

3.4 Cross-modal alignment heads

In cross-modal retrieval systems, visual and textual inputs are typically processed by separate pretrained encoders. Even when these encoders are trained with similar objectives and produce embeddings of the same dimensionality, their outputs can differ significantly in scale and geometric structure. As a result, embeddings from different modalities are often not directly comparable, even when they describe the same semantic content [22], [23].

To address this, a cross-modal alignment head is introduced. The purpose of this component is to map embeddings into a shared representation space. Rather than retraining large vision or language models end-to-end, modern retrieval pipelines commonly keep backbone encoders frozen and rely on lightweight alignment modules to adapt their outputs to a specific retrieval task [6], [7]. This design choice is computationally efficient and therefore well suited to large-scale datasets.

Formally, let $x \in \mathbb{R}^{d_{\text{in}}}$ denote an embedding produced by a frozen visual or textual encoder. The alignment head implements a function

$$f_{\text{align}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}},$$

which maps the input embedding to an aligned representation $z \in \mathbb{R}^{d_{\text{out}}}$ used for retrieval. Similarity between aligned embeddings is typically computed using cosine similarity. To simplify the geometry of the embedding space, aligned embeddings are commonly L2-normalized so that similarity depends only on angular difference [22], [24].

A common choice for implementing an alignment head is a multilayer perceptron (MLP), which is a neural network composed of linear transformations together with non-linear activation functions [25]. Even with their relatively simple structure, MLPs can model complex relationships and are widely used to map between different embedding spaces.

The simplest alignment consists of an affine transformation,

$$z = Wx + b.$$

This can be sufficient when the relationship between modalities is approximately linear. Even with a shared pretrained encoder, differences in modality processing can lead to mismatches in the embeddings. Since these models are trained on general data rather than ADAS-specific scenarios, subtle or rare driving events may not be well represented [1].

Introducing a hidden layer yields a two-layer MLP, defined as

$$h = \sigma(W_1 x + b_1), \qquad z = W_2 h + b_2,$$

where $\sigma(\cdot)$ denotes a non-linear activation function such as Rectified Linear Unit (ReLU), hyperbolic tangent (tanh), or Gaussian Error Linear Unit (GELU). The

non-linearity allows the alignment head to model more expressive and complex transformations while remaining lightweight.

At the same time, keeping the backbone encoders frozen and training only the alignment head allows the model to retain the broad semantic knowledge learned during large-scale pretraining, while still adapting to the specific retrieval task [7].

3.5 Training objective

The training objective is designed to learn a shared embedding space in which a driving log and its corresponding textual description are mapped close to one another, while unrelated log text pairs are pushed farther apart. Rather than predicting a discrete label, the model is therefore optimized to shape the geometry of the representation space so that similarity scores become meaningful for retrieval. This is naturally formulated as a contrastive learning problem, where matched pairs act as positive examples and non-matching pairs act as negatives. The following subsection introduces the contrastive loss used in this work and explains how it supports cross-modal alignment between video and text embeddings.

The contrastive objective used in this thesis is Information Noise-Contrastive Estimation (InfoNCE) loss, which is a contrastive objective used extensively in representation learning, especially in self-supervised learning frameworks [24].

Its goal is to maximize the similarity between positive pairs (e.g., a driving log and its corresponding textual description) while minimizing similarity with a set of negative samples.

Given a query representation q, a positive key k^+, and a set of negative keys $\{k_j^-\}$, the InfoNCE loss is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \left[\frac{\exp\left(\frac{\text{sim}(q,k^+)}{\tau}\right)}{\exp\left(\frac{\text{sim}(q,k^+)}{\tau}\right) + \sum_{j=1}^{M} \exp\left(\frac{\text{sim}(q,k_j^-)}{\tau}\right)} \right], \tag{3.10}$$

where $\text{sim}(\cdot,\cdot)$ denotes a similarity measure (typically cosine similarity), M is the number of negative samples, and τ is a temperature hyperparameter controlling the distribution sharpness. In the context of this thesis, the query q corresponds to an aligned text embedding, while k^+ and k_j^- correspond to aligned video embeddings.

The InfoNCE loss encourages the model to learn discriminative embeddings by pulling together representations of positive samples while pushing apart negatives. In retrieval tasks, the objective is not to predict a discrete class label but to learn an embedding space in which related items are close together and unrelated items are far apart. InfoNCE is therefore well suited to cross-modal retrieval because it directly optimizes the relative similarity between matching and non-matching pairs.

3.6 Evaluation theory for retrieval

The goal of a retrieval system is to rank a database of items such that the most relevant items for a given query appear near the top of the list. In this thesis, each query has a single ground-truth match in the database (the driving log that the query caption was generated from), and the system returns a ranked list of candidate logs based on embedding similarity. Let q be a query embedding and let $\{z_1, \ldots, z_M\}$ be the embeddings of the M database logs. Cosine similarity $sim(q, z)$ is used to score each candidate, and the candidates are sorted in decreasing order of similarity to produce a ranking.

The primary rank-based retrieval metric used in this work is Recall@K. It measures how often the ground-truth item appears within the top K retrieved results. For each query i, let rank_i be the 1-indexed position of the ground-truth log in the ranked list (so $\text{rank}_i = 1$ means the correct item is ranked first). Recall@K is then defined as:

$$\text{Recall@K} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{\text{rank}_i \leq K\},$$

where N is the number of queries and $\mathbf{1}\{\cdot\}$ is an indicator function that equals 1 if the condition is true and 0 otherwise. Recall@1 corresponds to top-1 accuracy, while larger K values (e.g., $K = 5$ or $K = 10$) measure whether the correct item appears among top K results.

Recall@K reflects how well the embedding space has been learned by measuring whether related items are placed close enough to be retrieved among the top K results. A high Recall@K indicates that the model has successfully structured the embedding space so that matching video–text pairs lie near each other, which in turn suggests effective retrieval behavior. However, it does not capture how strongly items are aligned or how the full ranking beyond the top K is organized.

4

Methodology

In this chapter, we present the dataset, pseudo ground-truth generation using VLMs, embedding extraction, the alignment layer, and how these subsystems are integrated into the overall system.

4.1 Training, validation, and testing datasets

In this work, we utilize camera recordings from NVIDIA's PhysicalAI Autonomous Vehicles dataset. This large-scale dataset contains approximately 1,700 hours of driving records collected across 25 countries and more than 2,500 cities, capturing a wide variety of environments, weather conditions, infrastructure types, and traffic scenarios [26]. The recordings encompass an extensive range of driving contexts, including highways, urban streets, residential areas, rural roads, tunnels, bridges, roundabouts, and railway crossings. In addition to geographic diversity, the dataset spans multiple environmental conditions such as clear, rainy, snowy, foggy, daytime, and nighttime settings, making it well-suited for robust perception and generalization studies.

Each driving sequence in the dataset is stored as a 20-second clip, with a total of 310,895 clips available. These clips include synchronized multi-sensor data, such as multi-camera video streams and LiDAR and radar measurements for a large subset of the recordings. The multi-camera configuration provides 360-degree coverage around the vehicle, offering diverse viewpoints essential for modeling spatial context, occlusion handling, and multi-view consistency [26].

For this thesis, we focus exclusively on the RGB camera recordings and consider two experimental settings: a single-view setting and a multi-view setting. In the single-view configuration, only a front-facing telephoto camera with a narrow field of view of approximately 30° is used, representing a common and computationally efficient baseline.

The multi-view camera setup consists of a front-facing telephoto camera with a 30° field of view, two wide-angle cameras with 120° field of view positioned on the left and right sides of the vehicle, two rear-facing cameras with 70° field of view positioned on the left and right rear, and a rear-facing telephoto camera with a 30° field of view. Together, these cameras provide complementary viewpoints covering the area surrounding the vehicle.

Figure 4.1 illustrates the multi-view camera configuration, where six synchronized cameras are mounted around the vehicle to jointly capture different perspectives of the driving environment.

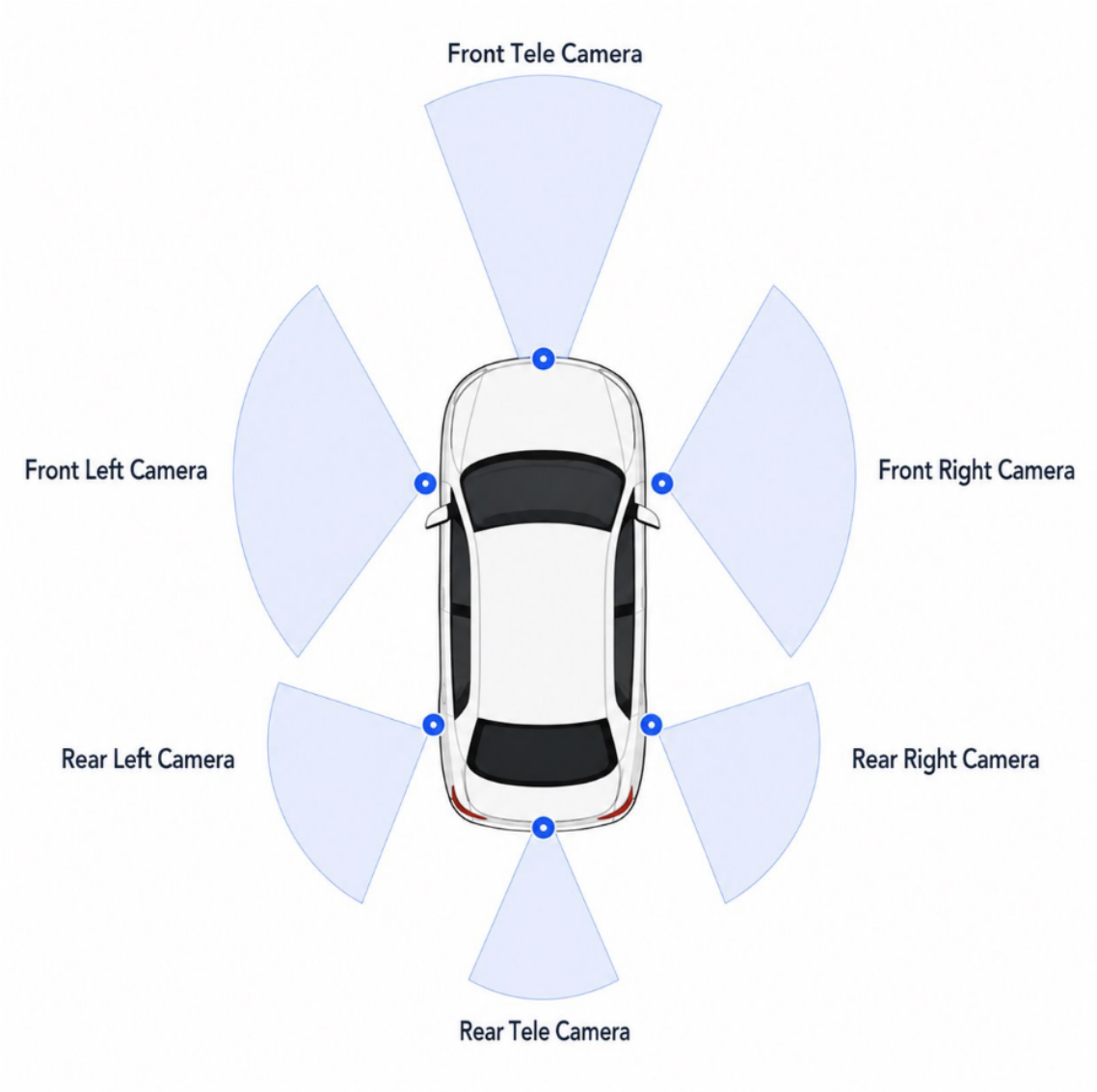


Figure 4.1: Top-view approximate illustration of the vehicle showing the camera setup and their respective fields of view.

Due to computational constraints, we use a subset of the PhysicalAI Autonomous Vehicles dataset for our experiments. From the available recordings, a total of 3,502 driving logs were randomly selected. The geographic distribution of the selected dataset is summarized in Figure 4.2, which shows logs collected from 12 countries. From the selected subset of driving logs, we create a fixed train/test split with 80% of the logs used for training, 10% used for validation, and 10% used for testing. The split is performed using a fixed random seed to ensure reproducibility.

4.2 Pseudo ground-truth generation

The dataset does not contain paired textual descriptions for each multi-view driving log. To enable supervised training, pseudo ground-truth captions are therefore

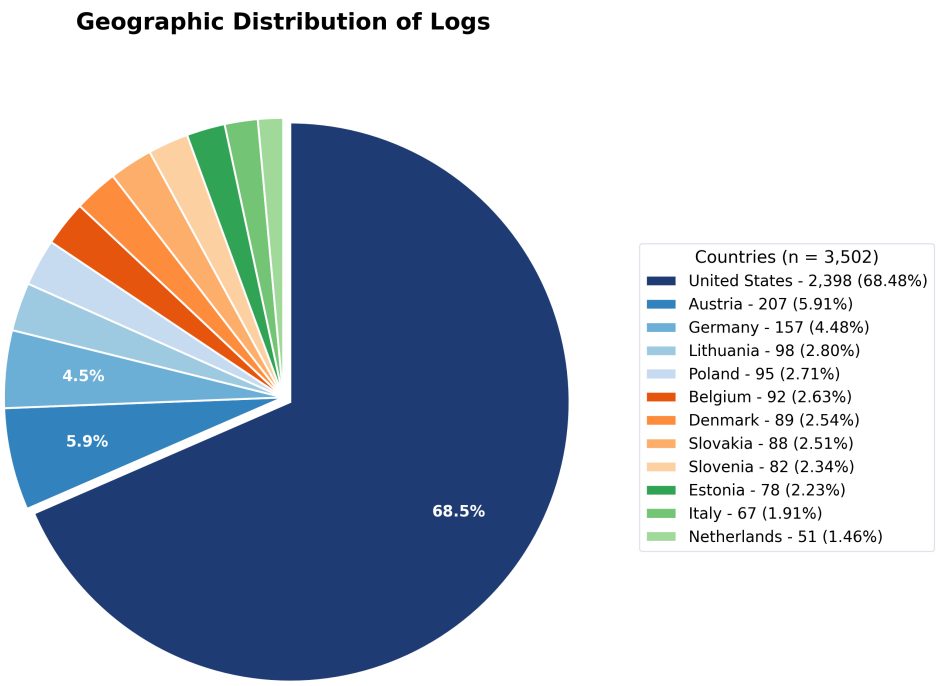


Figure 4.2: Overview of the geographic distribution of the 3,502 logs included in the dataset, collected from 12 countries.

generated from the video data using the Qwen 2.5 VL and Qwen 3.6 models [15], [27]. This makes it possible to construct large-scale aligned text–video pairs without manual annotation and provides the supervision used for training, validation, and evaluation throughout this thesis.

Because the captions are generated using a pretrained vision–language model, specifically Qwen 2.5 VL, the resulting supervision may reflect semantic biases and characteristic phrasing patterns inherited from the captioning model. Consequently, retrieval performance should be interpreted as alignment with these generated descriptions rather than with fully human-annotated scene understanding.

4.2.1 Frame sampling

Each driving log consists of 20 seconds of synchronized multi-view video streams captured by six cameras mounted around the vehicle. Processing all frames with a VLM would be computationally expensive and largely redundant due to the high temporal correlation between consecutive frames.

To balance computational efficiency and descriptive coverage, video frames are sampled at a fixed temporal rate from all cameras. In the single-view setting, frames are sampled only from the front-facing camera, while in the multi-view setting, frames are sampled from each camera. The sampled frames are assumed to provide a representative summary of the visual content across the log duration. In our experiments,

we evaluate four sampling densities and sample $V \in \{11, 8, 6, 4\}$ frames per log (per camera). These values correspond to approximately one frame every 1.8, 2.5, 3.3, and 5 seconds, respectively, which matched the cadence at which the scene-level semantic content of the driving logs typically changed in our visual inspection; denser sampling mostly added near-duplicate frames, while sparser sampling risked missing short events. The sampled frames are evenly distributed across the 20-second driving log, spanning the entire duration of the recording.

4.2.2 Caption generation with VLM

Each sampled frame is provided as input to a pretrained vision–language model, which generates a natural language description of the observed scene using our prompt instructions. We use the Qwen 2.5 VL 7B Instruct model [15] as the captioning model due to its practical trade-off between caption quality and computational cost, enabling scalable caption generation over a large volume of driving data.

Captions are generated independently for each sampled frame using a task-specific prompt designed to encourage factual, scene-level descriptions focused on visually observable elements; the exact captioning prompt used in this work is provided in Section A.1 of Appendix A. In the multi-view case, this process is applied separately to frames from each camera stream at a given timestamp, resulting in a set of frame-level captions, one for each camera view. At this stage, no temporal or multi-view aggregation is performed. Caption generation was performed offline during dataset preparation and required approximately 320 GPU-hours for the full dataset, highlighting the importance of efficient frame sampling strategies.

Figures 4.3 and 4.4 illustrate examples of generated captions for multi-view camera frames at a specific timestamp. Each caption describes the visual scene according to the instructions defined in the prompt presented in Section A.1 of Appendix A.

Front left camera

The scene depicts a quiet suburban street during daytime under clear skies with scattered clouds. The road appears to be a two-lane residential street with a smooth asphalt surface. A silver Honda Civic is prominently positioned in the foreground, angled slightly towards the viewer as it moves along the road. In the background, there's a pinkish-brown apartment building with multiple windows and balconies, partially obscured by trees and shrubs. To the left, a section of a gray industrial or commercial building with a metal fence is visible. The sidewalk is lined with greenery, including a large tree and smaller bushes, adding to the suburban feel. Power lines run above the street, and utility poles are present at regular intervals. The lighting suggests it is either morning or late afternoon, given the long shadows cast by the trees and poles. No pedestrians or cyclists are visible in the scene, and no construction or special vehicles stand out. The overall environment is calm and typical of a residential neighborhood.

Front camera

The scene depicts a multi-lane urban street with moderate traffic flow. In the foreground, a silver sedan is positioned in the right lane, while a green-covered truck is in the left lane, slightly ahead. A red car is visible further down the road in the left lane, and another vehicle can be seen in the distance, also in the left lane. The road appears to have a smooth asphalt surface with clear lane markings. On the left side of the street, there are buildings, including one with a sign advertising "Apollo Electric Inc." and another with a banner for "Auto Accident Attorneys." The area has a mix of vegetation, with trees and shrubs lining the sidewalk. On the right side, there's a pedestrian crossing sign indicating "AHEAD," and a chain-link fence runs along the edge of the sidewalk. The background shows a hilly landscape under a partly cloudy sky, suggesting it might be late afternoon or early evening due to the long shadows cast by the poles and buildings. The overall environment suggests a typical suburban or semi-urban setting with a mix of commercial and residential areas.

Front right camera

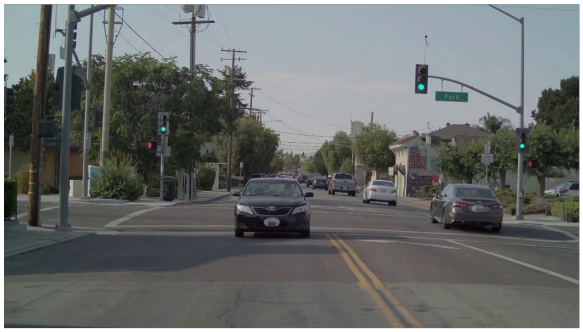
The scene depicts a quiet residential street viewed from inside a vehicle. The road appears to be a two-lane residential street with a smooth asphalt surface. A wooden fence runs along the right side of the road, separating it from a house with a light-colored exterior and a small porch area. The house has a red door and a small window above it. Trees with dense foliage line the street, casting shadows over parts of the road and sidewalk. A chain-link fence is visible further down the street, partially obscuring a view of what seems to be a yard or open space. The sky is clear with a few scattered clouds, suggesting a bright but slightly overcast day. The lighting indicates it could be late afternoon or early evening, as the shadows are long and angled. There are no pedestrians, cyclists, or other vehicles visible in the immediate vicinity. The overall environment suggests a calm suburban neighborhood.

Rear Left camera

The scene depicts a suburban intersection during daytime under clear skies with ample sunlight. The road is a two-lane street with a yellow dividing line, and the surface appears to be asphalt. A traffic light is visible at the intersection, showing green for the approaching traffic. Several cars are present, including a black sedan in the left lane, a white sedan further back, and a dark-colored car in the right lane. Additional vehicles can be seen parked along the curb on the right side of the image, near a building with a sign reading "Coin Laundry."The environment includes palm trees and other greenery scattered around the area, suggesting a warm climate. The buildings appear to be commercial establishments, with one prominently displaying a "Coin Laundry" sign. The road is lined with streetlights, and there are no pedestrians or cyclists visible in the scene. The overall atmosphere is calm, with no signs of congestion or heavy traffic. The lighting suggests it is either late morning or early afternoon, given the angle and intensity of the sunlight.

Figure 4.3: Example of multi-view driving scenes together with their generated captions produced by Qwen 2.5 VL 7B instruct model. The figure shows four surrounding camera perspectives: front left camera view, front camera view, front right camera view and rear left camera view.

Rear camera



The image depicts a suburban intersection during daytime under clear skies. The road appears to be a four-lane street with a center yellow dividing line, suggesting it's a divided highway. The surface looks smooth and well-maintained, with visible lane markings and traffic signals at each corner. \n\nIn the foreground, a black sedan is stopped at the intersection, facing the camera. To its right, a silver sedan is also stopped, slightly further back. A white sedan is positioned further down the road, closer to the center of the image, and appears to be moving forward. Another vehicle, possibly a light-colored SUV, is partially visible behind the white sedan. The environment includes trees lining both sides of the road, providing shade and greenery. There are utility poles with wires running along the street, and a few buildings are visible in the background, including a retail store with a sign indicating it might be a grocery or convenience store. The road signs include one that reads "Park," suggesting the name of the street or a nearby location. Traffic lights are present at the intersection, with the green light illuminated for the vehicles facing the camera. The overall lighting suggests it is either morning or late afternoon, given the angle and length of the shadows cast by the trees and poles

Rear right camera

The scene depicts a suburban street intersection during daylight hours under clear skies. A gray Hyundai Tucson is positioned at the center of the image, slightly angled towards the viewer as it approaches the intersection. The car appears to be in motion, with its headlights off, suggesting it's daytime. The road surface looks dry and smooth, with visible lane markings indicating a two-lane road configuration. \n\nTo the left, there is a residential area with houses featuring light-colored siding and sloped roofs. A small patch of overgrown grass and weeds is visible near the curb, adding to the suburban feel. Utility poles and power lines run parallel to the road, and a green street sign reading "Park" is mounted on one of the poles. \n\nIn the background, another vehicle, possibly a black sedan, is seen further down the road, heading away from the camera. The overall lighting suggests late afternoon or early evening, given the long shadows cast by the objects and the warm tone of the sunlight. There are no pedestrians or cyclists visible in the scene, and no construction or special vehicles are present. The environment appears calm and typical of a quiet neighborhood.

Figure 4.4: Example of multi-view driving scenes together with their automatically generated captions produced by Qwen 2.5 VL 7B instruct model. The figure shows two surrounding camera perspectives: rear camera view, and rear right camera view.

4.2.3 Caption aggregation and post-processing

To obtain a single textual description representing an entire multi-view video driving log, the frame-level captions are aggregated into a log-level description. In the multi-view setting, captions are aggregated across both time and camera streams to form a single summary of the scene. A practical challenge is that frame-level captions often contain repeated information, since consecutive frames are visually similar.

We perform caption aggregation in three steps. First, we collect the frame-level captions from all sampled time steps (and from all cameras in the multi-view setting). Second, we aggregate captions across cameras to produce a single description per timestep. Third, we aggregate the timestep-level descriptions into one log-level caption representing the full driving log. The prompts used for caption aggregation are provided in Section A.2 of Appendix A. For example, the following description was generated by aggregating the captions from the six camera views shown in Figures 4.3 and 4.4 at a specific timestamp.

Aggregated scene description across synchronized camera views at a single timestamp.

The video captures a series of scenes across different suburban environments during the daytime under clear skies. In one frame, a quiet residential street features a silver Honda Civic moving along a two-lane road lined with greenery and apartment buildings. Another frame shows a multi-lane urban street with moderate traffic, including a silver sedan and a green truck, alongside commercial signage and a pedestrian crossing sign. A suburban intersection is depicted with a gray Hyundai Tucson approaching, surrounded by residential houses and a "Coin Laundry" sign. The final frame presents a four-lane divided highway with vehicles stopped at an intersection, bordered by trees and a nearby retail store, all under the warm glow of late afternoon or early evening sunlight.

After generating a scene description for each timestamp by aggregating information across the synchronized camera views (one example is shown above), the timestamp-level descriptions are further aggregated across time. Using the aggregation prompt shown in Section A.2 of Appendix A, a final log-level description is generated for the complete driving log. The resulting output description for this example is shown below.

Aggregated log-level description generated across all timestamps.

Residential two-lane roads and multi-lane urban streets are featured throughout the video, capturing various driving scenarios across different times of day and environments. Vehicles such as sedans, pickups, and SUVs are frequently observed, often navigating through intersections and streets with clear lane markings and traffic lights. The scenes include retail stores, laundromats, and commercial buildings, with trees and greenery lining the streets. Pedestrians and cyclists are not visible in any of the frames. The weather conditions range from clear skies with scattered clouds to overcast skies, and the road surfaces are generally clean and well-maintained. Construction and roadworks are occasionally noted, particularly in suburban intersections. Traffic lights and signs, including speed limit signs and pedestrian crossing signs, are consistently present, ensuring safe navigation.

After aggregation, the global log-level description is further refined through a post-processing step using Qwen 3.6 [27] and the prompt described in Section A.3 of Appendix A. This step removes redundant phrasing, undesirable expressions, and less relevant information to make the text more suitable for retrieval. In particular, it eliminates explicit statements about the absence of objects or events (e.g., *"no pedestrians"* or *"no traffic signs"*) and reduces speculative reasoning about scene elements that are not clearly visible in the driving log. The refinement is guided by simple instructions that preserve factual content while improving clarity and consistency without adding new details.

The resulting multi-view log-level text description is used as pseudo labels and paired with the corresponding driving multi-view video log during training, validation, and testing of the retrieval system.

Log description after post-processing.

Residential two-lane roads and multi-lane urban streets featuring sedans, pick-ups, and SUVs navigating intersections with clear lane markings and traffic signals. Commercial environments include retail stores and laundromats bordered by greenery, with road surfaces remaining clean and well-maintained under varying sky conditions from clear to overcast. Traffic infrastructure consistently displays speed limit signs, pedestrian crossing indicators, and active traffic lights to support safe navigation. Occasional construction and roadworks are observed within suburban intersections, introducing dynamic lane configurations. Vehicle types and environmental factors such as scattered clouds and tree-lined streets provide diverse contextual data for scenario matching.

Figure 4.5 summarizes the full pseudo ground-truth generation pipeline used in this work. It illustrates the progression from frame-level multi-view captions to timestamp-level aggregation, log-level description synthesis, and final post-processing for improved consistency.

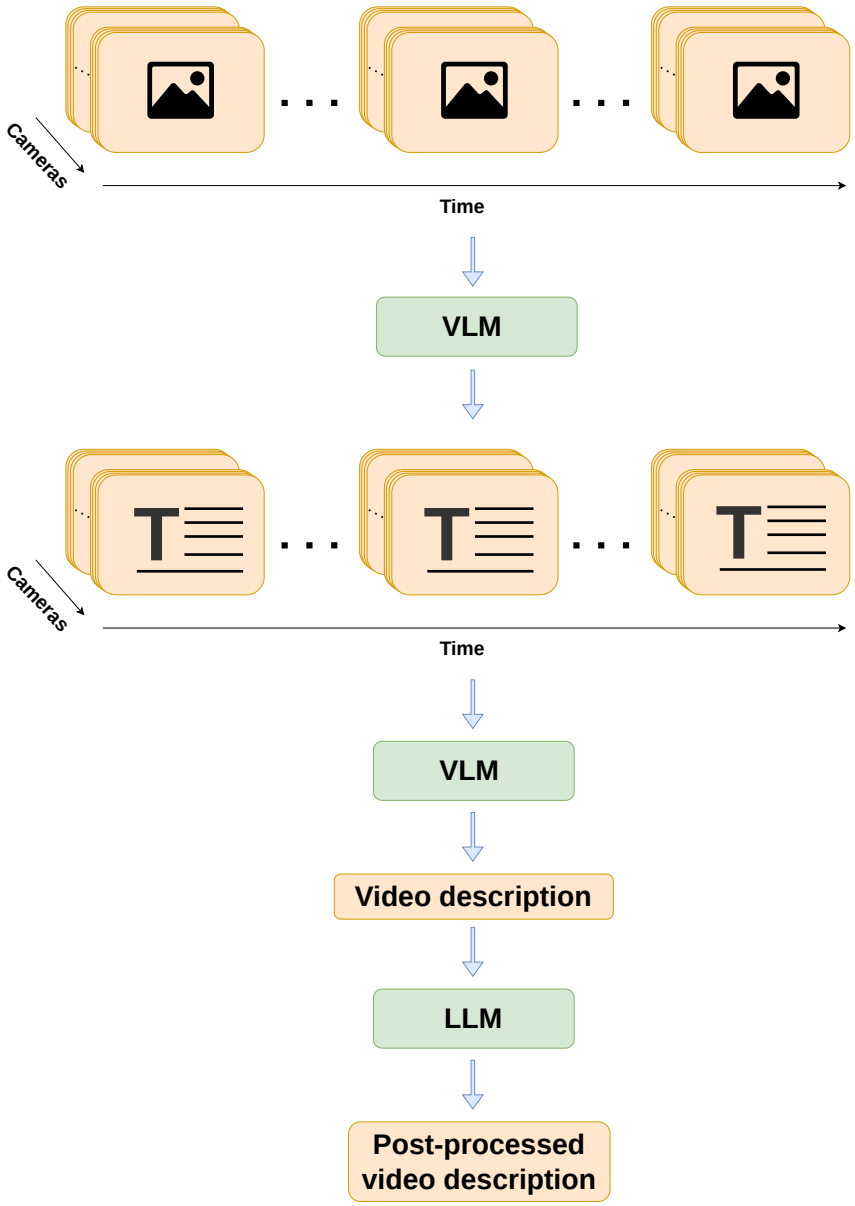


Figure 4.5: Overview of the pseudo ground-truth generation pipeline. Multi-view camera frames are first processed independently by a vision-language model (VLM) to generate frame-level captions. Captions from different camera views are then aggregated into timestamp descriptions, which are subsequently combined into a single log-level description. A final post-processing step is applied to improve consistency and reduce hallucinated content.

4.3 Embedding extraction

After generating captions for the multi-view videos, the vision and text modalities can be connected in the embedding space. Therefore, several vision-language embedding models were evaluated in initial experiments to assess their suitability for

large-scale retrieval of autonomous driving logs. These models differ in architectural design, embedding dimensionality, and pretraining data, which can influence both retrieval performance. A summary of the evaluated model and alignment combinations is provided in Table B.1 in Appendix B. Based on these experiments, jina-clip-v2 was selected for the main pipeline due to its strong and stable retrieval performance. Both its text and vision encoders were utilized throughout the implementation. Figure 4.6 shows the embedding extraction pipeline.

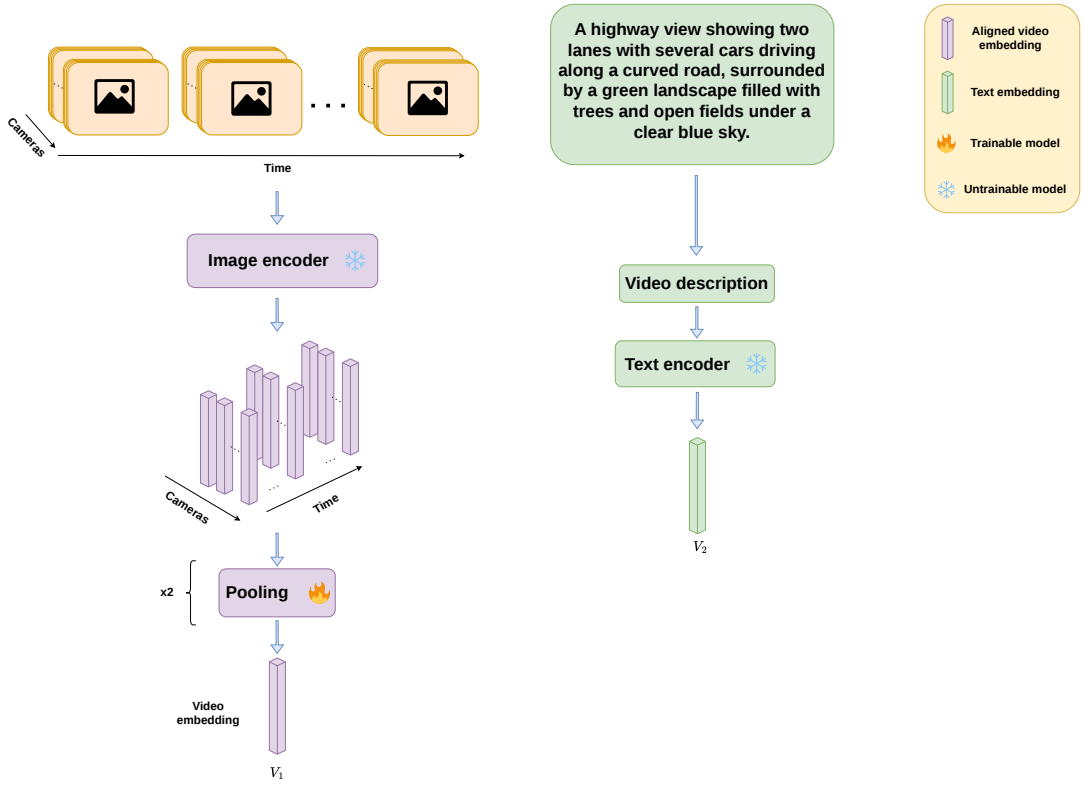


Figure 4.6: Embedding extraction pipeline using jina-clip-v2. Multi-view video frames are encoded by the vision encoder and aggregated into a single video embedding, while the corresponding log description is encoded by the text encoder into a text embedding. Both embeddings are projected into the same representation space and used for retrieval based on cosine similarity.

4.3.1 Vision embedding model

For the visual modality, each driving log is first decomposed into individual frames using the sampling procedure described in Section 4.2.1. Each frame is then preprocessed and passed through the jina-clip-v2 vision encoder to obtain a frame-level embedding vector. Since a multi-view video contains multiple frames across multiple camera streams, these frame embeddings are aggregated across camera views at each selected time step and then aggregated over time to produce a single video-level embedding representation, as described in detail in Section 4.4.

This allows us to summarize the visual content of the entire video into one fixed-

length vector, which can later be compared with the corresponding text embedding. The vision backbone of jina-clip-v2 used for frame-level embedding extraction is summarized in Table 4.1. jina-clip-v2 was selected because it provides a shared vision-language embedding space and has demonstrated strong retrieval performance across a range of multimodal tasks.

Model	Checkpoint	Input resolution	Embedding dim.
jina-clip-v2	`jinaai/jina-clip-v2`	512x512	1024

Table 4.1: Vision backbone used for frame-level embedding extraction.

4.3.2 Text embedding model

For the textual modality, the input caption is tokenized and fed into the text encoder of jina-clip-v2. The output of the text encoder is a fixed-length embedding vector that represents the semantic meaning of the caption. If the text exceeds the maximum context length supported by the model, it is truncated accordingly. The text encoder used for caption embedding extraction is summarized in Table 4.2.

Model	Checkpoint	Max tokens	Embedding dim.
jina-clip-v2	`jinaai/jina-clip-v2`	512	1024

Table 4.2: Text encoder used for caption embedding extraction.

After extraction, embeddings are L2-normalized and compared using cosine similarity for retrieval.

4.4 Multi-view aggregation

To construct a single log-level representation suitable for retrieval, frame-level embeddings are aggregated across both synchronized camera views and time. Since different camera perspectives and timesteps may contain varying amounts of semantic information, we evaluate both fixed pooling methods and learnable attention-based aggregation. Aggregation is performed hierarchically: embeddings are first combined across camera views at each timestep and then aggregated temporally to produce a final log-level embedding.

4.4.1 Aggregation architecture

At each sampled timestep, embeddings from all available camera views are aggregated into a single timestep representation. The resulting sequence of timestep embeddings is then aggregated across time to produce the final log-level embedding used for retrieval. This hierarchical structure allows the model to combine complementary spatial information across camera views while simultaneously summarizing temporal information across the driving log.

For both view-level and temporal aggregation, we evaluate mean pooling, max pooling, and learnable attention pooling. The pooling strategy used at each stage is configured independently, allowing different combinations of fixed and learnable aggregation methods to be evaluated.

For learnable aggregation (attention pooling) [28], the attention weights and weighted aggregation follow the generic formulation introduced in Section 3.3.2, using the temperature-scaled softmax and weighted sum in Eq. 3.8. In the proposed architecture, the score function is instantiated with a lightweight two-layer multilayer perceptron (MLP). Given a set of input embeddings $\{\mathbf{e}_1, \ldots, \mathbf{e}_N\} \in \mathbb{R}^D$, each embedding is passed through a LayerNorm-Linear-Tanh-Linear scoring network:

$$s_i = \mathbf{w}_2^\top \tanh\left(\mathbf{W}_1 \cdot \mathrm{LayerNorm}(\mathbf{e}_i) + \mathbf{b}_1\right) + b_2 \tag{4.1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h \times D}$ and $\mathbf{w}_2 \in \mathbb{R}^h$ are learned parameters with hidden dimension h.

This formulation allows the model to assign higher weight to informative views or timesteps while reducing the influence of redundant observations. Separate attention modules with the same scoring architecture are used for view-level and temporal aggregation.

For learnable attention pooling, aggregation is performed during training to allow gradients to propagate through the attention modules. In contrast, fixed pooling methods can be precomputed offline as a pre-processing step.

Additional experiments comparing pooling strategies, alignment architectures, and embedding model combinations are provided in Appendix B.

Figure 4.7 shows the attention pooling architecture used at both aggregation stages. It summarizes how camera-view and timestep embeddings are scored with a two-layer MLP, normalized with a temperature-scaled softmax, and combined through a weighted sum into a single video representation.

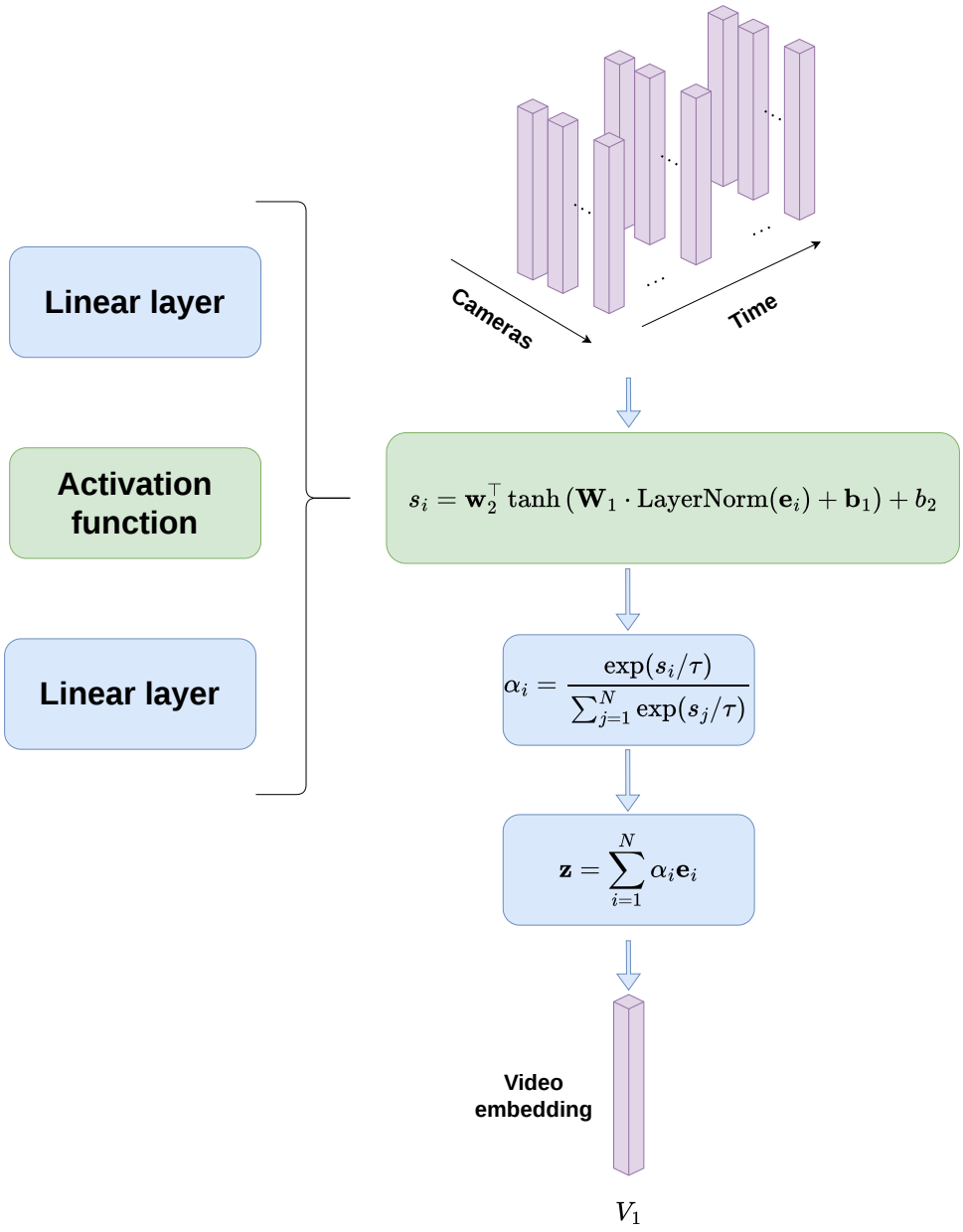


Figure 4.7: Architecture of the learnable attention pooling module. Input embeddings from multiple camera views and timesteps are passed through a two-layer MLP to generate attention scores. The scores are normalized using a temperature-scaled softmax, and the resulting attention weights are used to compute a weighted sum, producing a single aggregated video embedding.

4.4.2 Hyperparameter selection

The pooling method and its associated parameters, such as hidden dimension h and softmax temperature τ, are selected using grid search with nested cross-validation. The dataset is first split into 80% training, 10% validation, and 10% test partitions

using a fixed random seed. The separate 10% validation partition is not used inside the grid search. Instead, only the 80% training partition is further divided into $K = 3$ folds. For each candidate hyperparameter configuration, 3-fold cross-validation is performed within this training partition to estimate retrieval performance. The best-performing configuration is then retrained on the full 80% training partition, while the held-out 10% validation partition is used for early stopping and best-checkpoint selection. The 10% test partition is used only for the final reported evaluation.

The final configuration used for learnable attention pooling at both the view-level and temporal aggregation stages is shown in Table 4.3. Detailed grid-search results are reported in Table B.2 in Appendix B. The strongest attention/mean variants are tightly clustered between $R@1 = 0.323$ and 0.333 and between $R@5 = 0.620$ and 0.628, whereas max/max pooling drops to $R@1 = 0.249$ and $R@5 = 0.540$.

Parameter	**View-level aggregation**	**Temporal aggregation**
Pooling method	`attention (learnable)`	`attention (learnable)`
Hidden dimension (h)	128	128
Softmax temperature (τ)	1.0	1.0
Scoring architecture	LayerNorm–Linear–Tanh–Linear	LayerNorm–Linear–Tanh–Linear
Input normalization	L2	L2

Table 4.3: Final configuration used for learnable attention pooling.

4.5 Alignment layer

The purpose of the alignment layer is to map video and text embeddings into a shared embedding space. Although the selected vision and text encoders are pretrained using contrastive objectives, their raw output embeddings are not guaranteed to be optimally aligned for log-level retrieval in the autonomous driving domain, especially after hierarchical aggregation. Differences in modality, pretraining data, and embedding geometry can therefore lead to systematic mismatches when embeddings are compared directly.

To address this, we introduce a lightweight trainable alignment head after the multi-view aggregation (pooling). Let $e_v \in \mathbb{R}^{d_v}$ denote the aggregated log-level video embedding after temporal aggregation and, in the multi-view setting, view aggregation. The alignment layer is applied after aggregation as follows:

$$z_v = a_v(e_v), \tag{4.2}$$

which projects the vision modality and adapts it into a common embedding space of dimension d_{out} with the text modality, $e_t \in \mathbb{R}^{d_t}$. The aligned embeddings are normalized to unit L_2 norm, and cosine similarity is used to compute retrieval scores with the text embedding.

Rather than assuming a single optimal alignment design, we evaluate multiple alignment head architectures. The first is a linear projection,

$$z = Wx + b, \tag{4.3}$$

where x denotes the input embedding (e_v). The second is a two-layer multilayer perceptron (MLP),

$$h = \sigma(W_1 x + b_1), \qquad z = W_2 h + b_2, \tag{4.4}$$

where $\sigma(\cdot)$ is a non-linear activation function. For the MLP-based alignment head, we evaluate both GeLU and Leaky ReLU activations.

To identify the most suitable alignment strategy for this retrieval setting, these alternative heads are compared experimentally using the same embedding backbone and training setup.

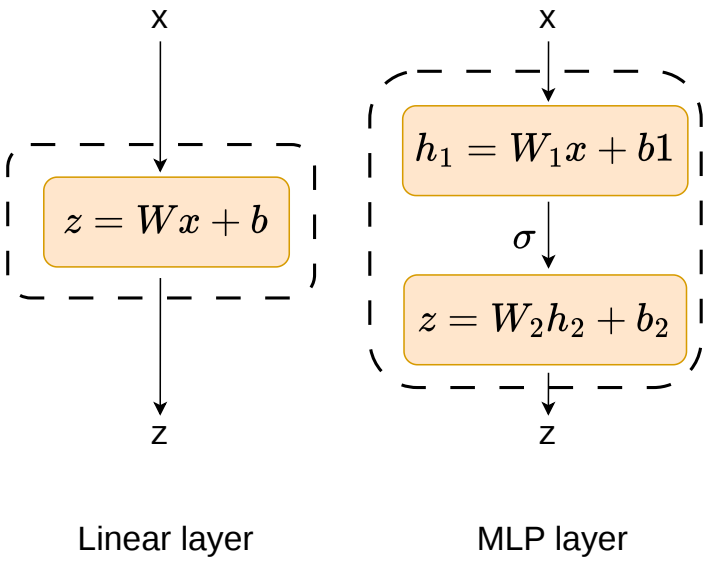


Figure 4.8: Alignment layer architectures evaluated in the model: a single linear projection and a two-layer MLP alignment head used to map video and text embeddings into a shared embedding space.

Our experiments show that a single linear projection trained with an InfoNCE contrastive loss provides the strongest and most consistent alignment results among the best-performing backbone combinations, as summarized in Table B.1 in Appendix B. This suggests that the pretrained embeddings produced by the vision and text encoders are already well-structured, and that retrieval performance primarily benefits from a simple geometric adjustment rather than a more expressive non-linear transformation. In contrast, the MLP-based alignment heads do not provide consistent improvements across the evaluated settings and appear more prone to overfitting under the pseudo-label supervision used in this work.

Based on these findings, we adopt a linear alignment layer combined with the InfoNCE loss as the final configuration for all experiments reported in this thesis. This choice offers a favorable trade-off between retrieval performance, training stability, and computational efficiency, while preserving the semantic structure learned by the frozen backbone encoders.

Figure 4.9 provides a conceptual view of this alignment objective. It illustrates how the trainable transformation brings semantically corresponding video and text embeddings closer in a shared representation space.

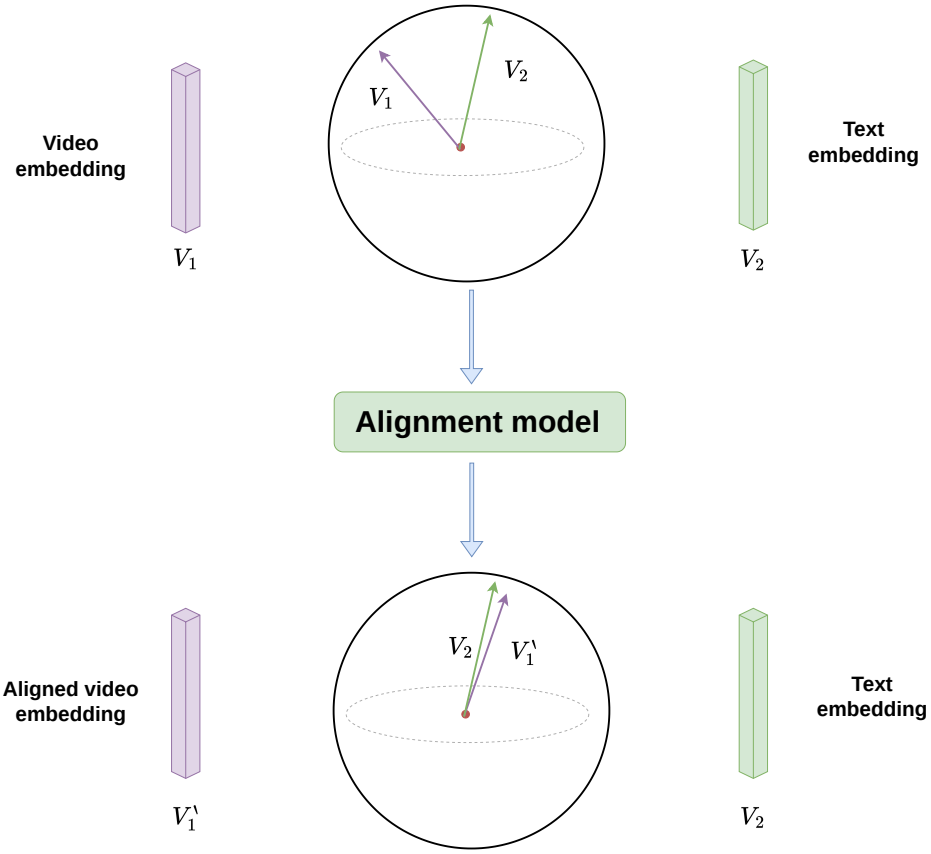


Figure 4.9: Conceptual illustration of cross-modal alignment. The alignment model learns a transformation that maps video embeddings into a shared embedding space, reducing the distance between semantically corresponding video and text representations.

4.6 Model training and inference

This thesis proposes an embedding-based retrieval system for driving logs recorded using synchronized multi-camera setups. The goal is to retrieve semantically similar driving logs based on a textual description. Figure 4.10 illustrates the full training pipeline, showing how raw video data is transformed into a log-level embedding used for retrieval.

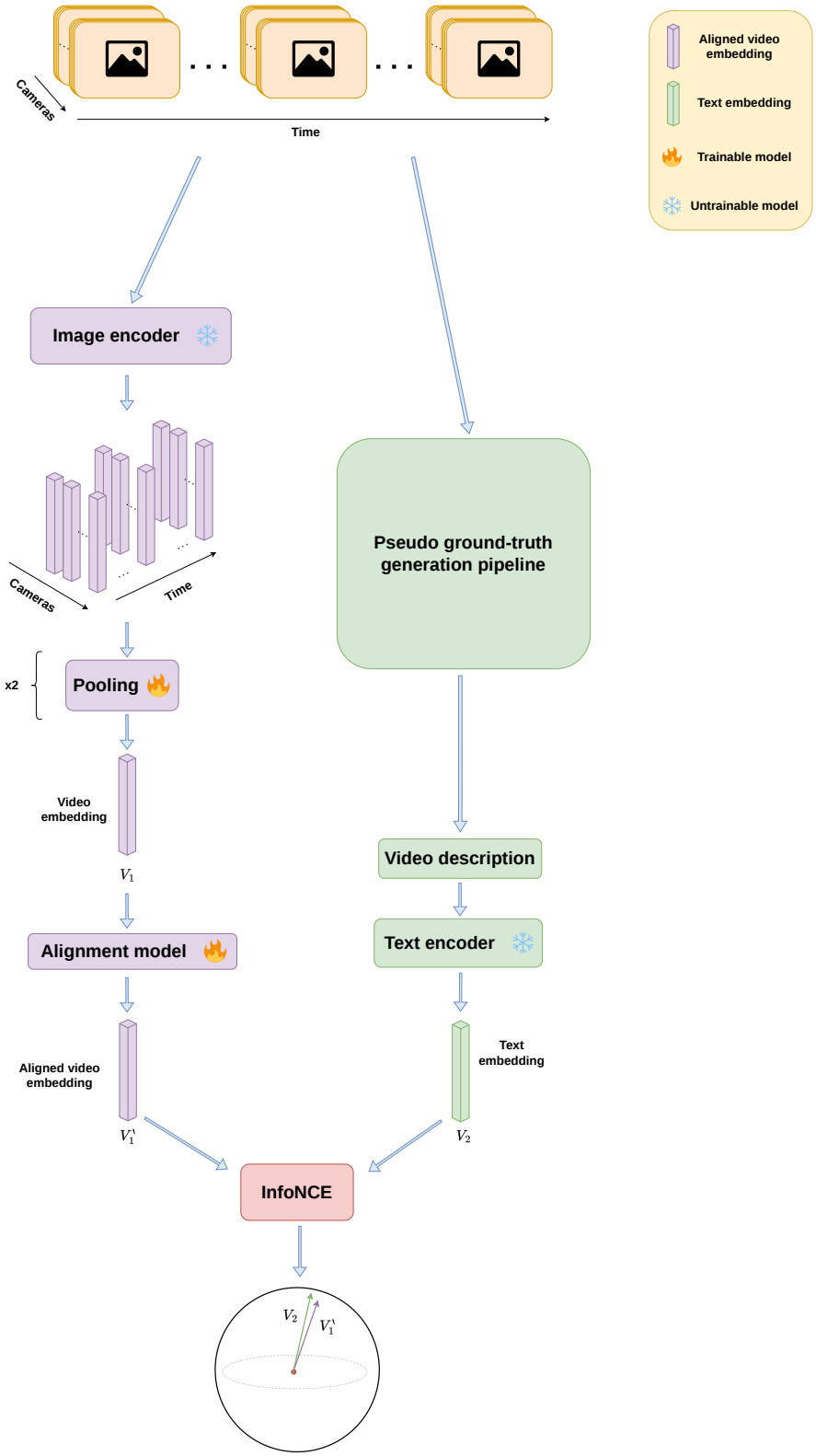


Figure 4.10: System-level overview of the main framework used for training, including all key stages on both the video and text sides.

4.6.1 Model training

The system is designed around three main stages: pseudo ground-truth generation, embedding extraction, and retrieval model training. To ensure scalability and computational efficiency, all large foundation models, visual encoders, and text encoders are kept frozen. Only lightweight alignment and fusion models are trained. This design choice allows the system to be applied to large-scale driving datasets without the need for expensive end-to-end training.

In the first stage, pseudo ground-truth text descriptions are generated from video using the Qwen 2.5 VL model [15]. Each driving log consists of 20 seconds of synchronized video streams captured from multiple cameras mounted around the vehicle. A limited number of frames are sampled and processed by a pretrained VLM to generate scene descriptions. In the single-view setting, captions are generated from a front-facing camera only, while in the multi-view setting, captions are generated per camera, across six cameras, and aggregated into a single log-level description. These descriptions are used as supervision during training and evaluation. Because captions are automatically generated, results reflect retrieval performance under pseudo-label supervision.

In parallel, visual and textual embeddings are extracted using frozen encoders. Sampled video frames are encoded and then aggregated across camera views and time to produce log-level video embeddings, while the log-level captions are encoded to obtain corresponding text embeddings. In the final stage, video and text embeddings are mapped into a shared embedding space using trainable alignment layers. For multi-view models, per-camera video embeddings are fused into a combined log-level representation.

The primary hardware utilized for model training and evaluation was an NVIDIA RTX 6000 Ada Generation GPU that is equipped with 48 GB of VRAM.

4.6.2 Training reproducibility

To make the final reported training setup reproducible, Table 4.4 summarizes the fixed training settings used for the thesis configuration. Pooling-specific hyperparameters are listed separately in Table 4.3. Unless otherwise stated, the pretrained vision and text backbones remained frozen and only the alignment layer and learnable pooling modules were optimized.

Setting	Value used in the final reported configuration
Dataset split	80% training, 10% validation, 10% test
Split seed	42
Validation usage	Held-out 10% validation split used for early stopping and best-checkpoint selection
Pooling search protocol	3-fold cross-validation performed only within the 80% training partition
Vision backbone	`jinaai/jina-clip-v2`
Text backbone	`jinaai/jina-clip-v2`
Alignment direction	Video-to-text projection
Final alignment head	Linear projection
MLP ablations	Hidden dimension 1024; activations GELU and Leaky ReLU. The final reported model uses the linear head
Loss function	InfoNCE
Optimizer	AdamW
Learning rate	0.01
Batch size	64 matched log–text pairs
Number of negatives in Eq. 3.10	In-batch negatives with $M = B - 1 = 63$ for batch size $B = 64$
Contrastive temperature	0.07
Maximum training length	50 epochs
Early-stopping criterion	Stop after 10 consecutive epochs without validation-loss improvement; retain the checkpoint with the lowest validation loss
Input normalization	$L2$ normalization before similarity computation
Vision encoder batch size	32 frames per forward pass
Operating system	Ubuntu 24.04 LTS
Hardware	NVIDIA RTX 6000 Ada Generation GPU (48 GB VRAM) and 64 GB system RAM

Table 4.4: Reproducibility settings for the final reported training configuration.

4.6.3 Model inference

At inference time, the system performs text-to-multi-view video retrieval using a two-stage pipeline. Unlike the training process, no caption generation is required. Instead, the trained model is used to map text queries and driving logs into a shared embedding space, where semantic similarity can be measured efficiently. As illustrated in Figure 4.11, the inference workflow consists of the following steps:

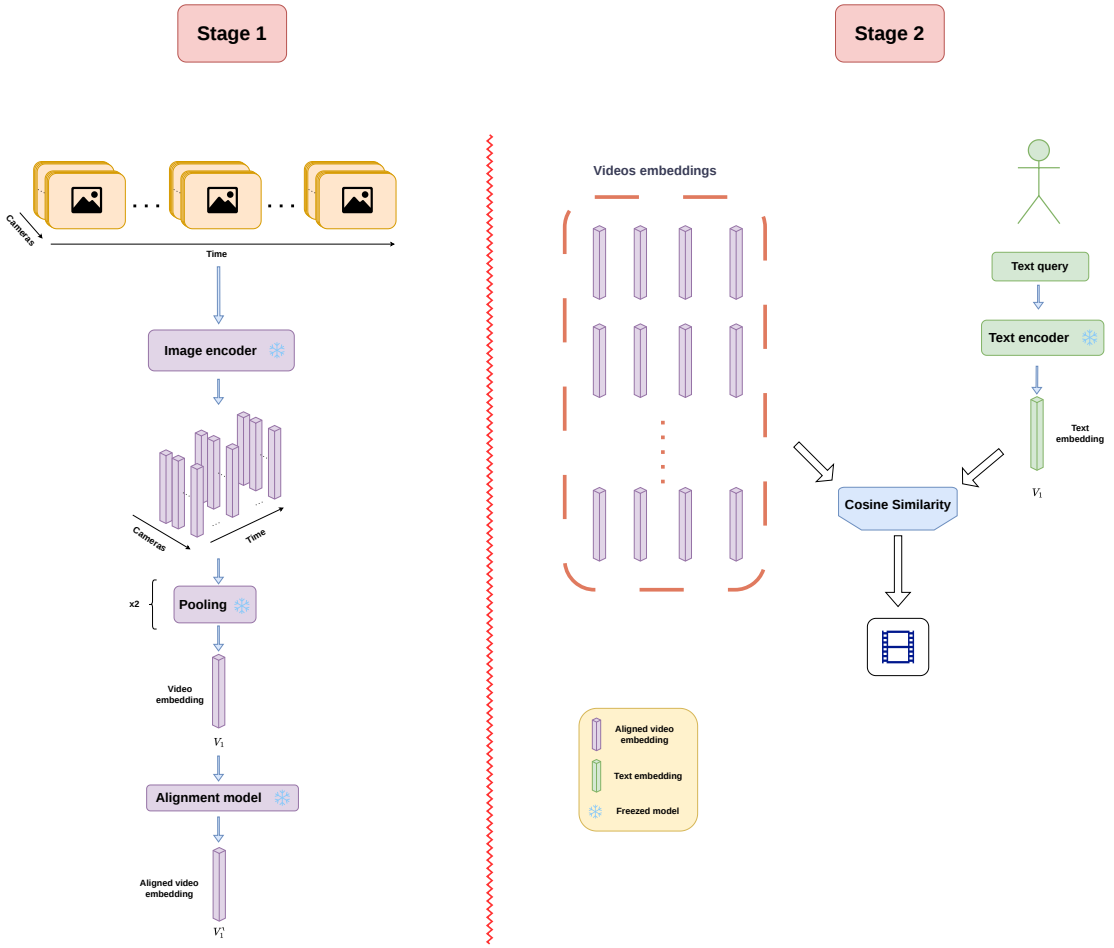


Figure 4.11: Inference workflow. **Stage 1:** Multi-camera driving logs are encoded, aggregated across camera views and time, and mapped into a database of precomputed video embeddings. **Stage 2:** A natural-language query is encoded into a text embedding and compared against the video embedding database using cosine similarity to retrieve the most relevant driving logs.

- **Stage 1 (Video Embedding Generation):** The multi-view camera streams from each driving log are processed sequentially. Frames from multiple cameras (e.g., Camera 1 to Camera 6) over time are passed through a visual backbone, namely the image encoder of jina-clip-v2. The resulting spatio-temporal feature tensors are aggregated via an attention-pooling model, as described in Section 4.4, and then mapped through a trained alignment model to generate a unified, compact video embedding vector (V_1) for each driving log. These vectors are precomputed and stored in a database of video embeddings.
- **Stage 2 (Text Querying and Retrieval):** A user-provided natural-language query is processed by the frozen jina-clip-v2 text encoder to extract a text embedding of identical dimensionality. Finally, cosine similarity is computed between the query embedding and the precomputed database of video embeddings. The driving logs are then ranked, and the scenarios with the highest similarity scores are retrieved.

4.7 Evaluation

This section outlines how the proposed retrieval system is evaluated. Two complementary evaluation approaches are used. First, Recall@K is used to assess how well the model has learned to align paired video and text samples within the shared embedding space. Second, an LLM-based semantic evaluation framework is used to assess the relevance of retrieved driving logs for free-form natural language queries.

Together, these evaluations provide insight into both the quality of the learned representations and the practical retrieval performance of the system. The evaluation is organized around the two research questions, allowing the effects of multi-view representations and temporal sampling to be analyzed independently while keeping the remainder of the retrieval pipeline fixed.

The first evaluation component is the rank-based metric Recall@K, which is used to assess the quality of the learned cross-modal embedding space.

Recall@K is a commonly used metric in retrieval tasks that measures how often the correct item appears among the top-K retrieved results. In this thesis, it is used to evaluate how well the model has learned to align video and text embeddings within the shared representation space.

To compute Recall@K, each text description is used to retrieve driving logs from the evaluation set. The retrieved logs are ranked according to their similarity to the text embedding, and the rank of the corresponding driving log is recorded. Recall@K is then calculated as the proportion of text–log pairs for which the correct driving log appears within the top-K retrieved results. Formally, Recall@K is defined as

$$\text{Recall@K} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\text{rank}_i \leq K)\,, \tag{4.5}$$

where N is the number of text-log pairs, rank_i denotes the rank of the correct driving log for text description i, and $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition is satisfied and 0 otherwise.

Throughout this thesis, Recall@5 and Recall@10 are reported. Higher values indicate that paired video-text samples are consistently placed close to one another in the learned embedding space, suggesting that the model has learned meaningful cross-modal representations.

It is important to note that Recall@K is evaluated using paired text-log samples from the dataset rather than the free-form natural language queries used in the retrieval experiments. As a result, Recall@K should be interpreted primarily as a measure of embedding-space quality and cross-modal alignment. The semantic relevance of retrieved results is instead evaluated using the LLM-based framework described in Section 4.7.1.

4.7.1 Semantic evaluation using LLM

Rank-based retrieval metrics evaluate whether the exact ground-truth item appears among the top results. However, in driving-scene retrieval, multiple different logs may be semantically relevant to the same query even if they are not the exact ground-truth instance. To account for this, we add a semantic evaluation to our rank-based metrics. This allows us to score how closely the actual meaning of the retrieved results (using captions from the multi-view videos) matches the query.

Figure 4.12 summarizes the semantic evaluation pipeline, where Qwen 3.6 acts as an LLM judge to compare query and retrieved descriptions across six predefined categories and produce an overall similarity score.

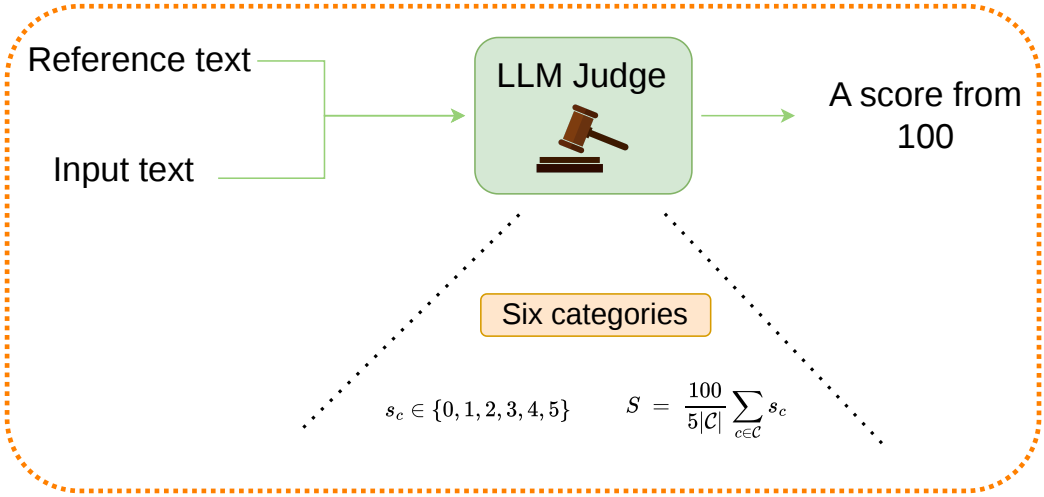


Figure 4.12: Overview of the LLM-based semantic evaluation framework. Qwen 3.6 is used as a judge to assess the semantic similarity between a query description and a retrieved driving-log description across six predefined evaluation categories.

We define semantic similarity using an LLM-based judge, Qwen 3.6 [27], that compares two text descriptions: the query description and a retrieved candidate description. The comparison follows a structured rubric with a fixed set of six semantic categories. For each category $c \in \mathcal{C}$, the judge produces a discrete similarity score

$$s_c \in \{0, 1, 2, 3, 4, 5\},$$

where 0 indicates no similarity and 5 indicates near-identical similarity in that category. The category scores are summed and normalized to produce a single score on a 0–100 scale:

$$S = \frac{100}{5|\mathcal{C}|} \sum_{c \in \mathcal{C}} s_c.$$

This provides a graded measure of semantic match between descriptions, and can be used to analyze whether retrieved results are meaningfully similar beyond exact instance matching.

The structured rubric evaluates similarity based on explicit information present in the descriptions across the following categories, $\mathcal{C}$:

- **Traffic composition and traffic density:** Evaluates the overall traffic volume (e.g., sparse, moderate, heavy) and the presence of specific vehicle types (e.g., trucks, buses, motorcycles).
- **Road environment and surrounding context:** Assesses the general setting, such as whether it is an urban or rural environment, the road type, surrounding structures, vegetation, and open areas.
- **Road infrastructure and layout:** Compares physical road features, including lane configurations, markings, traffic signs, traffic lights, intersections, exits, and construction-related changes.
- **Weather, visibility, and time-related conditions:** Focuses on meteorological conditions, road surface states, lighting, shadows, and time-of-day indicators.
- **Dynamic elements and non-vehicle road users:** Accounts for the presence, behavior, and interaction of vulnerable road users such as pedestrians, cyclists, or animals.
- **Special features and notable scenario-specific elements:** Captures distinct occurrences like construction zones, emergency vehicles, temporary obstacles, or other unique anomalies.

The semantic evaluation relies on a single LLM and prompt configuration and should therefore be interpreted as a relative proxy metric rather than an absolute ground truth. An additional validity threat is possible evaluation circularity: the retrieved log descriptions are generated and post-processed using Qwen-family models, while the semantic judge is Qwen 3.6 from the same model family. As a result, the judge may be partially sensitive to sibling-model phrasing and structural conventions rather than only to semantic content. To partially mitigate this risk, we performed a small manual spot-check on a subset of queries and compared the judge outputs with human ratings by us. The judgments were found to be broadly consistent, which supports using the metric as a comparative signal, although this limited validation does not remove the possibility of same-family bias. In addition, the LLM-judge evaluation was repeated in multiple runs to assess score stability. Across these repeated judge passes, the absolute mean scores varied only slightly and the relative ordering between the compared retrieval settings remained unchanged. The full prompt is provided in Section B.3 of Appendix A.

4.7.2 Evaluation for RQ1

Research Question 1 (RQ1) asks: How large is the retrieval performance gap between single-view and multi-view visual representations for log-level driving scene retrieval under otherwise matched conditions?

To isolate the effect of the video representation, we compare two models trained with the same text representation but different visual inputs. The first model uses only the front-view camera as visual input and is trained with shared multi-view text summaries. The second model uses multi-view video input and is trained with the same shared multi-view text summaries. With this setup, the difference in retrieval performance can be attributed to the use of single-view versus multi-view video information.

Across both settings, the same backbone encoders from jina-clip-v2, aggregation strategy, alignment layers, training procedure, and train/test split are used. Retrieval is performed by ranking database logs based on cosine similarity between L2-normalized log-level embeddings. Retrieval performance is evaluated using Recall@K with $K \in \{5, 10\}$. Recall@K is used to assess whether the learned embedding space exhibits a well-structured geometry in which the ground-truth log is placed among the nearest neighbors of its corresponding query. In this sense, Recall@K serves as a diagnostic metric for the geometric separability and consistency of the learned embedding space.

Because instance-level metrics alone do not capture whether retrieved results are semantically useful from a retrieval perspective, we additionally evaluate semantic relevance using an LLM-based similarity metric as mentioned in 4.7.1. This metric assesses how well the top-ranked retrieved logs semantically match the query description, independent of whether they correspond to the exact ground-truth instance. Semantic evaluation is performed on the top-5 retrieved results per query, and scores are aggregated across the test set. Details of the semantic evaluation rubric and prompting strategy are provided in Section B.3 of Appendix A.

4.7.3 Evaluation for RQ2

Research Question 2 (RQ2) asks: What minimum temporal sampling density per camera stream is required to achieve retrieval performance close to a denser reference? To answer this, we define a set of sampling configurations (number of sampled frames per log per camera) and evaluate retrieval quality across this sweep while keeping the remainder of the pipeline unchanged. This allows us to isolate the effect of temporal sampling on retrieval performance.

For each sampling configuration, we generate log-level video embeddings using the same frozen backbone encoders and the same aggregation and alignment setup. We then evaluate retrieval on the held-out test set using a fixed gallery of log embeddings. As a simple proxy for computational cost, we report the number of processed frames per log (per camera), since backbone inference scales approximately linearly with the number of frames processed.

We report rank-based retrieval performance using Recall@K for text-to-multi-view video retrieval, computed under cosine similarity after L2-normalization of embeddings. Recall@K is evaluated at $K \in \{5, 10\}$. The Recall@K results are compared across sampling configurations to quantify the cost–performance trade-off. To complement instance-level rank metrics, we also perform a semantic evaluation using an

LLM-based evaluator. For each query, the evaluator compares the query description with the descriptions of the top-5 retrieved logs and outputs a similarity score according to a structured rubric. The structured comparison prompt is provided in Section B.3 of Appendix A. We compute the mean score across the five retrieved results for each query, and then aggregate these per-query scores across the test set (e.g., by taking the mean) to compare semantic retrieval quality across sampling configurations.

Finally, we analyze robustness by visualizing the distribution of per-query semantic scores using violin plots for each sampling configuration. This provides a distribution-level comparison across sampling choices and highlights changes in variance and failure cases as sampling is reduced.

5

Results and discussion

5.1 Quantitative results

This section presents the experimental results of the proposed retrieval pipeline. First, quantitative retrieval performance is analyzed across a test dataset of 350 logs using standard retrieval metrics and semantic similarity evaluation. Thereafter, qualitative examples are presented to illustrate the behavior of the retrieval system under different query scenarios.

5.1.1 Evaluation for RQ1

Research Question 1 (RQ1) asks: How large is the retrieval performance gap between single-view and multi-view visual representations for log-level driving scene retrieval under otherwise matched conditions?

We first report rank-based retrieval performance using Recall@K for text-to-multi-view video retrieval. Recall@K measures whether the ground-truth log is retrieved among the top-K nearest neighbors in the learned embedding space under cosine similarity. While Recall@K is a useful and widely adopted metric for evaluating embedding quality, it is important to note that it does not fully capture practical retrieval performance from an end-user perspective. In realistic retrieval scenarios, users are unlikely to issue prompts that exactly match the pseudo ground-truth descriptions used during evaluation; instead, retrieval quality depends on whether semantically similar logs are retrieved even when queries are phrased differently. In this work, Recall@K is therefore used primarily to verify that a well-structured embedding space has been learned, rather than as a complete measure of retrieval usefulness.

We evaluate using Recall@K for $K \in \{5, 10\}$. Recall@K is appropriate for this task because it reflects how well the learned joint embedding space places relevant driving logs among the top-ranked retrieved results. Higher Recall@K therefore indicates a better geometric organization of the embedding space for retrieval.

For all Recall@K evaluations, each query was ranked against a test gallery of $M = 350$ logs. A random ranking would yield expected Recall@5 = 1.4% and Recall@10 = 2.9%.

To assess result stability, the main single-view and multi-view comparison was repeated in multiple runs. Across these repeated experiments, both Recall@K and the LLM-based semantic scores followed the same ordering, with the multi-view setting consistently outperforming the single-view setting and only small variation in the

absolute score values between runs.

Figure 5.1 highlights the performance gap between the single-view and multi-view settings. Recall@5 increases from 71% to 85%, and Recall@10 from 81% to 93%. Despite this gap, the single-view setup still demonstrates the ability to learn a meaningful embedding space, where relevant video–text pairs are placed sufficiently close to each other.

To complement Recall@K, we also use an LLM-based semantic similarity judge. This metric is not intended to measure exact-instance retrieval accuracy; instead, it evaluates how semantically similar the top retrieved results are to the query. In this way, it provides additional evidence about the semantic relevance of the retrieved driving logs beyond rank-based correctness alone.

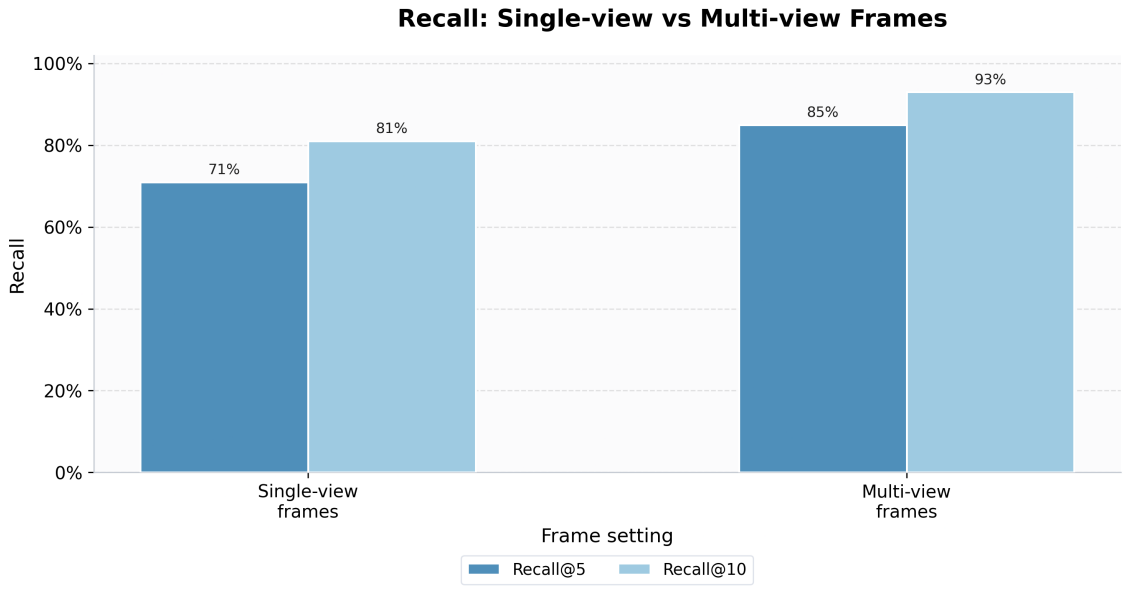


Figure 5.1: Recall@K comparison between the single-view and multi-view retrieval settings. The multi-view model achieves higher recall at all reported cutoffs, showing that access to multiple camera views improves the structure of the learned retrieval embedding space.

Figure 5.2 presents the distribution of LLM-judge scores for all query-log pairs and compares it with the mean top-5 retrieved scores for the single-view and multi-view settings. The overall distribution of all pairs has a mean score of 54.5, whereas the mean top-5 retrieved scores are substantially higher for both retrieval models: 76.9 for the single-view model and 79.1 for the multi-view model. This shows that both systems retrieve semantically relevant results well above the dataset-wide baseline, while the multi-view model provides a further improvement over the single-view model.

The difference between the two retrieval settings is smaller for the LLM-judge metric than for Recall@K, while remaining consistent in direction across repeated runs. This highlights that the gap between single-view and multi-view setups is more

pronounced when considering how the embedding space is structured, as reflected by Recall@K, and less pronounced when evaluating final retrieval behavior. Taken together, this suggests that although multi-view information contributes to forming a more separable embedding space, its relative advantage is modest but stable at the semantic retrieval level rather than being purely an artifact of judge noise.

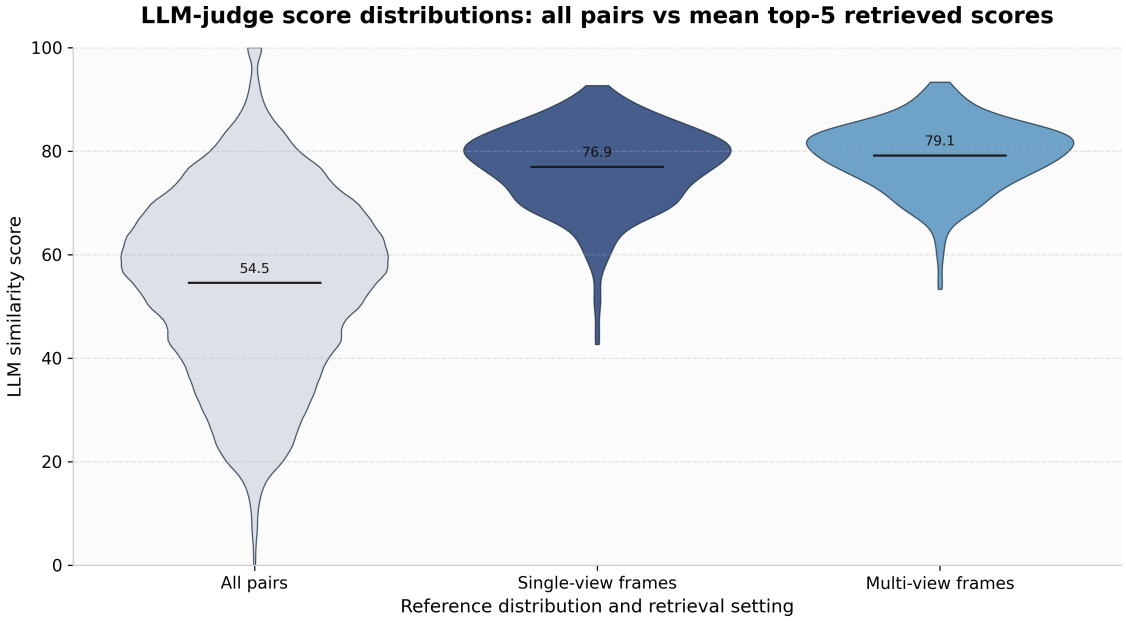


Figure 5.2: Distribution of LLM-judge similarity scores for all query-log pairs and for the mean top-5 retrieved results in the single-view and multi-view settings. Both retrieval settings substantially outperform the overall pairwise baseline, while the multi-view setting achieves the highest mean semantic similarity.

5.1.2 Evaluation for RQ2

Research Question 2 (RQ2) asks: What minimum temporal sampling density per camera stream is required to achieve retrieval performance close to a denser reference? We evaluate this by sweeping the number of sampled frames per log and measuring retrieval quality on the held-out test set under otherwise matched conditions.

We first report rank-based retrieval performance using Recall@K for text-to-log retrieval. Recall@K measures whether the ground-truth log is retrieved among the top-K nearest neighbors in the learned embedding space under cosine similarity. While Recall@K is a useful and widely adopted metric for evaluating embedding quality, it is important to note that it does not fully capture practical retrieval performance from an end-user perspective. In realistic retrieval scenarios, users are unlikely to issue prompts that exactly match the pseudo ground-truth descriptions used during evaluation; instead, retrieval quality depends on whether semantically similar logs are retrieved even when queries are phrased differently. In this work, Recall@K is therefore used primarily to verify that a well-structured embedding

space has been learned, rather than as a complete measure of retrieval usefulness.

We compute Recall@K for $K \in \{5, 10\}$ and compare how performance changes across sampling densities. Figure 5.3 shows Recall@K as a function of sampling density. As shown in the figure, performance remains largely stable as the sampling rate is reduced, suggesting that consecutive frames contain a substantial degree of redundant information for the evaluated task. This indicates that a dense temporal sampling is not strictly necessary to preserve embedding quality, at least within the higher sampling regimes.

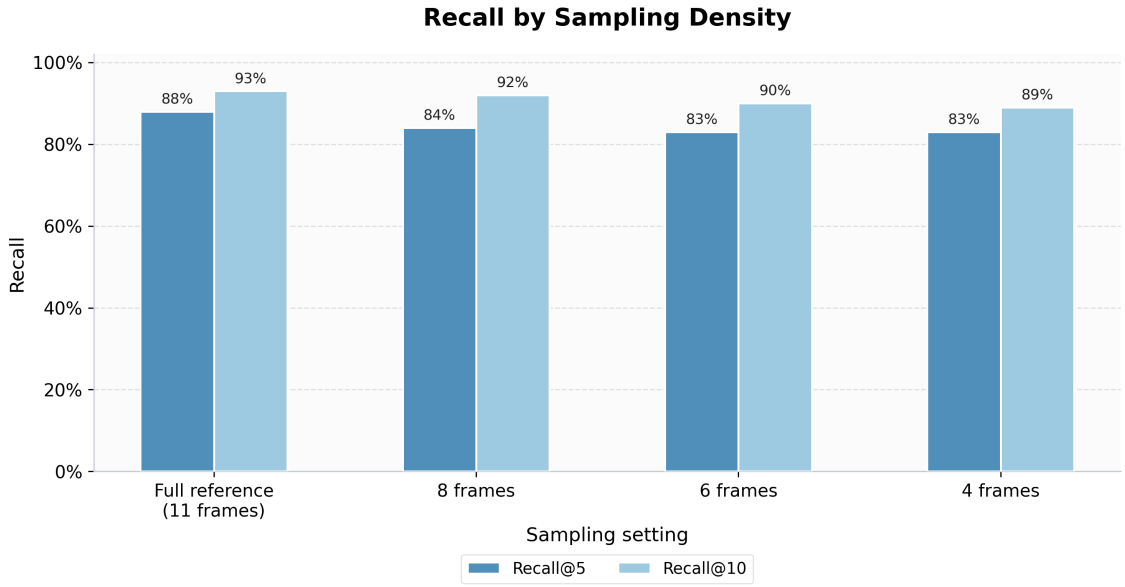


Figure 5.3: Recall as a function of temporal sampling rate per camera stream. Performance is reported relative to a dense reference configuration, illustrating how recall changes as the sampling rate is progressively reduced.

To analyze the efficiency–accuracy trade-off, we evaluate recall as a function of computational cost while progressively lowering the temporal sampling rate per camera. Computational cost is measured relative to the full-sampling reference and scales linearly with the total number of frames processed. Figure 5.4 plots Recall@K against relative compute cost, illustrating how the quality of the learned embedding space, as measured by rank-based recall, changes as sampling is reduced. This visualization provides a compact summary of how embedding consistency scales with computational effort under temporal down sampling. The results show that high recall can be preserved across a wide range of lower compute budgets. In particular, configurations with significantly reduced relative compute cost achieve recall levels close to the reference baseline, suggesting that much of the computation associated with dense sampling does not translate into proportional improvements in embedding-space alignment.

Finally, we analyze robustness across queries using violin plots that visualize the distribution of semantic judge scores for each sampling configuration. The plots are based on the per-query mean top-5 judge score and allow comparison of both central

tendency and variance across sampling densities (Figure 5.5). This distributional view assesses whether reduced sampling leads to uniform semantic degradation or increases the number of severe failure cases.

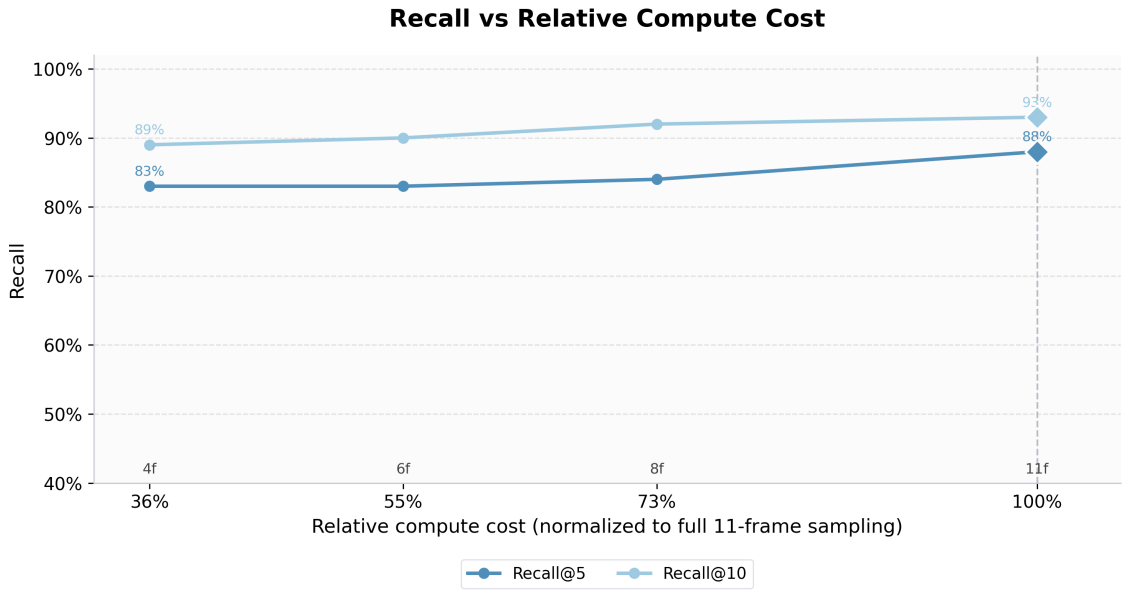


Figure 5.4: Recall plotted against relative compute cost normalized to the dense reference configuration. The figure highlights the cost–performance tradeoff of temporal downsampling and shows that high recall can be maintained across a wide range of reduced compute budgets

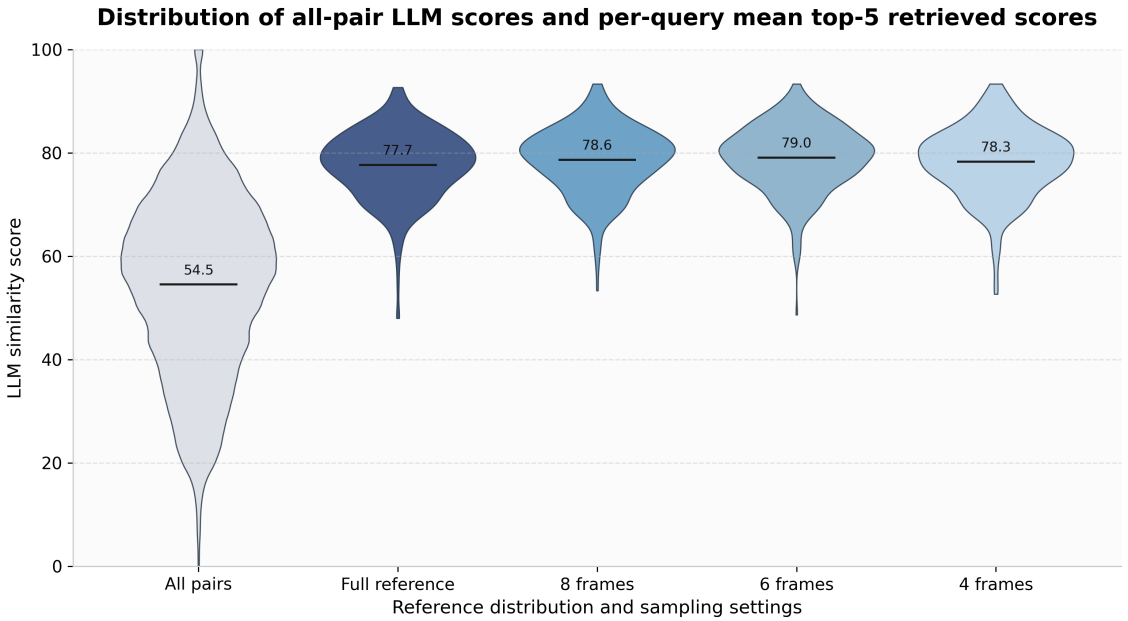


Figure 5.5: Violin plots showing the distribution of per-query mean top-5 semantic judge scores for different temporal sampling configurations. The plots illustrate similarity in both central tendency and variability across sampling rates, indicating that retrieval performance remains consistent even under reduced temporal sampling.

Overall, the score distributions remain highly similar across sampling rates, with comparable medians and spreads even at lower sampling densities. This indicates that semantic retrieval quality is largely insensitive to temporal down sampling within the evaluated range and that reduced sampling does not introduce increased variability or isolated failure cases. Even at more aggressive reductions, the distributions do not show a pronounced downward shift, suggesting that practical retrieval behavior remains robust due to temporal redundancy. An additional observation is that the full reference configuration achieves the lowest average similarity score, although differences between configurations are small and the distributions largely overlap. These variations should still be interpreted with caution, since the similarity metric is derived using a large language model. However, when the judge evaluation was repeated in multiple runs, the overall pattern across sampling configurations remained the same, indicating that the near-overlapping semantic scores are not driven by a single anomalous judge pass. Consequently, semantic similarity scores should be interpreted in relative terms and in conjunction with Recall@K and compute-cost analyses when evaluating RQ2.

Overall, these results indicate that temporal sampling can be substantially reduced per camera stream without degrading retrieval performance.

5.2 Qualitative results

This section presents qualitative retrieval examples from the dataset to illustrate the behavior of the proposed retrieval system. The examples highlight both successful retrieval cases and representative limitations, providing additional insights.

Figures 5.6, 5.7, 5.8 present examples of successful retrieval cases produced by the proposed retrieval system. For each query, the top three retrieved driving logs are semantically consistent with the queried driving scenario, demonstrating that the learned embedding space is able to capture high-level scene semantics across different driving environments and viewpoints.

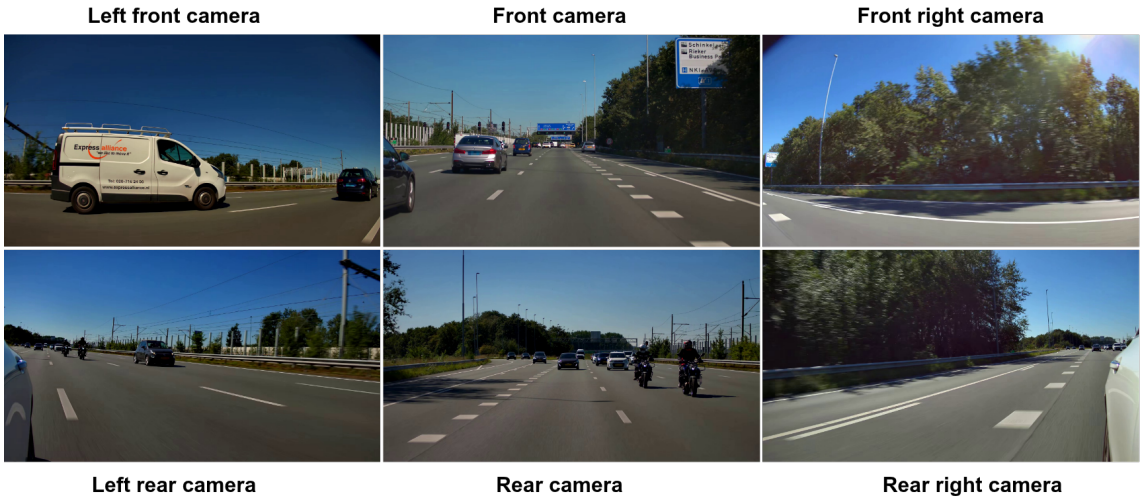


Figure 5.6: First retrieved driving log for the query: "Multi-lane highway with van and surrounding greenery".

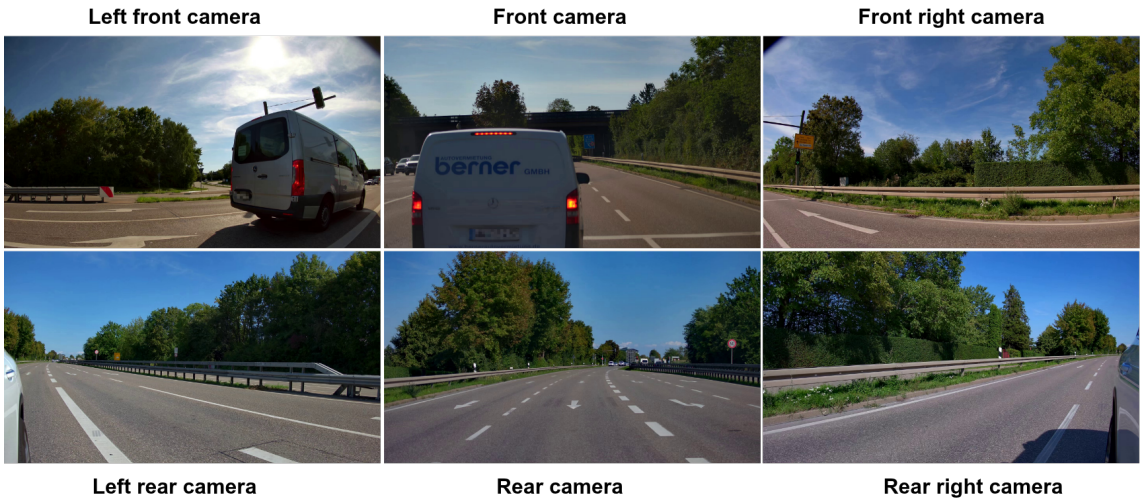


Figure 5.7: Second retrieved driving log for the query: "Multi-lane highway with van and surrounding greenery".

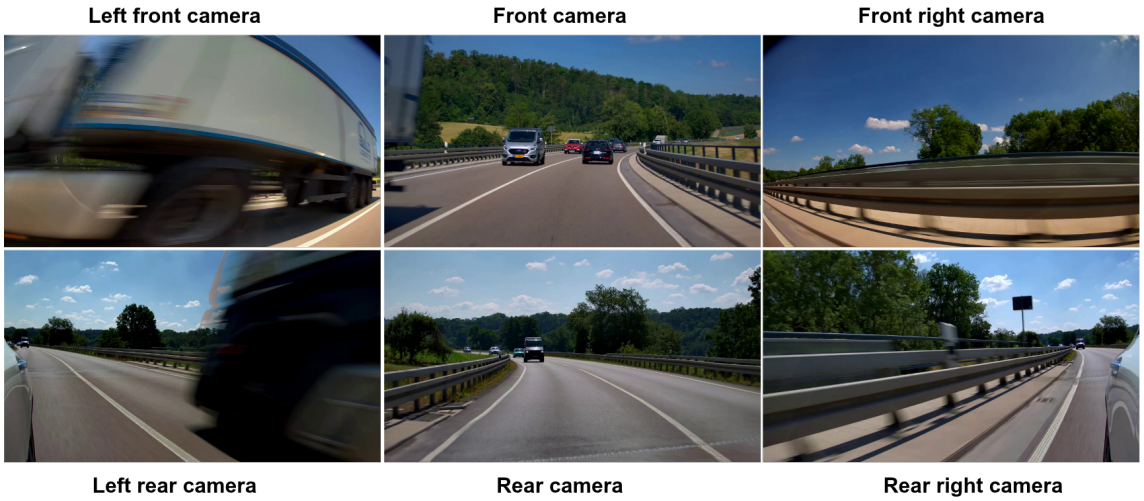


Figure 5.8: Third retrieved driving log for the query: "Multi-lane highway with van and surrounding greenery".

Figures 5.9, 5.10, and 5.11 illustrate a representative limitation of the retrieval system. Although the retrieved driving logs contain semantically related traffic situations and correctly capture the presence of a traffic light scenario, the exact query condition of three vehicles stopping at the traffic light is not consistently satisfied across the top-ranked results.

These examples highlight the difficulty of retrieving fine-grained scene details from compressed log-level embeddings, particularly when multiple visually and semantically similar traffic situations coexist in the dataset. In particular, the aggregation process does not explicitly reason about object counts across synchronized camera views, making precise counting-based queries challenging. The results suggest that the retrieval system captures the overall driving context effectively, while more fine-grained object-level reasoning and counting remain difficult.

Figure 5.9: Prompt: Three cars are stopped at a traffic light. 1st retrieved log.

Figure 5.10: Prompt: Three cars are stopped at a traffic light. 2nd retrieved log.

Figure 5.11: Prompt: Three cars are stopped at a traffic light. 3rd retrieved log.

5.3 Discussion

Taken together, the results support the central premise of the thesis that multi-view aggregation improves log-level driving-scene retrieval by reducing ambiguity in the visual representation. The strongest evidence for this appears in RQ1, where the multi-view model consistently improves Recall@K over the single-view baseline. This indicates that access to synchronized surrounding cameras helps organize the shared embedding space more effectively, which is consistent with the motivation presented in the background and theory chapters: complementary views provide scene evidence that a single front-facing camera may miss.

At the same time, the smaller gap in LLM-based semantic similarity than in Recall@K adds an important nuance to that interpretation. The multi-view setting does not dramatically change whether the retrieved logs are broadly relevant; rather,

it more clearly separates the exact ground-truth match from other semantically similar scenes. In practical terms, this suggests that multi-view fusion mainly strengthens discrimination among related traffic situations instead of fundamentally changing the type of scenes the system can retrieve.

The RQ2 results further show that this retrieval pipeline is relatively insensitive to dense temporal sampling. Because the method aggregates sampled frames into a single log-level representation, substantial temporal redundancy appears to remain in the driving videos for the evaluated task. This is an important result for scalability: it suggests that a large fraction of the computational cost of dense sampling can be removed without materially degrading either rank-based retrieval quality or semantic relevance.

Several validity limitations should nevertheless shape how these findings are interpreted. First, the supervision signal is derived from pseudo-labels generated by pretrained vision-language models, so the learned retrieval space is partly bounded by the descriptive quality, coverage, and biases of those captions. Second, the semantic evaluation relies on an LLM-based judge, which is useful for relative comparisons but should not be treated as absolute ground truth. Repeated runs support the reliability of the observed ranking trends, but the qualitative failure cases also show that fine-grained details such as object counts and precise scene composition remain difficult to recover from compressed log-level embeddings. These limitations make the current results strongest as evidence for scalable scene-level retrieval rather than precise object-level reasoning.

6

Conclusion and future work

This thesis investigated semantic retrieval of large-scale autonomous driving logs using embedding-based vision-language retrieval. In particular, the work focused on how synchronized multi-view driving data can be aggregated into compact log-level representations suitable for scalable retrieval.

To enable supervised training without manual annotation, pseudo ground-truth captions were automatically generated using pretrained vision-language models, allowing aligned text-video pairs to be constructed directly from the driving data. A hierarchical aggregation framework was then developed to combine embeddings across both camera views and time.

The experimental results show that extending retrieval from a single front-facing camera to six synchronized camera views improves the structure of the learned embedding space, with Recall@5 increasing from 71% to 85% and Recall@10 from 81% to 93%. At the same time, the LLM-based semantic similarity score improves only marginally, suggesting that multi-view fusion primarily strengthens discrimination between semantically similar driving logs rather than significantly expanding semantic coverage. Repeated experimental runs showed the same overall ordering for both Recall@K and the LLM-based semantic metric, which supports interpreting the multi-view advantage as stable even though the semantic-score gap is small. These improvements are achieved using frozen pretrained encoders together with lightweight aggregation and alignment modules, without requiring computationally expensive 3D scene representations.

The experiments also highlight several limitations. While the retrieval system captures the overall semantic driving context effectively, fine-grained reasoning tasks such as object counting and precise scene composition remain challenging when using compressed log-level embeddings. In addition, since supervision is derived from automatically generated pseudo captions, the retrieval pipeline may partially inherit semantic biases and phrasing structures from the pretrained captioning models. A further validity threat is that the LLM-based semantic judge belongs to the same Qwen family as the models used to generate and refine the retrieved descriptions, which introduces a risk of partiality toward familiar phrasing. A small manual spot-check by us indicated that the relative judge scores were generally aligned with human assessment, and repeated judge runs produced consistent relative rankings, but this validation remains limited and the semantic metric should still be interpreted as a proxy rather than absolute evidence.

Overall, the results indicate that scalable multi-view embedding aggregation is a promising direction for semantic retrieval of autonomous driving data. The thesis

demonstrates that meaningful retrieval performance can be achieved directly from synchronized surround-view recordings while maintaining computational efficiency and scalability.

6.1 Societal, ethical and ecological considerations

This work has a clear societal relevance because more effective retrieval of driving logs can support the validation and debugging of ADAS perception systems, which may help identify failure modes earlier and thereby contribute to safer road traffic and more reliable transport systems [29], [30], [31].

The thesis also raises ethical questions related to privacy, fairness, and dual use. The dataset used in this work was provided in anonymized form by NVIDIA, and all caption generation and model training were performed on a local GPU workstation rather than through external cloud services, which reduced unnecessary data exposure during experimentation. Even so, real-world traffic footage can still encode sensitive contextual information about pedestrians, vehicles, and locations, and a retrieval system of this kind could be misused for surveillance or targeted monitoring if access control and governance are weak. In addition, any imbalance in the underlying driving data may lead to uneven retrieval quality across environments, road users, or geographic contexts, which should be considered when interpreting the results [32], [33], [34].

From an ecological perspective, the proposed method is substantially lighter than end-to-end training of large video models because it relies on pretrained encoders, lightweight aggregation modules, and reduced temporal sampling. However, large-scale pseudo-caption generation and embedding extraction still require non-negligible GPU computation, so there remains an energy cost when scaling the pipeline to very large driving archives [35]. Future deployments should therefore weigh retrieval benefits against compute usage and prefer efficient inference settings where possible.

6.2 Future work

While this thesis demonstrates the effectiveness of multi-view driving scene retrieval using frozen pretrained encoders, several promising avenues for future research remain:

- **Improved pseudo-label generation:** Because the retrieval system is supervised by VLM-generated descriptions, its ceiling performance is heavily influenced by the quality of these pseudo-labels. Exploring native video-language models for temporally coherent captioning, or integrating a small subset of high-quality human-annotated logs, could enhance the semantic richness of the training data.
- **Advanced spatial-temporal fusion:** Although learnable attention pooling proved effective and efficient for view-level and temporal aggregation, more

sophisticated architectures (such as cross-view transformers or spatiotemporal attention mechanisms) might better capture the complex dynamic interactions between moving objects across different camera views.

- **Cross-dataset generalization:** Evaluating the proposed multi-view retrieval framework on other large-scale autonomous driving datasets, such as nuScenes or Waymo Open Dataset, would provide valuable insights into the model's robustness to different sensor configurations, camera extrinsics, and diverse geographic distributions.

References

[1] A. Singh, "Surround-view vision-based 3d detection for autonomous driving: A survey," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 3243–3252.

[2] A. Radford et al., *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.00020.

[3] T. Tang et al., *Bev-tsr: Text-scene retrieval in bev space for autonomous driving*, 2024. arXiv: 2401.01065 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.01065.

[4] M. Liu et al., *A survey on autonomous driving datasets: Statistics, annotation quality, and a future outlook*, 2024. arXiv: 2401.01454 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.01454.

[5] B. Zhu et al., "Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[6] Z. Wang, Y. Wu, K. Narasimhan, and O. Russakovsky, *Multi-query video retrieval*, 2022. arXiv: 2201.03639 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2201.03639.

[7] Z. Dai, K. Cheng, F. Shao, Z. Dong, and S.-C. Zhu, "Text–video retrieval re-ranking via multi-grained cross attention and frozen image encoders," *Pattern Recognition*, vol. 159, p. 111 099, 2025.

[8] Y. Duan, X. Huang, and D. Chen, *Enhancing subsequent video retrieval via vision-language models (vlms)*, 2025. arXiv: 2503.17415 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2503.17415.

[9] Y. Wang et al., "Internvideo2: Scaling video foundation models for multimodal video understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[10] M. A. Arefeen, B. Debnath, and S. Chakradhar, "Trafficlens: Multi-camera traffic video analysis using llms," in *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Sep. 2024, pp. 3974–3981. DOI: 10.1109/itsc58415.2024.10920144. [Online]. Available: http://dx.doi.org/10.1109/ITSC58415.2024.10920144.

[11] M. Ki et al., "Carim: Caption-based autonomous driving scene retrieval via inclusive text matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[12] M. Oquab et al., "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[13] S. Wozniak et al., "S3pt: Scene semantics and structure guided clustering to boost self-supervised pre-training for autonomous driving," in *Proceedings*

of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2025.

[14] G. Zhang, J. Ren, J. Gu, and V. Tresp, *Multi-event video-text retrieval*, 2026. arXiv: 2308.11551 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2308.11551.

[15] S. Bai et al., *Qwen2.5-vl technical report*, 2025. arXiv: 2502.13923 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2502.13923.

[16] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, *Object hallucination in image captioning*, 2019. arXiv: 1809.02156 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1809.02156.

[17] Q. Zhang et al., *Rethinking pseudo-label guided learning for weakly supervised temporal action localization from the perspective of noise correction*, 2025. DOI: 10.48550/arXiv.2501.11124. arXiv: 2501.11124 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2501.11124.

[18] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, *Towards general text embeddings with multi-stage contrastive learning*, 2023. arXiv: 2308.03281 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2308.03281.

[19] H. Alkaabi, A. K. Jasim, and A. Darroudi, "From static to contextual: A survey of embedding advances in nlp," *PERFECT: Journal of Smart Algorithms*, vol. 2, no. 2, pp. 57–66, Jul. 2025. DOI: 10.62671/perfect.v2i2.77.

[20] A. Vaswani et al., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, CA, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[21] J. Selva, A. S. Johansen, S. Escalera, K. Nasrollahi, T. B. Moeslund, and A. Clapés, "Video transformers: A survey," *arXiv preprint arXiv:2201.05991*, 2022. [Online]. Available: https://arxiv.org/pdf/2201.05991.

[22] A. Radford, J. W. Kim, C. Hallacy, et al., "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[23] L. Weng, "Contrastive representation learning: A survey," *arXiv preprint arXiv:2005.10242*, 2020.

[24] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[26] NVIDIA, *Physicalai-autonomous-vehicles dataset*, https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles, Accessed: 2026-03-16, 2025.

[27] Qwen Team, *Qwen3.6-35B-A3B: Agentic coding power, now open to all*, Apr. 2026. [Online]. Available: https://qwen.ai/blog?id=qwen3.6-35b-a3b.

[28] Z. Lin et al., *A structured self-attentive sentence embedding*, 2017. arXiv: 1703.03130 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1703.03130.

[29] United Nations Department of Economic and Social Affairs. "Goal 3: Ensure healthy lives and promote well-being for all at all ages," Accessed: Jun. 27, 2026. [Online]. Available: https://sdgs.un.org/goals/goal3.

[30] World Health Organization. "Global plan for the decade of action for road safety 2021–2030," Accessed: Jun. 27, 2026. [Online]. Available: https://cdn.

who.int/media/docs/default-source/documents/health-topics/road-traffic-injuries/global-plan-for-road-safety.pdf.

[31] Z. Wadud, "Quantifying the impact of deployments of autonomous vehicles on road safety," *npj Urban Sustainability*, vol. 3, p. 8, 2023.

[32] J. Zhang. "A literature review of data ethics in autonomous vehicles," Accessed: Jun. 27, 2026. [Online]. Available: https://nhsjs.com/2024/a-literature-review-of-data-ethics-in-autonomous-vehicles/.

[33] Numalis. "The ethics of ai in transportation: Balancing safety, privacy, and fairness," Accessed: Jun. 27, 2026. [Online]. Available: https://numalis.com/ethics-of-ai-in-transportation-safety-privacy-fairness/.

[34] Robotics Industries Association. "Ethics in autonomous industrial ai: Tackling bias and data governance," Accessed: Jun. 27, 2026. [Online]. Available: https://www.automate.org/ai/industry-insights/ethics-in-autonomous-industrial-ai.

[35] L. Smith, M. Garcia, and R. Patel, "Sustainability of autonomous cars: Environmental, social and economic dimensions," *Journal of Cleaner Transportation*, vol. 12, pp. 45–60, 2025.

A

Prompts for Pseudo-Label Generation and LLM-Based Evaluation

This appendix documents the prompts used in the different stages of pseudo ground-truth caption generation, aggregation, post-processing, and LLM-based evaluation. To preserve reproducibility, the prompts are reproduced verbatim and presented in the order in which they are applied in the pipeline.

A.1 Frame-Level Captioning Prompts

This stage uses a fixed system prompt together with a task-specific analysis prompt for each sampled frame.

System Prompt

```
You are an automotive scene analyst creating detailed,
   retrieval-quality captions for driving scenarios.

GUIDELINES:
- Provide rich, specific descriptions grounded strictly in
   visible evidence
- Stay factual - do not hallucinate, speculate, or infer
   hidden details
- Focus on elements relevant for scene understanding and
   retrieval

WHAT TO DESCRIBE (when visible):
1. Traffic composition: specific vehicle types, colors,
   positions (e.g., "a red sedan in the left lane, white
   truck ahead")
2. Environment details: road type, surrounding structures,
   vegetation, urban/rural characteristics
3. Weather and visibility: precise conditions (e.g., "
   overcast with wet road surface", "clear sunny conditions")
4. Road infrastructure: lane configuration, markings, signs,
   signals, intersections
5. Dynamic elements: pedestrians, cyclists, or other road
   users if present
6. Time indicators: lighting conditions, shadow directions
```

```
7. Special features: construction zones, emergency vehicles,
   notable landmarks (use generic terms)

Write 3-5 detailed, factual sentences that paint a complete
   picture of the scene.
```

Analysis Prompt

```
CRITICAL CONSTRAINTS:
1. Only describe what you see - never mention what is absent
   or missing
2. Ignore all business/store names on signs - use generic
   types (e.g., "retail store", "restaurant")
3. Never use: "no", "not", "without", "lack", "none", "absent
   "

Provide a detailed, comprehensive description of this driving
    scene.

Be specific and descriptive:
- Vehicle details: types, colors, approximate positions,
   quantities
- Environment: road type, structures, surroundings,
   vegetation
- Weather and lighting: specific conditions, visibility, time
    of day indicators
- Road features: lanes, markings, signs, signals, surface
   conditions
- If pedestrians, cyclists, or other road users are visible,
   describe them
- Any construction, special vehicles, or notable elements if
   present

Use concrete, descriptive language. Cover multiple aspects
   for a complete scene understanding.
```

A.2 Aggregation Prompts

The following prompts are used to aggregate captions first across synchronized camera views and then across timesteps.

Camera-Level Aggregation Prompt

```
You are an expert driving-scene summarization assistant.
Generate one concise, coherent paragraph from provided camera
   -view captions for a single timestep.
Stay strictly grounded in the provided captions.
```

```
CRITICAL RULE: Only describe what you see. Never mention what
    is absent or missing.

Now describe this driving scene. Only mention visible
   elements:
- Traffic level and vehicles
- Environment type
- Weather conditions
- Time of day
- Special vehicles
- Animals, pedestrians, cyclists
- Special vehicles
- Traffic lights and signs
```

Timestep-Level Aggregation Prompt

```
ADAS driving scenario summarization for retrieval systems.

*** CRITICAL CONSTRAINTS ***

NEVER mention absence or missing elements:
- FORBIDDEN: "no pedestrians", "no cyclists", "not visible",
   "without", "minimal", "lack of", "none"
- If something isn't in the summaries, omit it completely

Describe ONLY ADAS-relevant elements:
INCLUDE:
- Traffic level and vehicles
- Environment type
- Weather conditions
- Time of day
- Special vehicles
- Animals, pedestrians, cyclists
- Special vehicles
- Traffic lights and signs
- Road config, traffic patterns, weather/visibility,
   infrastructure, hazards, traffic lights, special vehicles,
    pedestrians/cyclists, animals
- EXCLUDE: House colors, architectural details, tree
   descriptions, aesthetic language ("quiet", "serene", "calm
   ")

Include ALL mentioned ADAS-critical elements from ANY
   timestep:
- If construction appears in any frame, include it
- If traffic light status is mentioned in any frame, include
   it
```

```
- Aggregate all infrastructure: signs, signals, crosswalks,
   barriers, roadworks
- DO NOT omit elements just because they appear infrequently
   - if mentioned in summaries, include it

Start directly with scenario - NO meta phrases:
- FORBIDDEN: "The video shows...", "This log captures...", "
   The footage depicts..."
- START WITH: "Residential two-lane roads..." NOT "The video
   shows residential roads..."


*** OUTPUT ***
Write 4-6 sentences. Technical language. ADAS-critical
   elements only. Start directly with scenario description.
IMPORTANT: Include ALL elements (signs, signals, crosswalks,
   barriers) mentioned in ANY timestep summary.
```

A.3 Post-Processing Prompt

The prompt below is applied to the aggregated log-level description as a final refinement step.

```
"You are a post-processing assistant for ADAS retrieval
   summaries. "
"Enhance the given global_summary to be more concise,
   technical, and focused on ADAS-relevant details. "
"It will be used for retrieving similar driving scenarios, so
    emphasize distinctive and technical aspects of the scene.
    "
"Stay grounded in the given text and do not invent facts. "
"CRITICAL RULE: Never mention absence, non-presence, or
   missing elements in any form. "
"If the input includes any absence statement, delete that
   statement entirely instead of rewriting it. "
"Forbidden wording includes: no, not, not visible, without,
   none, absent, lack, lacking, minimal, low, sparse, "
"no detected, no pedestrians, no cyclists, no vehicles, no
   signs of. "
"Start directly with scenario content and avoid meta phrases
   like 'The video shows'. "
"Keep only positive, directly observed ADAS-relevant facts (
   traffic, road layout, signs/signals, weather, visibility,
   vehicles, pedestrians/cyclists only when present). "
"Return only the rewritten summary text."
```

B

Additional experimental results

B.1 Extended retrieval results

Table B.1 reports the broader ablation over embedding backbones, alignment heads, and loss functions that informed the final model choice.

Table B.1: Text-to-multi-view video retrieval performance on the full dataset for different combinations of text encoders, video encoders, alignment architectures, and loss functions. Results are reported using Recall@K (R@1, R@5), Median Rank (MedR), and Mean Reciprocal Rank (MRR).

Text embedding model	**Video embedding model**	**Alignment model**	**Loss**	R@1	R@5	**MedR**	**MRR**
MiniLM-L6-v2 (1x384)	SigLIP2 base patch 16-224 (1x768)	1 linear layer	InfoNCE	0.23	0.48	6	0.36
MiniLM-L6-v2	SigLIP2 base patch 16-224	MLP GeLU	InfoNCE	0.22	0.51	5	0.36
MiniLM-L6-v2	SigLIP2 base patch 16-224	MLP Leaky ReLU	InfoNCE	0.20	0.48	6	0.33
MiniLM-L6-v2	SigLIP2 base patch 16-224	1 linear layer	L2	0.26	0.53	5	0.38
MiniLM-L6-v2	SigLIP2 base patch 16-224	MLP GeLU	L2	0.21	0.49	6	0.34
MiniLM-L6-v2	SigLIP2 base patch 16-224	MLP Leaky ReLU	L2	0.19	0.47	6	0.33
jina-clip-v2 (1x1024)	jina-clip-v2 (1x1024)	1 linear layer	InfoNCE	0.41	0.72	2	0.56
jina-clip-v2	jina-clip-v2	MLP GeLU	InfoNCE	0.35	0.70	2	0.51
jina-clip-v2	jina-clip-v2	MLP Leaky ReLU	InfoNCE	0.33	0.66	3	0.48
jina-clip-v2	jina-clip-v2	1 linear layer	L2	0.37	0.71	2	0.52
jina-clip-v2	jina-clip-v2	MLP GeLU	L2	0.34	0.69	2	0.50
jina-clip-v2	jina-clip-v2	MLP Leaky ReLU	L2	0.33	0.66	3	0.48
LongCLIP (1x512)	LongCLIP (1x512)	1 linear layer	InfoNCE	0.39	0.74	2	0.54
LongCLIP	LongCLIP	MLP 2-layer GeLU	InfoNCE	0.31	0.65	3	0.48
LongCLIP	LongCLIP	MLP 2-layer Leaky ReLU	InfoNCE	0.28	0.60	4	0.42
LongCLIP	LongCLIP	1 linear layer	L2	0.36	0.67	2	0.51
LongCLIP	LongCLIP	MLP 2-layer GeLU	L2	0.35	0.67	3	0.48
LongCLIP	LongCLIP	MLP 2-layer Leaky ReLU	L2	0.35	0.64	3	0.48

B.2 Grid search

Table B.2 lists the full pooling sweep used to compare view-level and temporal aggregation choices.

Table B.2: Grid-search results for text-to-multi-view video retrieval. attn = attention, t = temperature, h = hidden dimension.

Camera	Temporal	R@1	R@5	Med. rank
attn(t=1.0, h=256)	attn(t=0.5, h=256)	0.332	0.623	3.0
attn(t=1.0, h=256)	attn(t=0.5, h=128)	0.326	0.618	3.0
attn(t=1.0, h=256)	attn(t=1.0, h=128)	0.326	0.607	3.0
attn(t=0.5, h=128)	attn(t=0.5, h=256)	0.326	0.621	3.0
attn(t=0.5, h=256)	attn(t=0.5, h=128)	0.331	0.613	3.0
attn(t=1.0, h=256)	attn(t=1.0, h=256)	0.333	0.623	3.0
attn(t=0.5, h=256)	mean	0.330	0.628	3.0
mean	attn(t=1.0, h=128)	0.333	0.626	3.0

Continued on next page

Camera	Temporal	R@1	R@5	Med. rank
attn(t=1.0, h=128)	attn(t=0.5, h=128)	0.324	0.618	3.0
attn(t=1.0, h=128)	attn(t=1.0, h=128)	0.329	0.620	3.0
mean	attn(t=1.0, h=256)	0.330	0.622	3.0
attn(t=0.5, h=128)	attn(t=1.0, h=128)	0.325	0.612	3.0
attn(t=0.5, h=256)	attn(t=1.0, h=256)	0.322	0.615	3.0
mean	attn(t=0.5, h=256)	0.325	0.622	3.0
mean	mean	0.323	0.627	3.0
attn(t=0.5, h=128)	attn(t=0.5, h=128)	0.326	0.618	3.0
attn(t=0.5, h=128)	mean	0.326	0.615	3.0
mean	attn(t=0.5, h=128)	0.326	0.623	3.0
attn(t=1.0, h=128)	attn(t=0.5, h=256)	0.320	0.614	3.0
attn(t=0.5, h=128)	attn(t=1.0, h=256)	0.325	0.616	3.0
attn(t=1.0, h=128)	attn(t=1.0, h=256)	0.326	0.620	3.0
attn(t=1.0, h=128)	max	0.321	0.622	3.0
attn(t=1.0, h=128)	mean	0.319	0.625	3.0
max	mean	0.312	0.606	3.0
attn(t=0.5, h=256)	attn(t=1.0, h=128)	0.318	0.620	3.3
attn(t=1.0, h=256)	mean	0.309	0.606	3.3
max	attn(t=1.0, h=128)	0.306	0.599	3.3
attn(t=0.5, h=128)	max	0.316	0.609	3.0
attn(t=0.5, h=256)	max	0.315	0.611	3.0
mean	max	0.312	0.609	3.0
max	attn(t=0.5, h=256)	0.314	0.609	3.0
attn(t=0.5, h=256)	attn(t=0.5, h=256)	0.308	0.612	3.0
attn(t=1.0, h=256)	max	0.313	0.609	3.0
max	attn(t=1.0, h=256)	0.304	0.601	3.0
max	attn(t=0.5, h=128)	0.317	0.604	3.3
max	max	0.249	0.540	4.7

B.3 LLM-Based Semantic Evaluation Prompt

The prompt below was used by the LLM judge to score semantic similarity between a query description and a retrieved log description.

```
You are an expert evaluator of ADAS (Advanced Driver
   Assistance Systems) driving scenarios.

Your task is to assess how semantically similar two ADAS
   scenarios are for retrieval, comparison, and benchmarking.

Evaluate similarity using the following criteria. Each
   criterion must be scored from 0 to 5, where: 0 = no
   similarity 5 = near-identical

Scoring criteria:

Traffic composition and traffic density (overall traffic
   density such as sparse, moderate, or heavy traffic,
   presence of specific vehicle categories if mentioned e.g.,
    trucks, buses, motorcycles, emergency vehicles)
Road environment and surrounding context (road type, urban/
   rural setting, surrounding structures, vegetation, built
   environment, open areas)
```

```
Road infrastructure and layout (lane configuration, markings,
    signs, traffic lights, intersections, merges, exits,
   construction-related changes)
Weather, visibility, and time-related conditions (weather
   conditions, road surface state, visibility, lighting
   conditions, shadows, time-of-day indicators)
Dynamic elements and non-vehicle road users (pedestrians,
   cyclists, animals, or other vulnerable road users, their
   presence, behavior, and interaction with traffic)
Special features and notable scenario-specific elements (
   construction zones, emergency vehicles, temporary
   obstacles)

Rules:

Base scores strictly on the information explicitly present in
    the scenarios.
Ignore differences in wording, phrasing, or narrative style.
Do not assume or infer missing details.
Favor semantic and situational meaning over surface-level
   similarity.
Output format (do not add extra text):

Return exactly one JSON object with this shape:

{
    "scores": {
        "traffic_composition": x,
        "environment_context": x,
        "road_infrastructure": x,
        "weather_time_conditions": x,
        "dynamic_elements": x,
        "special_features": x
    },
    "final_similarity_score": S
}

Where each x is a number from 0 to 5 and S = (sum of the six
   scores / 30) x 100.
Do not wrap the JSON in markdown fences.
```

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY