BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

LARGE LANGUAGE MODELS FOR TRAFFIC SURVEILLANCE VIDEO UNDERSTANDING

VELKÉ JAZYKOVÉ MODELY PRO VYHLEDÁVÁNÍ V DOPRAVNÍCH VIDEÍCH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. MICHAL PYŠÍK**
AUTOR PRÁCE

SUPERVISOR **Ing. ONDŘEJ KLÍMA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2025

Master's Thesis Assignment

164073

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Pyšík Michal, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Mathematical Methods
Title:	**Large Language Models for Traffic Surveillance Video Understanding**
Category:	Artificial Intelligence
Academic year:	2024/25

Assignment:

1. Get familiar with existing Large Language Models for video understanding, and focus on approaches suitable for application to traffic surveillance videos.
2. Design a system to search for video events based on a user-supplied text description.
3. Implement the designed system as a web application using available tools and libraries.
4. Evaluate chosen search approaches and the implemented system using the provided data.
5. Present the achieved results.

Literature:
Dle doporučení vedoucího.

Requirements for the semestral defence:
Points 1, 2, and partially 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Klíma Ondřej, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2024
Submission deadline:	21.5.2025
Approval date:	12.11.2024

Abstract

Existing systems for searching and analyzing traffic surveillance footage often rely on predefined event detection methods and lack the ability to interact with users through natural language. The main goal of this thesis is to create such system by utilizing multimodal large language models (MLLMs) and related technologies, namely multimodal embedding models, both of which have seen rapid advancements in recent years. The system is designed to support multiple models of both types, enabling flexible integration and comparison. Furthermore, the available models are benchmarked specifically in the domain of traffic footage to evaluate their performance and suitability for practical deployment.

Abstrakt

Existující systémy pro vyhledávání a analýzu záznamů z dopravních kamer často spoléhají na předem definované metody detekce událostí a postrádají schopnost interagovat s uživateli prostřednictvím přirozeného jazyka. Hlavním cílem této diplomové práce je vytvořit takový systém s využitím multimodálních velkých jazykových modelů (MLLM) a souvisejících technologií, konkrétně multimodálních embedovacích modelů, přičemž oba typy modelů v nedávných letech zaznamenaly rychlý rozvoj. Systém je vytvořen tak, že podporuje výběr mezi více modely obou typů, čímž umožňuje jejich flexibilní integraci a porovnání. Všechny dostupné modely jsou dále porovnány specificky v oblasti dopravních záznamů za účelem zhodnocení jejich výkonu a vhodnosti pro praktické nasazení.

Keywords

traffic, videos, CCTV, system, search, analysis, multimodal, model, embedding, large language model, CLIP, GPT, benchmark, machine learning, artificial intelligence, cars, traffic signs, Python

Klíčová slova

doprava, videa, CCTV, systém, hledání, analýza, multimodální, model, embedding, velký jazykový model, CLIP, GPT, porovnání, strojové učení, umělá inteligence, auta, dopravní značky, Python

Reference

PYŠÍK, Michal. *Large Language Models for Traffic Surveillance Video Understanding.* Brno, 2025. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ondřej Klíma, Ph.D.

Rozšířený abstrakt

Frekvence nasazování dopravních sledovacích kamer v posledních letech stále roste, CCTV záznamy se staly klíčovým nástrojem pro monitorování veřejných prostor, analýzu dopravních vzorců, detekci nehod a vymáhání předpisů. Většina současných systému pro vyhledávání a analýzu dopravních videí však často spoléhá na předem definované metody detekce událostí a postrádá možnost interakce s uživateli prostřednictvím přirozeného jazyka.

V nedávných letech se velkého rozvoje, popularity a hromadného nasazení dočkaly velké jazykové modely (LLM). Multimodální rozšíření těchto modelů jim umožňují kromě textu pracovat také s daty jiných typů (modalit), jako jsou například obrázky, zvuk či videa. Jejich rozvoj ovlivnil také vývoj souvisejících modelů z oblasti strojového učení—například multimodálních embedovacích modelů, které umožňují zakódování dat z více ruzných modalit do stejného vektorového prostoru. Primárním cílem této práce je vytvořit pomocí zmíněných modelů systém, který umožní uživatelům efektivně vyhledávat události či objekty v kolekci dopravních videí a také tato videa interaktivně analyzovat pomocí přirozeného jazyka.

Práce začíná úvodem do problematiky, který pokrývá témata jako CCTV dopravní záznamy, techniky zpracování obrazu a videa či vybrané typy úloh z oblasti strojového učení, které úzce souvisí s funkcionalitou navrženého systému. Dále jsou představeny multimodální LLM (MLLM) a multimodální embedovací modely, které tvoří jádro systému. Vybrané konkrétní state-of-the-art modely obou typů jsou poté společně s existujícími podobnými systémy představeny v rámci rozboru současných řešení, přičemž většina popsaných modelů je k dispozici pro použití v implementovaném systému.

Návrh i implementace systému jsou důkladně popsány. Popis začíná samotným konceptem dvoufázového procesu, který deleguje vyhledávání na multimodální embedovací model spolu s vektorovou databází a analýzu nalezených či manuálně specifikovaných úseků videí na MLLM přizpůsobený pro práci s videi. Systém je implementován formou webové aplikace s jednoduchým a přehledným grafickým uživatelským rozhraním, přičemž zvolené modely umožňují systém provozovat i lokálně na moderním domácím počítači. Implementovaný systém nabízí uživateli výběr mezi čtyřmi různými multimodálními embedovacími modely a čtyřmi různými video-MLLM, přičemž je navržen tak, aby umožňoval relativně snadné přidání podpory dalších modelů. Výběr dostupných modelů je v práci řádně popsán a odůvodněn.

Kromě samotného systému je jedním z hlavních přínosů této práce také porovnání všech použitých modelů v doméně dopravních videí. Multimodální embedovací modely byly porovnány prostřednictvím vyhledávání obrázků na základě textu na datasetu obsahujícím fotografie konkrétních modelů aut, přičemž bylo také zohledňováno zda vybrané obrázky odpovídají alespoň správné automobilce. Porovnání těchto modelů bylo podobným způsobem provedeno také na datasetu obsahující fotky dopravních značek (i se zohledněním jejich kategorií) pořízených v České republice. Dostupné MLLM modely byly porovnány na základě odpovídání na otázky týkající se příslušných dopravních videí, přičemž otázky pokrývaly nejen popis dění ve videu, ale také predikci budoucího vývoje, zpětné uvažování a jiné složitější úvahy. Na závěr je celý systém stručně vyhodnocen za použití modelů, které si v uvedených porovnáních vedly obecně nejlépe.

Large Language Models for Traffic Surveillance Video Understanding

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Ondřej Klíma, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis. OpenAI's ChatGPT service, specifically the GPT-4o model, was used for research paper summarization and information retrieval; however, all information obtained through this service was subsequently reviewed and verified for accuracy.

. . . . . .
Michal Pyšík
May 16, 2025

Acknowledgements

I would like to express my gratitude towards my supervisor, Ing. Ondřej Klíma, Ph.D., for his continuous support during the writing of this thesis, and for providing the relevant data, namely preparing the dataset containing images of Czech traffic signs. I would also like to thank my family, partner, and friends for their constant support throughout my studies.

Contents

1 Introduction **2**

2 The problem domain **3**
2.1 CCTV footage of traffic . . . 3
2.2 Video and image processing techniques . . . 5
2.3 Temporal sentence grounding in videos . . . 10
2.4 Multimodal search . . . 12

3 Multimodal embedding models and multimodal LLMs **14**
3.1 Multimodal embedding models . . . 14
3.2 Multimodal Large Language Models (MLLMs) . . . 18

4 Existing solutions and suitable models **23**
4.1 Whole-system solutions . . . 23
4.2 Suitable multimodal embedding models . . . 26
4.3 Suitable MLLM-based models . . . 29

5 Concept of the system **33**
5.1 The main idea . . . 33
5.2 System architecture concept . . . 35

6 Implementation of the system **38**
6.1 Overview of system components . . . 38
6.2 The available AI models . . . 39
6.3 Backend of the web application . . . 43
6.4 Frontend of the web application . . . 47

7 Benchmarks of the available models and evaluation of the system **51**
7.1 Benchmarks of the multimodal embedding models . . . 51
7.2 Benchmark of the MLLMs . . . 59
7.3 Evaluation of the system . . . 63

8 Conclusion **66**

Bibliography **68**

A Contents of the external attachment **78**

B Poster **79**

Chapter 1

Introduction

The deployment of traffic surveillance cameras has been steadily increasing in recent years, driven by advancements in technology and a growing emphasis on road safety, efficient traffic management, and crime prevention. CCTV footage has become a critical resource for monitoring public spaces, analyzing traffic patterns, detecting incidents, and enforcing regulations. However, existing systems for searching and analyzing traffic footage often rely on predefined event detection methods and lack the capability to interact with users through natural language.

In recent years, large language models (LLMs) have gained significant attention and achieved remarkable progress in performance and capabilities. Notably, the capabilities of multimodal LLMs (MLLMs) go beyond text processing, as they also integrate information from other modalities such as images, videos, or audio. Their popularity also accelerated advancements in related technologies, that often correspond to a part of their architecture, such as multimodal embedding models, which map inputs from multiple different modalities into a shared embedding space. The aim of this thesis is to utilize video-MLLMs and related technologies (multimodal embedding models) to create an efficient and intuitive system for searching and analyzing videos of traffic.

The theoretical part of the thesis begins with an analytical overview of the problem domain in chapter 2, covering topics such as CCTV footage of traffic, video and image processing techniques, and certain machine learning tasks closely related to the indended system functionality. Chapter 3 introduces multimodal large language models (MLLMs) and multimodal embedding models, both of which are at the system's core. Chapter 4 presents research of existing solutions, covering both existing whole-system solutions similar to the target system, and existing MLLMs and multimodal embeding models that were evaluated as suitable for the system and most of which are made available for use there.

The practical part of the thesis begins with the concept of the system in chapter 5, which introduces and expands upon the idea of a two-step pipeline that utilizes an image-text multimodal embedding model for search and a video-MLLM for analysis. Chapter 6 describes the implementation of the system, that also allows the user to choose between four different AI models of both types. Finally, chapter 7 covers the scientific contribution of the thesis in the form of benchmarks of the available models in the domain of traffic footage. The multimodal embeding models were benchmarked on image-text retrieval, and the MLLMs were benchmarked on video question answering (VQA). Chapter 7 also includes a brief evaluation of the system with the best-performing models configured.

Chapter 2

The problem domain

The goal of this chapter is to introduce and analyze the problem without too much focus on specific technologies that will be used to create the system itself.

Section 2.1 analyzes the characteristics of CCTV footage, which is the type of data the system aims to work with, while section 2.2 discusses some general video (and image) processing techniques which are relevant in this context. Moving towards some relevant specific problematics, sections 2.3 and 2.4 introduce temporal sentence grounding, a rather modern machine learning task type, whose goal is to localize a video section based on a given text query description of the scene, and multimodal search, the idea of searching through data not only based on text, but also searching by images or other media.

2.1 CCTV footage of traffic

Closed-circuit television (CCTV), also known as video surveillance [40], is commonly used to monitor both public (for public safety, traffic monitoring, special event security, etc.) and private (business or residential security, private institutions, etc.) spaces in many parts of the world. As opposed to broadcast television, where the signal is transmitted openly, the signal from (a given set of) CCTV cameras is transmitted only to a specific, restricted set of monitors and recording devices, forming a "closed" system [14]. CCTV camera systems vary in their price ranges, capabilities, use cases and other attributes, and can be deployed both indoors and outdoors.

This thesis is concerned specifically with CCTV footage of traffic, which is essential for monitoring traffic flow, identifying accidents, and enforcing traffic laws. Traffic surveillance cameras are generally placed in areas where traffic flow is heavy or needs close monitoring, like intersections, highways and major roads, toll booths (mainly for license plate recognition), or parking lots and garages. Nowadays, advanced algorithms are often used to analyze the footage, automating tasks like object detection, license plate recognition and identifying traffic violations. Most of the modern algorithms tend to be AI-based, and some of them can be even used in real-time. The main focus of the thesis is creating a system, that allows for a very free and flexible analysis of multiple recordings, including the option to localize a described event (or object). Although the system is meant to be able to process general CCTV footage of traffic, independent of location, resolution and other characteristics, auxilary footage from street Božetěchova was supplied for the work (two batches of videos that share the same camera location, but differ in resolution and bitrate),

and will be used as a reference most of the time. An example frame from footage captured at daytime is shown in figure 2.1.

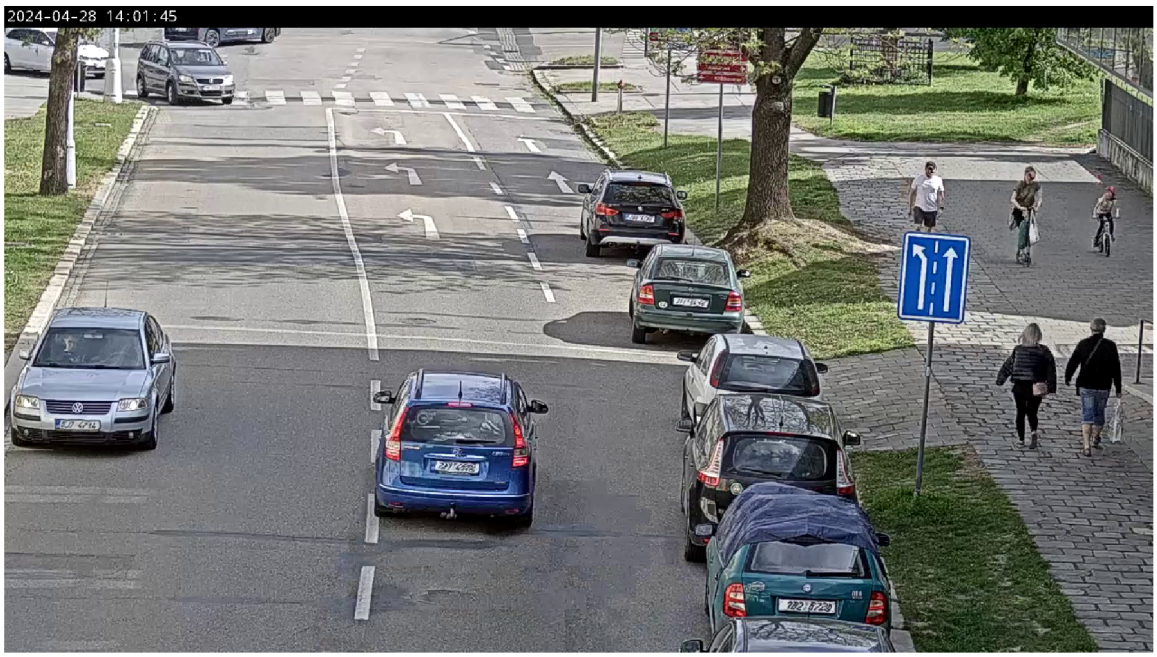

Figure 2.1: An example of daytime CCTV footage from Božetěchova street. Visible license plates and people's faces have been blurred for privacy reasons.

Properties and limitations of CCTV traffic footage

Most footage from CCTV cameras shares some common characteristics. Since the system is meant to work specifically with this kind of video files, it would be wise to prepare for what properties and limitations to expect from such footage:

- **Resolution and bitrate** – Since these cameras are designed for monitoring places in a 24-hour continuous operation, the resolution and bitrate tend to be rather limited compared to other video-media. For example, the first batch of videos from Božetěchova share a resolution of only 640x360 pixels (with 220–450 kbps bitrate at 5 FPS), and the second batch has HD resolution of 1280x720 pixels (with 800–1700 kbps bitrate at 5 FPS). This limits the amount of detail that can be reasonably processed.

- **Camera position** – Cameras are usually mounted in fixed positions to cover large areas. This can result in footage captured at oblique angles, with objects like vehicles or pedestrians appearing small or partially obscured. Combined with the lower resolution, this may greatly limit the effectiveness of object recognition, facial recognition, and other types of specialized algorithms. The camera placement may also create blind spots, further limiting the visible area.

- **Sound and color** – Most CCTV footage lacks audio recording due to legal restrictions or technical limitations [34]. Additionally, while modern systems often capture color video, lighting conditions (e.g., nighttime or low-light settings) can reduce the accuracy of color representation or force cameras to switch to grayscale modes, as can be seen in figure 2.2.

- **Video length** – CCTV cameras record continuously (this is the case with the footage from Božetěchova) or in short bursts triggered by motion detection. The resulting files can range from a few seconds to hours, making it challenging to process long-duration footage efficiently. This means that either some algorithm capable of handling videos of variable length, or at least some kind of data preprocessing is needed.

- **Frequency of important events** – Traffic-related footage often includes long periods of minimal activity interrupted by sudden events (e.g., a car going by, more rarely an accident), this is even more prominent in night-time footage (see figure 2.2). This may be a major problem, since currently, a lot of state-of-the-art AI-based video processing techniques use visual entertainment, like movies, which obviously tend to be filled with meaningful events, changes of scenes and dialogues, as main benchmarks for long video understanding (see [84] and [99]). The system should be capable of efficiently filtering out irrelevant portions of a video, while not accidentally overlooking critical moments, to ensure meaningful analysis.

Figure 2.2: An example of night-time CCTV footage from Božetěchova street. The location and angle of the camera is the same as in figure 2.1.

2.2 Video and image processing techniques

From a certain point of view, video is nothing more than a series of still images transitioning between one another at a certain frequency (frames per second, FPS for short), at least as long as audio can be ignored, which is the case here (see section 2.1). Exactly for this reason, an overview of image processing techniques is present in this section too and located right before temporal video processing techniques, both with a special focus on CCTV footage. In general, what separates advanced video analyzing algorithms from only image analyzing ones (or video analyzing algorithms limited to frame-based analysis), is

the ability to understand temporal relationships (like motion). A video search engine, or a similar system, limited to frame-based analysis, could probably find a specified object (e.g., an orange SUV) and even list the corresponding timestamp(s), but it certainly wouldn't be able to accurately understand time-sensitive queries (e.g., an orange SUV doing a U-turn while using its left turn signal), compared to a system based on temporal analysis.

Image processing

Image processing techniques relevant for CCTV footage range from a simple grayscale conversion, to a cutting edge LLM-based multimodal agent (see section 3.2) answering complex queries about the image. This summary, some parts of which are inspired by [30], aims to introduce primarily the more common and general ones. Please note that while these techniques are very often used in the context of processing CCTV video, the reason they are mentioned here is that the video processing summary will be more focused on temporal-analysis, whose input data can be also preprocessed by these techniques.

Some techniques perform operations over entire images, without the usual focus on (specific) objects. Their usual goal is to prepare the image for further analysis and processing [105]. The most basic example is probably the already mentioned conversion of an RGB(A) image to grayscale, a process of converting each pixel from a 3-channel (4 channels when including alpha[1]) to a single-channel representation using a specified formula[2]. Similar in nature are simple preprocessing steps, like color normalization or correction, which fall under image enhancement techniques. Considering the more complex methods, image enhancement improves visual quality through techniques like noise reduction, sharpening, and super-resolution[3]. These tasks often rely on advanced CNN-based (Convolutional Neural Network) machine learning models. Notably, technologies like NVIDIA's DLSS[4] (Deep Learning Super Sampling) demonstrate how deep learning can upscale images with remarkable detail in real-time [96].

Possibly more interesting are techniques that focus on the objects present in the image, which is often already preprocessed. **Object detection** is a computer vision technique that identifies and locates objects in an image. Soft detection refers to just the detection of the presence of an object, while hard detection includes localization of the object too. There are multiple approaches to object detection, the two most common ones are region proposal-based methods [24] and single-shot detection (SSD) methods [52]. Although object detection is usually used in the context of videos, and some of these methods can be even used in real-time (this applies more to SSD), they still don't process temporal information. Region-based proposal methods first identify areas in an image that are likely to contain objects (these are called regions of interest, ROIs for short), these regions are then passed to a classifier[5] (usually a CNN), often followed by an additional step of refining the ROIs to better align with the object boundaries. The most famous model of this type is probably Faster R-CNN [77], which is very commonly used as a benchmark in object detection. Single-shot detection methods, on the other hand, perform object localization and classification in a single forward pass through the network (unlike region-based proposal

[1] Alpha is an additional channel that specifies the opacity of a given pixel.

[2] The most common is the NTSC formula: $0.299 \times \text{Red} + 0.587 \times \text{Green} + 0.114 \times \text{Blue}$.

[3] The process of upscaling an image to a higher resolution, often using methods like deep learning.

[4] https://www.nvidia.com/en-us/geforce/technologies/dlss/

[5] Classification is the process of assigning a class label to a data sample based on its features, such as identifying the type of object in an image (e.g., cat, dog, car). The term "classifier" refers to a model made for this purpose.

methods, that use a two-stage architecture). Their streamlined single-stage architecture makes them much faster (which makes them the popular choice for real-time applications), usually for the cost of sacrificing some accuracy. An image is first passed through a backbone network (like VGG [81], ResNet [27], etc.) to extract feature maps, then divided into a grid, where each grid cell is responsible for detecting objects whose center lies within that cell. Each cell predicts multiple bounding boxes (called anchors) with predefined aspect ratios and scales, and assigns class probabilities for the objects within those boxes. Finally, Non-Maximum Suppression (NMS) is applied to eliminate redundant boxes, keeping only the most confident detections. The most famous SSD models include YOLO (You Only Look Once; see figure 2.3) [75] and SSD (Single Shot MultiBox Detector) [52].

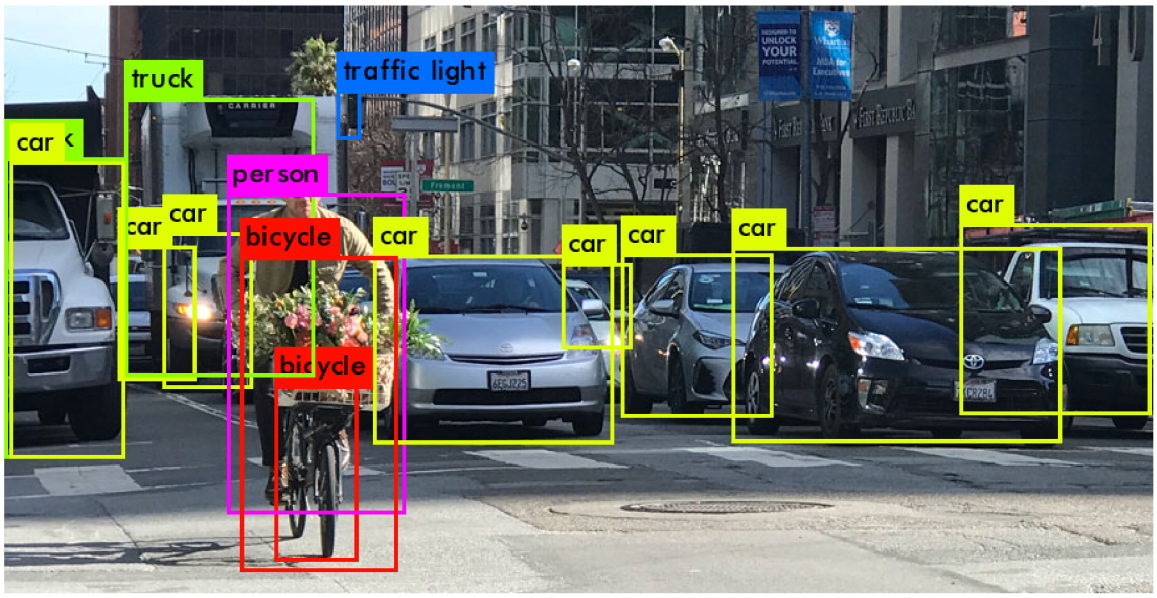

Figure 2.3: A sample output of the YOLO object detection process [104, fig. 5].

There are some specific methods, that can be considered special cases of object detection specialized for a single certain type of objects. In the case of face detection (and recognition), the object is a human face. The task involves detecting the face in an image (locating it with a bounding box) and often identifying or verifying the identity of the person. While face detection is just specialized object detection, face recognition goes a step further by matching the detected face to a database of known faces [66]. In the context of CCTV footage, face recognition is only feasible for close-up shots or footage taken with cameras specialized for this kind of task. Given that cameras monitoring traffic are usually placed at a distance and from an elevated position to cover more area, face recognition may not be a viable option. On a different note, very specific to traffic footage is automated number plate recognition (ANPR) [58], also known as automated license plate recogntinion (ALPR). This technology is used to automatically read and identify the license plates of vehicles. It is widely used in various applications, particularly in traffic monitoring, law enforcement, and security systems. The process typically involves capturing (and preprocessing) an image, detecting the license plate, segmenting individual characters, and using optical character recognition (OCR) to read and convert them into a digital format.

Some other techniques worth mentioning are focused on objects in a different way—they separate objects from each other and from the rest of the image. These operations are often

done in advance, to inform object detection and related tasks [67]. The most notable is image segmentation, which splits an image into various tiny parts called segments. These segments, usually representing objects (but the sky, or a road, etc. can be segments too), are colored differently in the produced output, to easily differentiate between them. Image segmentation can be classified into three main categories: semantic segmentation (objects of the same category share the same color; non-object things like the sky are included), instance segmentation (each separate object is considered unique; non-object things are ignored), and panoptic segmentation (like instance segmentation, but includes non-object things). These differences are visually demonstrated in figure 2.4. Another technique similar to image segmentation is background subtraction, which extracts moving objects and removes the rest of the image (the background).

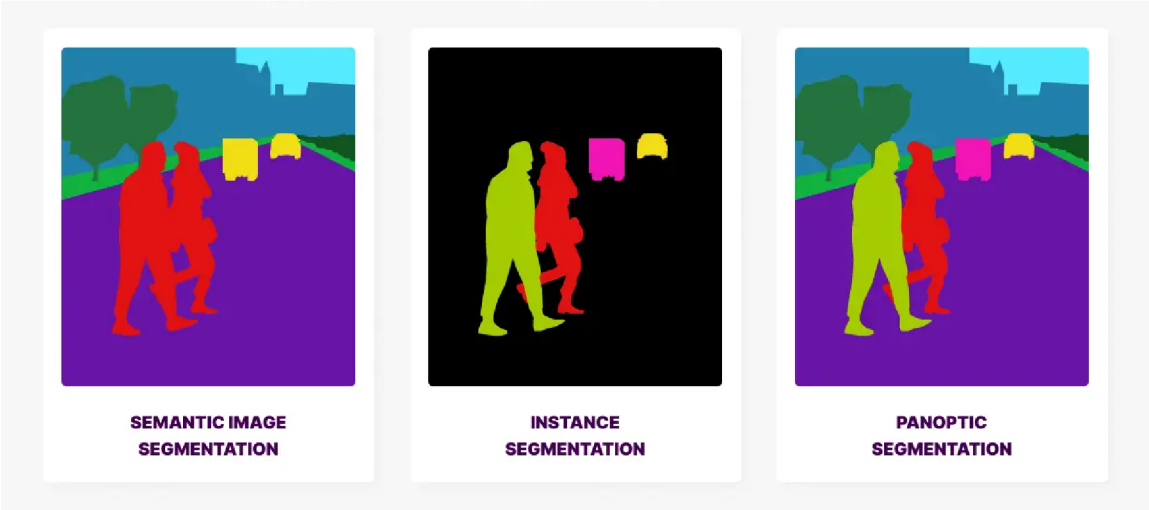

Figure 2.4: Comparison of the different types of image segmentation [61].

Temporal video processing

Temporal relations are the main factors for developing a strong understanding of content in a video. The models capable of this kind of processing are usually vision transformers[6] [19], however, 3D CNNs, RNNs (Recurrent Neural Networks), and hybrid architectures remain competitive in applications where computational efficiency or smaller datasets are priorities. Although temporal video processing is more "cutting-edge" than frame-based techniques, processing CCTV footage may not benefit from them that much, since the temporal complexity is relatively low (cars traveling along predictable paths, pedestrians walking in routine patterns) compared to scenarios like sports or movies.

One of the main differences between image and video processing is the inclusion of an additional axis, time, to the input. There are two main approaches for extracting tokens (discrete units of data) from a video or embedding[7] a video clip [37]:

[6]Transformers are a type of deep learning model designed for sequential data, originally introduced in natural language processing, that rely on a mechanism called self-attention to weigh the importance of different parts of the input sequence when making predictions. Vision transformers adapt this architecture for image data by dividing images into patches (like words in a sentence) and processing them as a sequence, enabling advanced understanding of spatial relationships in visual tasks.

[7]Converting objects or abstract concepts into mathematical representations, capturing their inherent properties and relationships.

- **Uniform frame sampling** – A method of tokenizing the input video by sampling frames at regular intervals, embedding each frame independently (using the same methods as in image processing), and concatenating all these tokens together. While this approach doesn't directly account for all temporal dynamics, it simplifies the analysis pipeline, ensures consistency in input size, and reduces the volume of data without losing key temporal information.

- **Tubelet embedding** – A more advanced method that extends 2D image embedding to 3D (it corresponds to a 3D convolution), by creating "tubelets"—non-overlapping sequences of bounding boxes over multiple frames (spatiotemporal "tubes"). After these tubes, capturing both spatial (image) and temporal (motion) information, are extracted from the video, they get flattened to build video tokens. This means that spatiotemporal information is fused during tokenization, instead of leaving the fusing process to the model itself.

One of the video processing techniques worth mentioning is optical flow, which quantifies the motion of objects between consecutive frames. Optical flow algorithms attempt to capture the apparent motion of brightness patterns in the image, therefore they might exhibit too much sensitivity to lighting changes or noise in CCTV footage (although [70] shows how successful a model consisting of a CNN and LSTM can be in those conditions). Optical flow-based analysis can be done using both traditional methods (e.g., Lucas-Kanade method [54]) and machine learning methods (e.g., FlowNet [20]). Its applications in traffic footage can be, for example, identifying vehicle trajectories, measuring their speed or detecting motion irregularities, and detecting congestions or other stopped vehicles. A visualization of optical flow (the green vectors pointing in the direction of motion) in traffic footage is shown in figure 2.5.

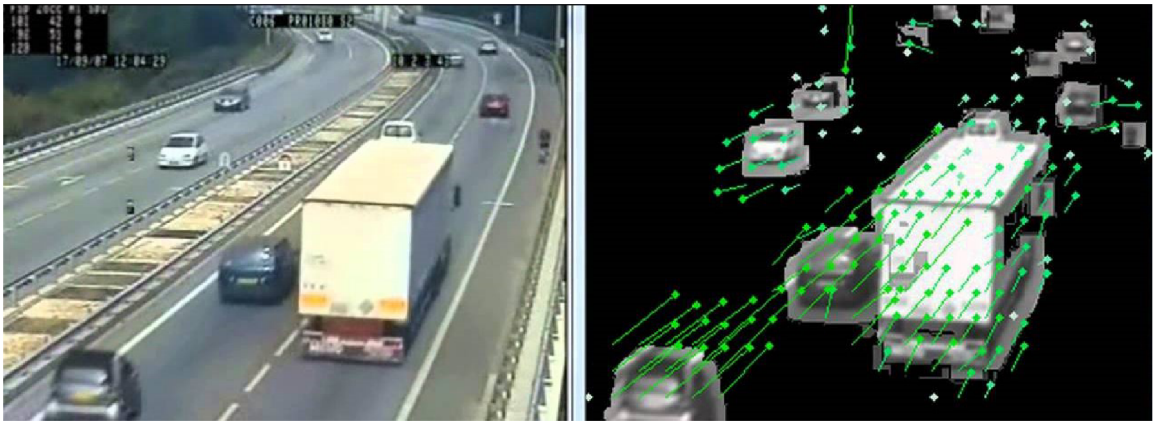

Figure 2.5: Visualization of optical flow in traffic footage [25].

Other relevant techniques include using tubelet-embedded data as inputs for 3D CNNs (like I3D [7]) or vision transformers (like TimeSformer [5]), that can be set up to perform different types of tasks, like classification or detection of some object or event. In the case of traffic footage, good examples might be detecting time-sensitive actions (e.g., illegal turns, pedestrian crossing violations), or a better understanding of vehicle interactions (e.g., a vehicle avoiding a pedestrian). Worth mentioning is also action recognition, which uses predefined action labels to classify events like running, waving, or jumping—this could potentially be applied to pedestrians in CCTV footage. Another relevant method is

extracting features from the video, and using data that consists of a timeseries of these features as inputs for RNNs, or some of their more advanced versions like LSTM (Long Term Short-Term memory) or GRU (Gated Recurrent Unit) [106]. Even though these recurrent neural network architectures have been largely replaced by transformers in recent times, they might still provide more lightweight solutions to problems that include detecting long-term traffic patterns, like gradual lane shifts, or predicting traffic density and anomalies over time.

2.3 Temporal sentence grounding in videos

Most of the information in this section is based on a recent survey by Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou [110]. Temporal sentence grounding in videos (TSGV), also known as natural language video localization (NLVL) or video moment retrieval (VMR), aims to retrieve a temporal moment that semantically corresponds to a language query from a video. It is a relatively recent and specialized task in the field of machine learning (particularly in the domains of computer vision and natural language processing), whose inputs are a single untrimmed video and a text query (sentence) describing a particual action, event or condition that happens somewhere in the video, and whose output is typically a temporal interval (or multiple intervals), for example `[32s, 35s]` (but the output can also contain sampled frames, etc.). Even though the main problem this thesis discusses is localizing moments in *multiple* videos, TSGV is still one of the most important subproblems. Other problems combining vision and language contain visual grounding (locating the most relevant object or region in an image, based on a text query), video retrieval (given a set of videos and a text query, retrieve the most relevant video and/or rank all the videos based on their relevance to the query), video question answering (VideoQA; answering questions in text form about objects or events happening in a given video), and video grounded dialogue (VideoDial; to conduct a multi-turn conversation, based on the visual and audio aspects of a given video).

Although TSGV methods may feature various sophisticated architectures, they conceptually consist of five to six fundamental components, as depicted in figure 2.6. Both the video and the text query are first processed by their corresponding preprocessor, the preprocessed inputs are then converted into feature representations by the feature extractors (textual feature extractor, visual feature extractor). In the next step, the feature encoder maps visual and textual features to the same dimension, and refines their feature representations by encoding their corresponding contextual information. Following this, the feature interactor learns cross-modal interactions between the video and query to fuse their representations, focusing on portions of the video most relevant to the query's semantics. Finally, the answer predictor identifies the start and end timestamps of the target moment based on the learned multimodal representations. Optionally, if a proposal generation module is used, it may produce candidate temporal segments within the video for consideration (at various positions in the pipeline)—based on this criterion, TSGV methods can be broadly classified into two main categories: proposal-based and proposal-free approaches. Proposal-based methods (further categorized into sliding window-based, proposal generated, and anchor-based methods) generate candidate temporal segments (proposals) and match them with the query to localize the target moment, whereas proposal-free methods (further categorized into regression-based and span-based methods) bypass this step, directly predicting the temporal boundaries of the target segment.

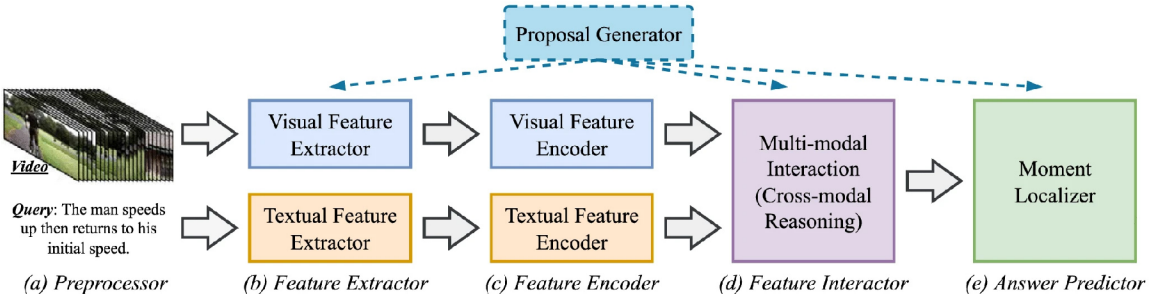

Figure 2.6: The general pipeline of TSGV methods [110, fig. 3].

Quality datasets are essential for building and evaluating TSGV models and they typically contain a collection of videos. Each video may come with one or more annotations; each annotation has a query corresponding to a specific moment in the video. Several benchmark datasets exist, such as DiDeMo [28], which contains over 10,000 videos about various human activities sourced from Flickr[8], TaCoS dataset [76], composed of cooking videos, or MAD [83], a large scale dataset containing 650 mainstream movies with over 1,200 hours of total video length.

Evaluation metrics for TSGV

Evaluation metrics for TSGV generally assess the overlap between predicted and ground-truth moments, with three widely used metrics:

- **Mean Intersection over Union (mIoU)** – Calculates the average IoU (a metric commonly used in object detection) across all queries, where IoU is the ratio of the intersection area to the union area between predicted and ground-truth temporal segments (see figure 2.7, IoU equals the length of the intersection interval divided by the length of the union interval). The formula is:

$$\text{mIoU} = \frac{1}{N_q} \sum_{i=1}^{N_q} \text{IoU}_i \tag{2.1}$$

  where N_q is the number of queries and IoU_i is the IoU for the i-th query.

- **Recall at n with IoU threshold m ⟨R@n, IoU@m⟩** – This metric evaluates the percentage of queries where at least one of the top-n predictions achieves an IoU greater than or equal to m. The formula is:

$$\text{R@}n, \text{IoU@}m = \frac{1}{N_q} \sum_{i=1}^{N_q} r(n, m, q_i) \tag{2.2}$$

  where $r = (n, m, q_i) = 1$ if at least one of the top-n predictions for query q_i satisfies $\text{IoU} \geq m$, and 0 otherwise.

- **Discounted Recall at n with IoU threshold m ⟨dR@n, IoU@m⟩** – This metric extends R@n by applying a discount based on the temporal distance between the

[8] https://www.flickr.com/

predicted and ground-truth boundaries, penalizing overlong predictions. The formula is:

$$\text{dR@}n, \text{IoU@}m = \frac{1}{N_q} \sum_{i=1}^{N_q} r(n, m, q_i) \cdot \alpha_{s_i} \cdot \alpha_{e_i} \tag{2.3}$$

where $a_{s_i} = 1 - |p_{s_i} - g_{s_i}|$ and $\alpha_{e_i} = 1 - |p_{e_i} - g_{e_i}|$, representing normalized temporal distances between predicted (p) and ground-truth (g) start (p_{s_i}, g_{s_i}) and end (p_{e_i}, g_{e_i}) timestamps.

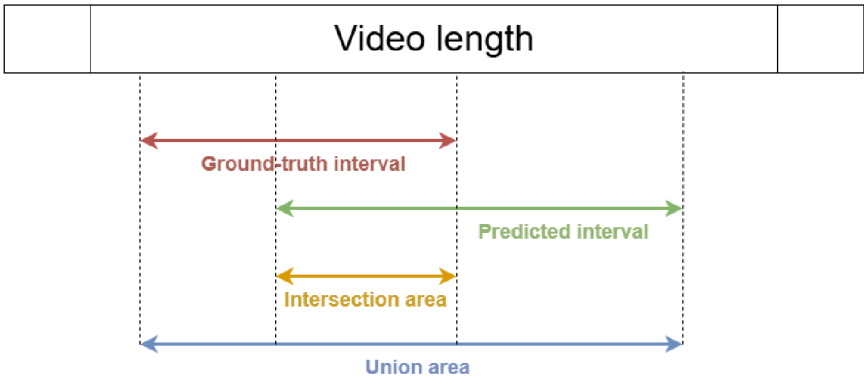

Figure 2.7: A visualization of how IoU is measured in the case of videos.

2.4 Multimodal search

Another concept that is very closely related to what the system is supposed to be able to do is multimodal search. Most people are familiar with conventional search engines that can find results based on a text query, which are typically limited to matching textual content or metadata to the input query. However, multimodal search expands the boundaries of traditional search systems by allowing queries to span multiple modalities, such as text, images, audio, and video, and retrieving results that align with the meaning or content of these different forms of data. Be aware that searching by any form of data other than text does not mean that the given search system is multimodal (e.g., a system where a user uploads an image, which is then used to search for similar images), as *multimodality* refers to the ability to handle different types of data (modalities), with a key emphasis on connections and meaning across the different modalities.

The most common and effective approach to multimodal search is to convert the given data of different possible types into the same embedding space [36], basically "translating them into the same language". The embeddings, which are just (usually fixed-size) vectors of floating-point values, can then be compared to find the best match(es) given an input embedding, and various operations can be performed on them too. In the case of this thesis, the multimodality of the system comes from combining images and text (see figure 2.8). The system may have the embeddings of some frames sampled from the traffic footage videos saved in a database, and once a user triggers a search by inputting a text query, the query gets converted to the same embedding space and compared to the embeddings present in the database. The structure of embeddings, and what relationships and other information they usually encode, will be explained later in section 3.1.

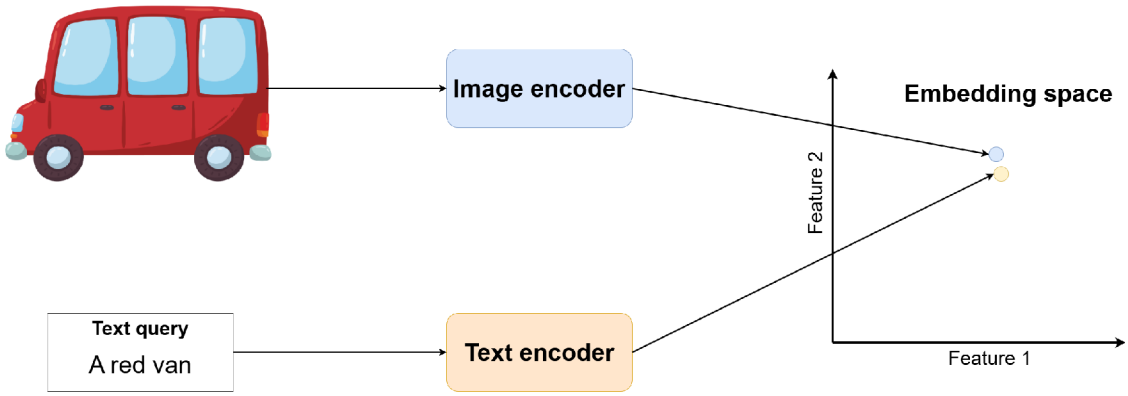

Figure 2.8: A simple visualization of converting an image and a semantically similar text to the same embedding space.

While mapping modalities to a shared embedding space is the most common approach for multimodal search, alternative methods include late fusion, where features from each modality are processed separately and combined later for decision-making, or early fusion, which integrates raw data from different modalities before processing them jointly. Furthermore, metadata-driven approaches and attention-based models that dynamically link modalities without explicit shared embeddings are also viable techniques [111].

Chapter 3

Multimodal embedding models and multimodal LLMs

This chapter introduces multimodal embedding models and (multimodal) large language models (LLMs), two types of machine learning models highly relevant in modern AI research and applications. To understand the relevance of these two types of models in the thesis' context, it is recommended to read the beginning of chapter 5 (specifically section 5.1) first.

Section 3.1 introduces multimodal embedding models, with an emphasis on image-text retrieval, while also explaining the structure and meaning behind embeddings. Section 3.2 dives into LLMs, including both their general overview, and a focus on their multimodal extensions, which include multimodal LLMs capable of understanding videos.

3.1 Multimodal embedding models

Following on from section 2.4, multimodal embedding models are tools that enable the transformation of data of various types into a unified representation space. By learning to represent and align data from multiple modalities, they can be used to perform cross-modal tasks, such as retrieving an image using a text query or generating descriptive text for a video. These models are also increasingly used in commercial applications, including content search, recommendation systems and video indexing [82]. Possibly the most widely used (standalone) multimodal embedding model is OpenAI's CLIP (Contrastive Language-Image Pretraining, see section 4.2), which works with text and images, and shows impressive zero-shot capabilities (performing tasks without task-specific fine-tuning) [71]. Apart from standalone models, learning (multimodal) embeddings is a crucial part of every transformer architecture [90] and also of some other machine learning model types.

Structure of embeddings

Embeddings are compact, high-dimensional vector representations encoding the semantic meaning of input data. Word embeddings were first introduced in the context of natural language processing (NLP) around the early 2000s, and the key breakthrough came with the development of Word2Vec by Tomáš Mikolov and his colleagues at Google in 2013 [60]. Their work started a fundamental shift in how words were represented in machine learning,

moving away from sparse, high dimensional vector representations (like one-hot encoding[1]), to dense, low-dimensional vectors that capture semantic relationships between words. Later, transformers popularized contextualized embeddings, in which the same word can be encoded in different ways based on context (e.g, a baseball bat versus a bat as an animal) [51].

Embedding vectors are usually 128 to 1024-dimensional vectors of floating point (sometimes integer) values, where each dimension corresponds to a value of some latent feature that emerges from training, so each word (or concept in general) is represented by a point in an N-dimensional space (as previously shown in figure 2.8). For simplicity and learning purposes, it is often suggested that the dimensions capture explicit attributes like size (small in one direction, big in the other direction), the ability to fly, masculinity, etc., but in reality, the individual dimensions usually don't have a clear human-interpretable meaning. Instead, the model tries to figure out the best way to represent words in a space where similar words are closer together (semantic alignment), and the resulting dimensions are a byproduct of this process [80]. Relationships in the embedding space are geometric in nature, so metrics like cosine similairity (3.1) or Euclidean distance (3.2) can be used to measure proximity and relevance between words based on the following formulas:

$$\text{Cosine similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{3.1}$$

$$\text{Euclidean distance}(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \tag{3.2}$$

where $\mathbf{u}, \mathbf{v}$ are two n-dimensional embedding vectors and i is used to index their dimensions. Vector operations demonstrate how embeddings encode analogical reasoning, a very famous example is $king - man + woman \approx queen$ (or *woman is to queen as man is to king*), which is visually demonstrated in figure 3.1.

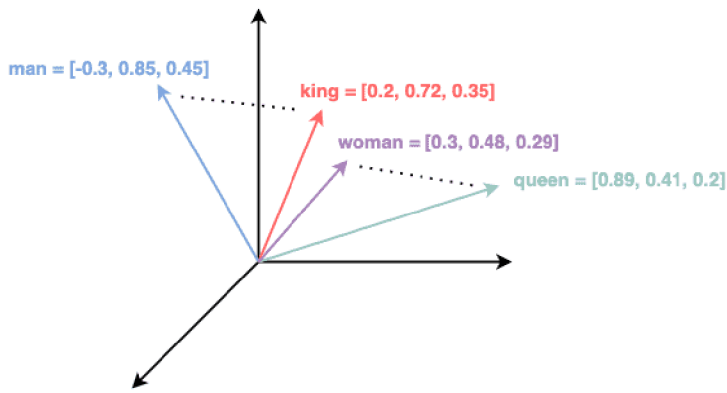

Figure 3.1: Embeddings of words "man", "king", "woman" and "queen" in a 3-dimensional space [102].

[1]In one-hot encoding, each word is converted to a vector of the same length as the number of all words in the dictionary, with a single vector component with value 1 representing the word, and the rest set to 0.

Underlying mechanisms and training

The architecture of a (standalone) multimodal embedding model needs to have a separate encoder for each compatible data type. An encoder is the component that processes input data and transforms it into an embedding. Here is how encoders are usually implemented for the two data types that are relevant for this thesis:

- **Text encoder** – It can be a simple lookup table (e.g., Word2Vec [60]), or based on a complex architecture of some transformer-based model (e.g., BERT [17]) that leverages self-attention[2] mechanism to generate context-aware embeddings. RNN-based architectures (like LSTM or GRU [106]) are sometimes also used. The text is tokenized and the tokens are mapped to dense embedding vectors, and in the case of transformers, the attention mechanism is also used to extract contextual features.

- **Image encoder** – Usually implemented using a CNN (e.g., ResNet [27]) or a vision transformer (ViT [19]). In the case of ViT, the image is first divided into patches. Then, convolution (CNN) or self-attention layers (ViT) are used to generate feature maps. The feature maps are then flattened or pooled into compact embeddings.

Multimodal embedding models are typically trained using supervised, self-supervised, or semi-supervised learning strategies. The encoders (usually a pre-trained model) for each modality are usually fine-tuned together to bind their latent space representations using a contrastive loss function [78] (alternatives include triplet loss, etc.). Minimizing contrastive loss during training essentially means that representations of similar examples across modalities in the joint vector space are pulled closer together, while distinct examples are pushed apart. The exact formula for contrastive loss Le Cunn came up with in 2005 [26] is:

$$\begin{aligned} \mathcal{L}(W) &= \sum_{i=1}^{P} L(W, (Y, \overrightarrow{X_1}, \overrightarrow{X_2})^i) \\ L(W, (Y, \overrightarrow{X_1}, \overrightarrow{X_2})^i) &= (1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \end{aligned} \tag{3.3}$$

where $(Y, \overrightarrow{X_1}, \overrightarrow{X_2})^i$ is the i-th labeled sample pair ($Y = 1$ if $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$ are considered similar, $Y = 0$ otherwise), P is the number of training pairs, D_W is a parametrized distance function with parameters W, and m is a margin defining the threshold beyond which the distance between dissimilar pairs is penalized.

Cross-modal retrieval

One key application of multimodal embedding models is cross-modal retrieval, where an input query of some modality is used to retrieve relevant results of some other modality (see section 2.4). For example, in image-text retrieval, a text query is used to search for relevant images. To optimize this process, a special type of database, called a vector database, is usually used to store the existing embeddings. These databases allow for fast retrieval using proximity-based search algorithms, even with large amounts of data. Notable examples include the Faiss library [35] and the Milvus database [91]. Since recently, some researchers

[2]Self-attention is a mechanism that allows a model to weigh the importance of different tokens (words) within a sequence relative to each other when making predictions.

are even making an effort to automatically refine user's text queries using LLMs for a more efficient and accurate search, as can be seen in [59].

The backbones of vector search systems are the search algorithms. As a note, all of the following techniques are implemented in both Faiss and Milvus. The simplest search algorithm is brute force search (normal linear search), which is slow but 100% accurate. Approximate nearest neighbor (ANN) is a family of algorithms that aim to find the closest vectors to the query vector with high efficiency, but without guaranteeing an exact match. They use various techniques and data structures to reduce the number of comparisons, and their examples include HNSW (Hiearchical Navigable Small Word) and LSW (Locality-Sensitive Hashing). Inverted File Index (IVF) is an ANN search method based on the idea of partitioning the vector space intro clusters and associating each vector with the nearest cluster. During search, only the most relevant clusters are examined. Overall, choosing a search algorithm (and tuning its parameters) comes down to a trade-off between speed and accuracy [89].

Let's introduce the most common **metrics** to measure the performance of a model performing cross-modal retrieval. All of these metrics, taken from [94, p. 23], assume that for each input query, the model retrieves the top-k most relevant results:

- **Precision@k** – The ratio of relevant items among the top-k retrieved items for each query, mean Precision@k is then the average over all queries. The formula for each query is:

$$Precision@k = \frac{TP}{TP + FP} \tag{3.4}$$

  where TP and FP are the counts of true positives (relevant items that were retrieved) and false positives (items that were retrieved but are irrelevant to the query) in the top-k retrieved results.

- **Recall@k** – The ratio of relevant items among the top-k retrieved items to the total number of relevant items for each query, mean Recall@k is then the average over all queries. The formula for each query is:

$$Recall@k = \frac{TP}{TP + FN} \tag{3.5}$$

  where FN is the count of false negatives (items relevant to the query, that were not retrieved) in the top-k retrieved results.

- **AP@k** – Average Precision at k accounts for the order of the top-k retrieved items. For each query, it involves calculating Precision@i among the first i retrieved items (k times), while gradually incrementing the value of i from 1 to k. Mean AP@k is then the average over all queries. The formula for each query is:

$$AP@k = \frac{\sum_{i=1}^{k}(P@i * rel(i))}{\sum_{i=1}^{k} rel(i)} \tag{3.6}$$

  where $rel(i)$ is an indicator function with value 1 if the i-th retrieved result is related to query q (else 0), and $P@i$ denotes the Precision@k metric from equation (3.4).

- **NDCG@k** – Normalized Discounted Cumulative Gain at k is the cumulative gain for each query, discounted by the logarithm of the rank. The gain is normalized by

the ideal gain for each query. The formula for each query is:

$$NDCG@k = \frac{DCG}{IDCG}$$
$$DCG = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}, IDCG = \sum_{i=1}^{|rel|_k} \frac{rel_i}{log_2(i+1)} \tag{3.7}$$

where rel_i is the correlation degree between the i-th retrieved result and the query sample, and $|rel|_k$ is an ideal set consisting of the top-k samples after sorting all the retrieved samples according to the real correlation.

3.2 Multimodal Large Language Models (MLLMs)

In recent years, the field of artificial intelligence has seen a remarkable progress with the invention of large language models, which have since demonstrated previously unseen capabilities in understanding, generating, and manipulating text. Ever since LLMs became widely available to the public through interfaces like OpenAI's ChatGPT[3], many people started using these models to assist them with text-based tasks in their daily lives (as noted in the declaration, GPT-4o [63] was even used to assist with writing some parts of this thesis [64]). However, many real-world problems don't involve just textual data, but a combination of different modalities. Multimodal LLMs (MLLMs) extend the capabilities of traditional LLMs to processing, understanding, and reasoning across different modalities, allowing them to perform complex tasks such as visual question answering, image and video captioning, or even being at the core of certain autonomous systems. This section begins with a general introduction to LLMs, and then shifts focus to multimodal extensions of their capabilities, especially in video understanding.

Large Language Models (LLMs)

An LLM is a type of generative AI model designed to process text and generate a coherent and contextually relevant output by learning patterns and relationships between words and concepts within large-scale text datasets. The development of LLMs required facing key challenges of the NLP field, such as understanding the semantics, syntax, pragmatics, and context in human language. Apart from a very fluent use of language, LLMs have demonstrated remarkable semantic understanding of concepts (at least to a certain degree [79]), which opened doors for them to be used in various applications in broad fields like medicine, education, and science. The two probably most important and influential LLMs, or rather families of models, are GPT (Generative Pre-trained Transformer) [72] developed by OpenAI and BERT (Bidirectional Encoder Representations from Transformers) [17] developed by Google.

The ability of LLMs to perform tasks with minimal (few-shot learning) or no (zero-shot learning) task-specific training is enabled by the facts that currently, these models usually contain hundreds of thousands up to hundreds of billions of trainable parameters, and are trained on tens of gigabytes up to several terabytes of text data [73]. These models usually operate by predicting the next token (a word, or a comma, space, etc.) in a sequence based on the provided context. Autoregressive models, like GPT, generate text token by token by always predicting the next token based on the existing sequence up to that point.

[3] https://chatgpt.com/

On the other hand, masked language modeling (used by BERT) is a method designed to understand context in both directions from the currently generated token. During training, some randomly chosen tokens are masked (hidden) by the model, and the model's goal is to always predict the masked token based on the surrounding context. There are also other types of LLMs, for example Sequence-to-Sequence models (e.g., BART [41]), whose goal is to transform one sequence into another sequence, making them the usual choice for translation. Comparison between the different pre-training approaches used by these three distinct LLM types is shown in figure 3.2. In general, LLMs are typically trained using a multi-stage process, which includes learning general language understanding from massive amounts of unlabeled data (self-supervised learning; this stage is called pre-training), adapting the model to specific tasks with labeled data through fine-tuning (supervised learning), and additional fine-tuning based on human feedback to align the model's behavior with human preferences (reinforcement learning) [29].

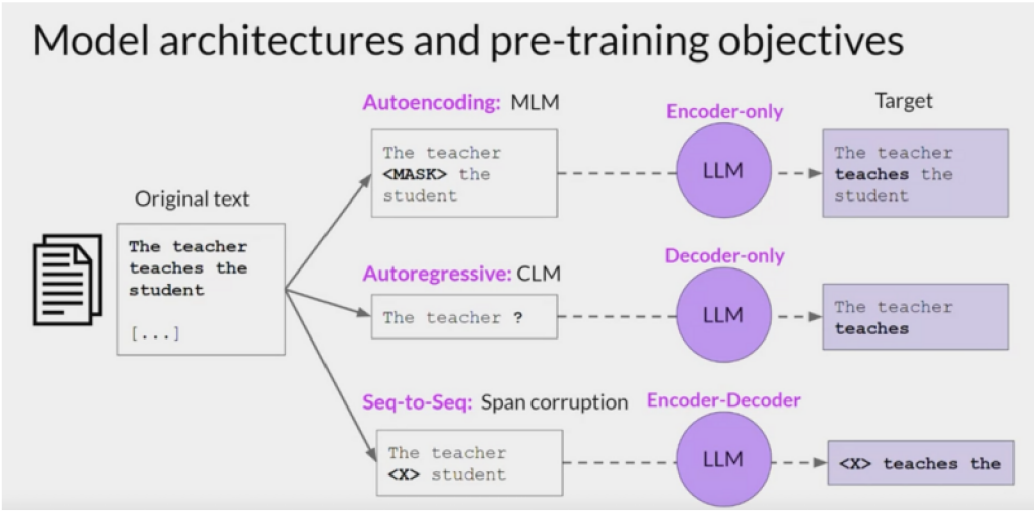

Figure 3.2: Comparison of pre-training approaches between three different LLM types—Autoencoding (Masked Language Modeling), Autoregressive (Casual Language Modeling), and Seq-to-Seq (Span corruption[4]) models [15].

Basically all modern LLMs are based on the **Transformer architecture** introduced by Google in the 2017 paper titled *Attention is All You Need* [90], which revolutionized the NLP field. Transformers have the ability to capture long-range dependencies (Google's Gemini 1.5 Pro has a context window of two million tokens)[5], they offer great scalability and can process tokens in parallel (unlike RNNs), and contextual embeddings allow them to have contextual understanding of words. The model architure of a transformer is shown in figure 3.3 and it consists mainly of an encoder and a decoder, both containing self-attention mechanisms (the key invention regarding transformers) and traditional feedforward neural network layers. A transformer's architecture generally consists of the following parts [6]:

[4] A pre-training strategy for Seq-to-Seq models, that involves masking out contiguous spans of tokens (rather than individual tokens) in the input sequence, and the model's task is to predict the missing spans based on the remaining context.

[5] https://developers.googleblog.com/en/new-features-for-the-gemini-api-and-google-ai-studio/

1. **Tokenization** – The whole text is split into tokens (words, commas, etc.). Tokens are usually integers representing their positions in a predefined dictionary.

2. **Embedding layer** – Tokens are converted to high-dimensional vectors capturing their semantic meanings.

3. **Positional encoding** – This is added to the embedding layer to help retain the order of tokens during their parallel processing.

4. **Self-attention mechanism** – Weights the importance of every input token relative to each other, dynamically adjusting their influence on the output. For each token, three vectors are calculated—query, key, and value. The dot-product of queries with keys determines the token's relevance. The results are normalized using the SoftMax activation function, and the attention weights are applied to the value vectors, producing context-aware token representations through a weighted sum.

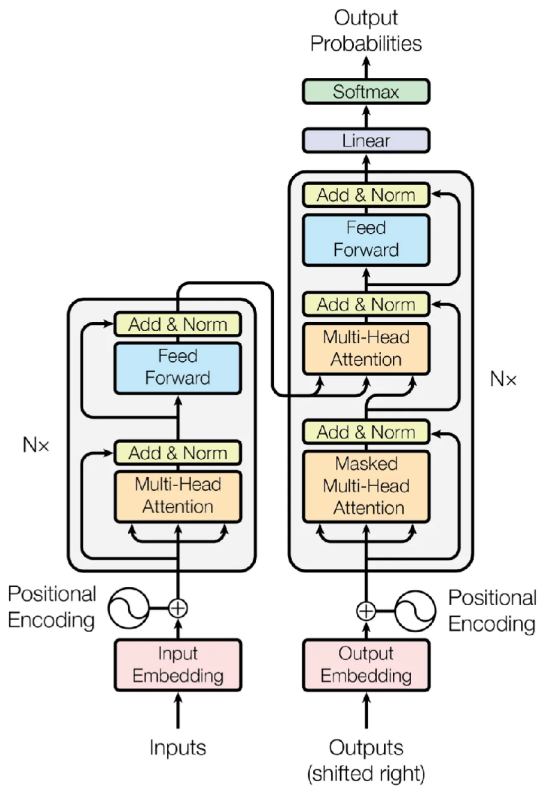

Figure 3.3: The original Transformer model architecture [90, fig. 1].

5. **Multi-head attention** – Multiple attention heads are used to capture different aspects of the input sequence in parallel, with each head performing self-attention independently, focusing on different patterns. Their outputs are then concatenated and linearly transformed, resulting in a better understanding of complex language structures.

6. **Feed-forward layers** – They typically process each token independently, and consist of two linear transformations, with the ReLU activation function applied between them to add non-linearity.

7. **Encoder** – Processes the input sequence, producing a context-rich representation. It involves multiple layers of multi-head attention and feed-forward layers.

8. **Decoder** – Generates the output sequence, by processing the encoder's output using an additional cross-attention mechanism, connecting sequences together.

9. **Output generation** – Each final token is generated by passing the decoder's output through a SoftMax layer, converting its vector representation into a probability distribution over the vocabulary. Usually, the vocabulary token with the highest probability is selected.

MLLMs and Large Multimodal Agents (LMAs)

Multimodal LLMs extend the capabilities of traditional LLMs by being able to process multiple distinct types of data (e.g., text, images, audio, video). This allows them to perform actions that are beyond the scope of text-only LLMs, such as reasoning across different modalities, vision-language tasks (e.g., image captioning, visual question answering, scene understanding), text-audio tasks (e.g., audio captioning, music and audio analysis), or even embodied tasks in robotics, such as navigating and interacting with physical environments [98]. Notable examples of MLLMs include LLaVA (Large Language-and-Vision Assistant) [50], an end-to-end trained model that connects a vision encoder with an LLM for general-purpose visual and language understanding, and OpenAI's GPT-4o [63] ('o' stands for "omni"), which combines text, vision, and audio capabilities into one integrated model.

The training of MLLMs usually consists of pre-training and instruction fine-tuning (this time involving multimodal data). However, instead of starting from scratch, a pre-trained, instruction-fine-tuned traditional LLM is typically used as the base model. The process of encoding different modalities was already explained in section 3.1, and multimodal embedding models can be used as part of the MLLM architecture. Generally, there are two main approaches to building MLLMs [74]:

- **Unified embedding decoder architecture approach** – Each modality is encoded into a unified embedding space, and a single decoder is used for generating outputs. Typically easier to implement, as it doesn't require modifications to the LLM architecture, which has to process just one unified type of tokens.

- **Cross-modality attention architecture approach** – Employs a cross-attention mechanism to integrate different modalities directly within attention layers. Processing of modalities gets integrated more deeply into the LLM architecture and can be more computationally efficient in some cases.

The information in this paragraph is based on a recent survey from 2024 [100]. LLM-based multimodal agents, often called **large multimodal agents (LMAs)**, are complex systems that combine the capabilities of MLLMs with action-based, goal-oriented decision-making and reasoning. They are designed to perceive their environment and make appropriate decisions based on observations to reach specific goals. They typically consist of four basic components—**perception** (processing of multimodal information), **planner** (responsible for deep reasoning about the current goal and compiling a corresponding plan), **action component** (translates plans into specific actions, such as use of tools or interactions with interfaces), and **memory** (for saving information and experiences for later use). An LLM or an MLLM usually functions as the planner while also interacting with other components. Some applications of LMAs include autonomous driving, robotics, and video understanding. Some various examples of specific LMA models are LLaVA-Interactive [8] (a research prototype of a large language-and-vision assistant) and GPT-Driver [57] (a motion planner for autonomous vehicles).

MLLMs and LMAs in video analysis

One application of MLLMs and LMAs is understanding and processing videos. Tasks in this domain include video captioning (generating a textual description of what is happening in a video), temporal sentence grounding (see section 2.3), video question answering (VQA;

answering user's questions about what's happening in a video), video summarization, action recognition (detecting and labelling actions occuring in a video), and many more. An exemplary model is Video-LLaVA [47], which extends the LLaVA model to process video inputs and perform video-specific tasks.

Since LLMs are inherently sequential models, LLM-based video analysis models typically don't require additional mechanisms for understanding temporal relationships, that were introduced in the second part of section 2.2 (temporal video processing). On the other hand, the image processing techniques from the first part of the same section become highly relevant, as many video-LMAs, such as OmAgent [112] (later introduced in section 4.3), can interact with tools, which usually include some object detection model, facial recognition model, etc.

A major challenge with video MLLMs and LMAs is effectively aligning linguistic data with video, which is a quite complicated modality. Given the complexity of videos, other major problems are scalability and high computation costs— sampling each frame of a long video, analyzing it (possibly also with tools like an object detector), and remembering the information would lead to very poor performance, and the LLM's context window's size probably wouldn't suffice. For these reasons, current models usually have to make significant compromises, and effectively understanding long-form videos remains a highly relevant challenge [95].

Chapter 4

Existing solutions and suitable models

This chapter explores existing solutions, by first introducing some end-to-end solutions of systems tailored for very similar problems, and then shifting focus to specific multimodal embedding models and MLLM-based models that were deemed suitable for the target system. Again, to understand the roles these two types of models will play in the system, it is highly recommended to get familiar with chapter 5, especially with section 5.1 first.

Section 4.1 introduces solutions of entire systems that are of similar nature to the target system, sections 4.2 and 4.3 then serve as a theoretical introduction to specific models that will be made available for use in the target system (and also some models that may not be used, but are still worth mentioning).

4.1 Whole-system solutions

This section introduces some existing systems that have a lot in common with what the target system should do. These similarities include searching for specific moments in a collection of multiple videos, or searching for objects (vehicles) and analyzing traffic footage.

Contextual Search Engine with LLM

On August 3 2023, a user with username "Jesus" published a blog post where he demonstrated how conveniently can LLMs and other related machine learning models be used to build a system that allows for very quick search of specific moments in a collection of videos [32]. Although a blog posts may be considered too informal to serve a serious purpose in an academic thesis, this post served as both an initial inspiration and an initial proof that the thesis' core idea is actually feasible. The system, heavily utilizing Google Cloud Platform services[1], was built for videos of sport matches (NBA, NFL, MLS), but that serves just as a specific example as the whole system is easily generalizable to other domains.

When a video gets uploaded, two main things happen—frames are sampled from the video, which then get converted to embeddings using a multimodal embedding model (sometimes incorrectly called an MLLM by the author) and stored in a vector database, and transcription of the video's audio is extracted, which is then both summarized and classified (based on sport type to optimize searching) by an LLM (these attributes are then stored

[1] https://cloud.google.com/docs

with the embeddings). To search for a specific event, an input text query gets converted by the same multimodal embedding model to the same embedding space, and the database is searched for k-closest matches. The described pipeline is shown in figure 4.1. Part of the system is also a simple web interface that shows a picture and a similarity score for each match, and the entire source code is public[2].

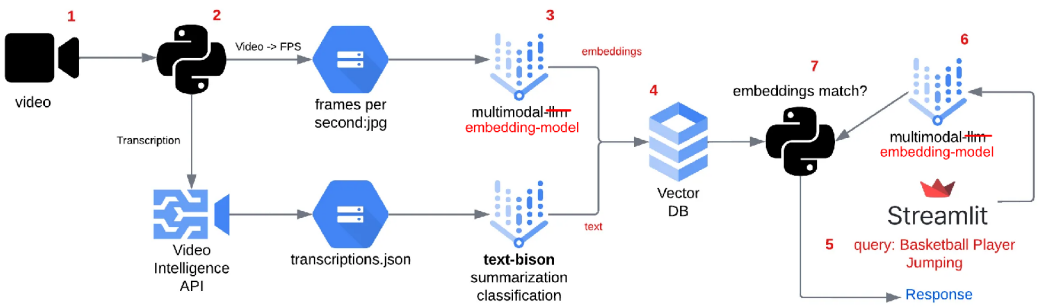

Figure 4.1: Pipeline of the *Contextual Search Engine with LLM* system (corrected, the original picture was taken from [32]).

VISE: Vehicle Image Search Engine with Traffic Camera

VISE (Vehicle Image Search Engine) was presented to support fast search of similar vehicles from low-resolution traffic camera images [12]. An image of a specific vehicle is used as an input, and the system retrieves visually similar vehicles from a vast database of traffic camera footage. VISE can be used to trace and locate vehicles for applications such as police investigations even when high-resolution footage is not available. As shown in figure 4.2, the system consists of three components: an interactive user interface for querying and browsing identified vehicles, a scalable search engine for fast similarity search on millions of visual objects, and an image processing pipeline that extracts feature vectors of objects from video frames.

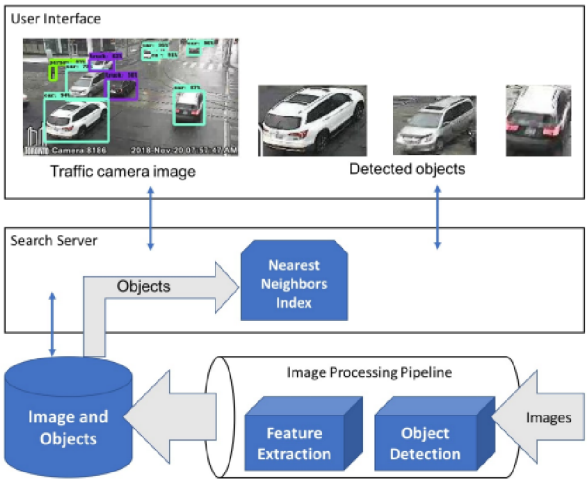

Figure 4.2: VISE's architecture [12].

[2] https://github.com/jchavezar/vertex-ai-samples/tree/main/gen_ai/video

On a more technical level, its core relies on a multi-stage process that involves object detection, feature extraction, and similarity-based retrieval. At first, vehicle objects get detected and extracted from traffic camera images using a Region-based Fully Convolutional Network (RFCN) [13] with a ResNet-101 backbone, which got adopted due to its high recall rate for object detection and high inference speeds. To represent each object based on its semantic features rather than raw pixels, ResNet-50 [27] (which is another CNN model) is used to extract its features as a fixed-sized vector.

For fast and scalable similarity searches, VISE uses Hiearchical Navigable Small World (HNSW) [56], which is a K-nearest neighbor search index, to efficiently find objects that are highly-likely the same object as the input query based on cosine distance between the corresponding feature vectors. Since the used CNN is pre-trained on general images, rather than domain-specific data, additional refinement techniques are incorporated—a custom classifier trained on labeled pairs is used for removing false positives, and a technique called multi-object query is used for getting more results and avoiding false negatives (the user inputs multiple images of the same vehicle grouped as one query, and the final search result is a union of the individual search results, which improves recall).

TrafficVLM: A Controllable Visual Language Model for Traffic Video Captioning

TrafficVLM is a multimodal dense video captioning model specifically designed for traffic surveillance and urban safety applications [18]. Unlike retrieval-based systems that store and index data for later searching, this is a video captioning model which can precisely localize and describe incidents directly within a continuous video stream at different levels of analysis, both spatially and temporally (even though it's technically a model rather than a system, it's included in this section as it's not an interactive chatbot like models in section 4.3). This model can not only detect traffic events, but also generate fine-grained descriptions of vehicle and pedestrian interactions at different phases of an event. TrafficVLM also introduces a conditional component to control the generation outputs, and a multi-task fine-tuning paradigm to enhance its learning capabilities.

Video input (the model is designed for vehicle and overhead camera views) is processed using a two-stage feature extraction method. First, sub-global visual features are extracted by cropping and resizing video frames to a standardized resolution, and encoding them using the CLIP model [71] (specifically version ViT-L/14). These features are further refined by trimming irrelevant frames and applying temporal subsampling. In the second stage (local feature extraction), bounding boxes around pedestrians are used to crop and encode localized features. If bounding box information is missing for certain frames, a learnable tensor is used to fill in missing embeddings, to ensure robust representation across all phases of an event.

For temporal modeling, TrafficVLM utilizes a vision transformer encoder to capture sequential dependencies between frames. The encoded video features are then concatenated with conditional embeddings, allowing the model to differentiate between vehicle and pedestrian descriptions. The system uses T5-Base (Text-To-Text Transfer Transformer developed by Google [73]) as its text decoder, which can autoregressively generate captions describing traffic events (object behavior, location, interactions, etc.). The decoder is trained with an extended tokenizer that includes additional time tokens, which enables it to predict event boundaries and textual descriptions in a single sequence.

Additionally, a multi-task fine-tuning approach that separates vehicle and pedestrian captioning as individual training objectives is used to improve training efficiency (the training loss is computed as a weighted combination of losses based on the two separate tasks). For fine-tuning and evaluation, the authors used the WTS dataset [38], which contains a large amount of videos from diverse traffic scenarios. The TrafficVLM model has demonstrated strong performance by ranking third on the blind test set of the AI City Challenge 2024 Track 2 [93] (an annual competition focused on applying AI models to traffic-related problems).

4.2 Suitable multimodal embedding models

This section introduces (image-text) multimodal embedding models that were evaluated as suitable for the target system (most of which will be available for use in the system, and later benchmarked in section 7.1). The reasons for why these exact models were chosen (which include their applicability to the problem domain, computing costs, hardware limitations, etc.) will be explained later in section 6.2. Since most models of this nature are not exclusively limited to just generating embeddings from images and text, models that are limited in their additional capabilities (limited to text-image similarity measurement, classification, etc.) were strongly preferred to make the system somewhat lightweight (as opposed to, for example, loading or dissecting an entire MLLM just to use some of its components to generate embeddings). Please note that this doesn't imply any performance limitations, as many of the selected models (especially CLIP) are often used as the de facto standard component for generating embeddings even in large, complex multi-modal models or systems.

CLIP

Contrastive Language-Image Pre-traning (CLIP), introduced by OpenAI in 2021, is an efficient method of learning from natural language supervision [71]. Instead of being trained to predict a fixed set of predetermined object categories, the CLIP model was trained on approximately 400 million image-text pairs (each image has a corresponding natural language description[3]) by using a contrastive representation learning approach, learning to pull corresponding pairs closer together and unrelated pairs further apart in a unified embedding space. The authors found that the model can be generally used in various contexts it was not directly trained for, which is supported by the fact that it achieved remarkable zero-shot performance on various benchmarks (such as Imagenet [16], matching the performance of the original ResNet-50 [27] that was specifically trained on the data [71, fig. 5]).

The model has two main components, which were trained from scratch—a transformer-based text encoder for embedding text, and an image encoder for embedding images (implemented either by the original Vision Transformer (ViT) [19] in the better performing CLIP versions, or by ResNet-50). If CLIP was trained using a standard contrastive learning approach, it would probably be shown triplets ("anchor", "positive", "negative") where, for example, "anchor" could be an image of a dog, "positive" could be string `"Image of a dog"`, "negative" could be string `"Image of a plane"`, and the model would learn to

[3]These descriptions were not manually crafted but were instead sourced from the internet, capturing text that naturally accompanies images, such as captions, alt text, or surrounding metadata.

minimize the distance between "anchor" and "positive" while maximizing the distance between "anchor" and "negative". In reality, CLIP optimizes and generalizes this approach by using a multi-class N-pair loss. Given a batch of N image-text pairs, an $(N \times N)$ similarity matrix is constructed, where each row corresponds to an image embedding and each column corresponds to a text embedding, as visualized in figure 4.3 (1). The model then learns to maximize the cosine similarity of the N corresponding pairs (found on the diagonal), while minimizing the cosine similarity of the embeddings of the remaining $(N^2 - N)$ incorrect pairings.

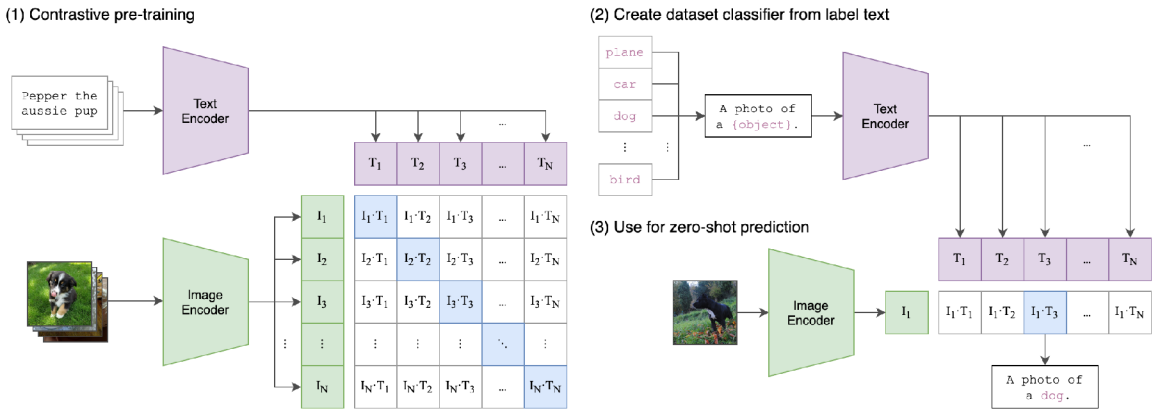

Figure 4.3: Architecture overview of CLIP [71, fig. 1].

It is also worth mentioning that since its original release in 2021, countless new variations or extensions of CLIP have been introduced. For example, OpenCLIP [11] is an open-source reproduction of CLIP, often used to create CLIP-like models fine-tuned for specific domains, and X-CLIP [55] is a minimal extension of CLIP for video, created for video-text retrieval.

ALIGN

ALIGN (**A** **L**arge-scale **I**ma**G**e and **N**oisy-text embedding) is a vision-language model introduced by Google Research (shortly after OpenAI introduced CLIP) in 2021 [33]. It's very similar to CLIP, but differs in some key ways primarily related to the training data. This model has been trained on ≈ 1.8 billion (noisier and weakly filtered) image-text pairs massively scraped from the internet, as opposed to CLIP, whose 4.5 times smaller dataset was more carefully curated. The two models, which share the same contrastive learning approach, achieve comparable performance on most benchmarks [33, tab. 1, 4, 5], with some scenarios favoring ALIGN's quantity-focused approach, and some scenarios favoring CLIP's quality-focused approach. ALIGN also shares the same dual-encoder architecture, but uses EfficientNet [86] as the image encoder, and BERT [17] as the text encoder, both of which are Google's own models (and again, both encoders were trained from scratch).

SigLIP

SigLIP (Sigmoid Loss for Language-Image Pretraining) was introduced by Google Research in 2023 as an improvement over CLIP [107]. It is based on CLIP, but replaces the original loss function by a simple pairwise sigmoid loss. Unlike standard contrastive learning with softmax normalization, the sigmoid loss operates solely on image-text pairs and does not require a global view of the pairwise similarities for normalization, which leads to more

stable training dynamics not dependent on batch composition and size, as the image-text pairs are processed independently without the need for negative (incorrect) pairs.

Even though SigLIP outperforms CLIP on multiple benchmarks including ImageNet-1k [107, tab. 3], the lack of contrastive hard negatives in training may lead to slightly weaker discrimination between similar classes. Since this thesis focuses on traffic footage, and concepts like "hatchback" and "sedan" could probably be considered very similar given that both of these models are general-purpose (domain independent), the comparative benchmarks of these models in this domain (see section 7.1) may yield interesting and unpredictable results.

BLIP

BLIP (Bootstrapped Language-Image Pre-training) is a vision-language model introduced by Salesforce Research in 2022 [44]. Unlike the other models previously introduced in this section, BLIP is much more than just an embedding model, as it can perform various multi-modal tasks including visual question answering (VQA), image-text retrieval (and image-text matching), and image captioning. Its architecture, referred to as Multimodal mixture of Encoder-Decoder (MED), allows it to function both as a unimodal encoder (like the previously introduced models), and as a generative model for text generation. It consists of a vision transformer for image encoding, and a BERT-based text encoder/decoder. Unlike the previous models that rely on large (and often noisy) web datasets, BLIP introduces a Captioning and Filtering (CapFilt) mechanism, where a synthetic caption generator creates additional text descriptions, and a filter is used to remove noisy text to improve training data quality. BLIP was pre-trained using three objectives: Image-Text Contrastive loss (ITC) for representation learning, Image-Text Matching loss (ITM) for fine-grained alignment, and Language Modeling loss (LM) for text generation.

The only component of BLIP that will be used in the target system is its feature extractor (for both images and text), which by itself still follows a dual-encoder architecture. While the ITC loss is similar to that of the previous models, BLIP's main differences (apart from the used encoders) stem from the use of ITM loss, which goes beyond contrastive similarity by learning whether a matched pair is truly related (in the form of a binary classification task), and from the utilized CapFilt mechanism. BLIP managed to outperform both CLIP and ALIGN on certain image-text retrieval benchmarks [44, tab. 5, 6], while requiring much less training data.

Cloud API services

As an alternative to running multimodal embedding models locally, many large tech companies provide cloud API services offering remote access to their latest state-of-the art models (for reference, Google's cloud services were utilized by the first solution listed in section 4.1). These services include Google Cloud Vertex AI[4], Microsoft Azure AI[5], Amazon Bedrock[6], and others. In most cases, these are closed-source models, such as Amazon Titan Multimodal Embeddings model G1[6], and these API services are the only way to use them.

[4] https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings

[5] https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/how-to/image-retrieval

[6] https://docs.aws.amazon.com/bedrock/latest/userguide/titan-multiemb-models

4.3 Suitable MLLM-based models

This section introduces MLLM-based[7] models capable of video understanding and user interaction, which were evaluated as suitable for the target system (and again, most of which will actually be available for use there) based on their applicability to the problem domain (apart from other factors, like hardware limitations). Rather than models specialized for long video comprehension (excelling in narrative-driven content like movies), slightly more generic models with strong image-centric capabilities (object recognition, OCR for reading license plates, etc.) were preferred given the traffic-oriented use case. In the same manner as the multimodal embedding models, the reasons for why these models were chosen will be explained later in section 6.2, and the comparative benchmarks of the selected models in context of the problem domain will be described and evaluated later in section 7.2.

LLaVA-OneVision

LLaVA-OneVision is a family of MLLMs specifically designed to excel across multiple computer vision tasks, including single-image, multi-image, and video-based scenarios [42]. Introduced in August 2024, it is a recent addition to the series of LLaVA open-source models [50], and is built upon techniques previously used to develop LLaVA-NeXT [49]. LLaVA-OneVision leverages a combination of high-quality, large-scale datasets, and a carefully curated training process. Unlike previous models, it demonstrates strong cross-modal and cross-task transfer capabilities thanks to its modality-mixing training approach, allowing it to generalize effectively from static images to complex video analysis tasks. Its architecture[8] (see figure 4.4) connects an LLM (the authors chose Qwen-2 [103]) with a vision encoder (for which they chose the already introduced SigLIP [107]) through a 2-layer Multilayer Perceptron (MLP), which projects image features into the word embedding space. This setup allows the model to process and understand a variety of visual inputs, while maintaining robust language-based reasoning.

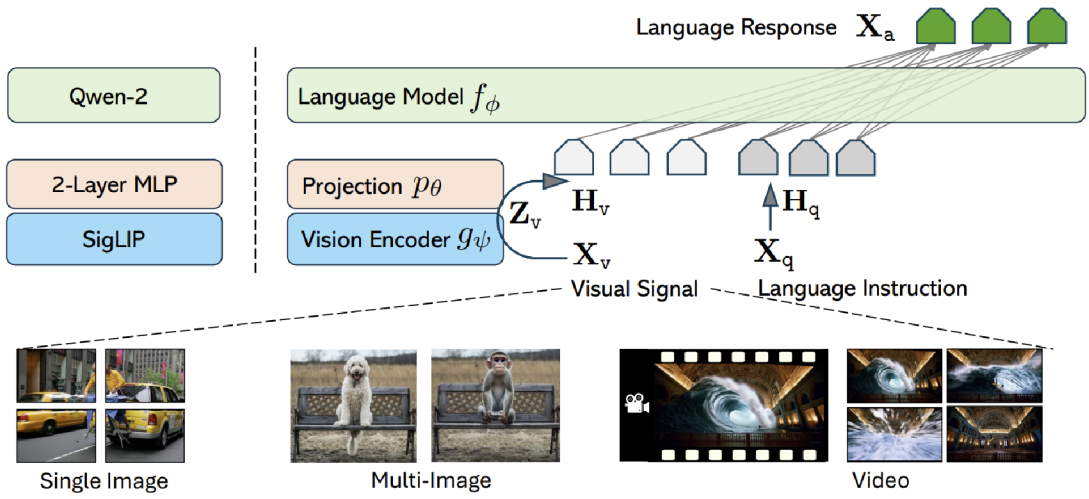

Figure 4.4: LLaVA-OneVision network architecture. The general component overview is on the right; the specific models chosen for the components are shown on the left [42, fig. 1].

[7]The term *MLLM-based models* is used to not only include MLLMs, but also LMAs and similar models.
[8]This is a common design choice in MLLMs, also adopted by VideoLLaMA 3 and Qwen2.5-VL.

LLaVA-OneVision also uses and further improves AnyRes, a flexible visual input representation strategy that allows the processing of images of various resolutions efficiently. Instead of requiring fixed-size inputs, the model can dynamically adjust the image resolution and the number of visual tokens depending on the input scenario. LLaVA-OneVision further introduces bilinear interpolation and refined token allocation strategy to this technique, which has a positive impact on video understanding as more frames can be processed efficiently. On certain video benchmarks, even its compact 7-billion parameter version achieved performance that is on par with some much larger, often proprietary models [42, tab. 5].

GPT-4o

GPT-4o („omni“) is a closed-source proprietary MLLM released by OpenAI in May 2024, which can both process and generate (except for video) text, audio, image, and video within a single end-to-end neural network [63]. Unlike previous models, like GPT-4V [65], that handled different modalities separately, GPT-4o integrates them seamlessly, leading to faster and more efficient cross-modal reasoning. The model matches GPT-4 Turbo[9] performance on English text-based tasks (and significantly improves multilingual performance), while also being significantly faster and 50% cheaper to run in the API[10], but its main improvements stem from the outstanding vision and audio understanding.

When it comes to video understanding, GPT-4o consistently ranks among the top-performing models in most benchmarks [42, tab. 5]. In VideoVista benchmark, which comprises 25,000 questions derived from 3,400 videos spanning 14 categories, it dominated in scenarios requiring both understanding and reasoning [46, tab. 2, 4]. It showcased superior performance in fine-grained video tasks, including temporal localization and object tracking, as well as in logical and causal reasoning tasks.

VideoLLaMA 3

Released in January 2025 by researchers at DAMO Academy (Alibaba Group), VideoLLaMA 3 [108] is an advanced multimodal foundation model for image and video understanding, offering significant improvements over the two previous VideoLLaMA [109] versions. Its core design philosophy is vision-centric, which influences both its training paradigm and framework design. For training, the authors favored high-quality image-text data over large-scale video-text datasets, whose quality is often lacking. The training process consists of four stages—Vision Encoder Adaptation (to accept images of variable resolutions as input), Vision-Language Alignment, Multi-task Fine-tuning, and Video-centric Fine-tuning. The authors fine-tuned SigLIP [107] for the vision encoder, and utilized Qwen2.5 [69] models for the LLM. By exposing VideoLLaMA3 to a variety of downstream tasks, including interactive question answering and video captioning, the model developed a comprehensive understanding of both static and dynamic visual information.

Furthermore, the model uses a differential frame pruning mechanism and bilinear interpolation to optimize video processing (quite similar to what LLaVA-OneVision [42] used, along with the variable resolution mechanism and training focused on high-quality data and combining modalities). On the evaluated video understanding benchmarks (performed separately with 2 and 7-billion parameter models), the model outperformed its competi-

[9] https://help.openai.com/en/articles/8555510-gpt-4-turbo-in-the-openai-api
[10] https://platform.openai.com/docs/api-reference

tors (including Qwen2-VL [92]) in all three areas (general video understanding, long video understanding, temporal reasoning) [108, tab. 7, 8].

Qwen2.5-VL

Released in February 2025 by Alibaba Group's Qwen Team, Qwen2.5-VL [4], the latest flagship model of the Qwen vision-language series [3], is the most recent model introduced in this section. This vision-language model achieves a major leap forward through its enhanced video recognition, precise object localization, and long-video comprehension. Similarly to LLaVA-OneVision [42] and VideoLLaMA 3 [108], one of its key innovations is dynamic resolution processing, but it also introduces dynamic frame rate and absolute time encoding strategies, allowing it to efficiently understand very long videos. For the architecture, the vision encoder employs a redesigned Vision Transformer architecture [19] trained from scratch, incorporating window attention to optimize inference efficienty, and Qwen2.5 [69] is used for the LLM. One feature of this model that may prove very beneficial in traffic footage understanding is its fine-grained spatial understanding system, giving it the ability to localize objects using bounding boxes or points accurately.

Qwen2.5-VL was trained on an extensive dataset consisting of 4.1 trillion tokens, covering a wide range of multimodal tasks (VQA, object grounding, multimodal mathematics, agent-based tasks, video understanding, etc.). The flagship 72-billion parameter model achieves state-of-the-art performance on multiple benchmarks, rivaling GPT-4o [63] and Gemini 1.5 Pro [22] on (not only) video benchmarks [4, tab. 8], and even significantly outperforming them on grounding and OCR benchmarks [4, tab. 5, 6] (in these cases, even the smaller 3B and 7B versions still outperform these flagship models). With its combination of strong vision-language alignment, efficient tokenization, and broad task generalization, Qwen2.5-VL establishes itself as a leading open-source MLLM as of February 2025.

OmAgent

The original OmAgent [112], implemented as OmAgent version 0.1.0[11], was a framework for creating LMAs capable of complex video understanding, combining retrieval-augmented generation (RAG) with an autonomous AI agent. It consisted of two main components: a Video2RAG video preprocessor for extracting and storing generalized information from a given video for efficient retrieval, and a Divide-and-Conquer (DnC) loop for task planning and execution (an autonomous reasoning system capable of recursive task decomposition, see figure 4.5), which was also equipped with external tool invocation capabilities (web search, invoking an object detection or face recognition model, etc.). Unlike traditional methods at the time, relying on static frame extraction or full-text conversion, OmAgent dynamically retrieved relevant video segments based on queries and refined its reasoning through an iterative problem-solving process. Additionally, it integrated a "rewinder" tool for revisiting specific parts of a video. Based on the provided comparisons, OmAgent outperformed multiple similar models in varied disciplines [112, tab. 1, 2, 3].

[11] https://github.com/om-ai-lab/OmAgent/releases/tag/v0.1.0

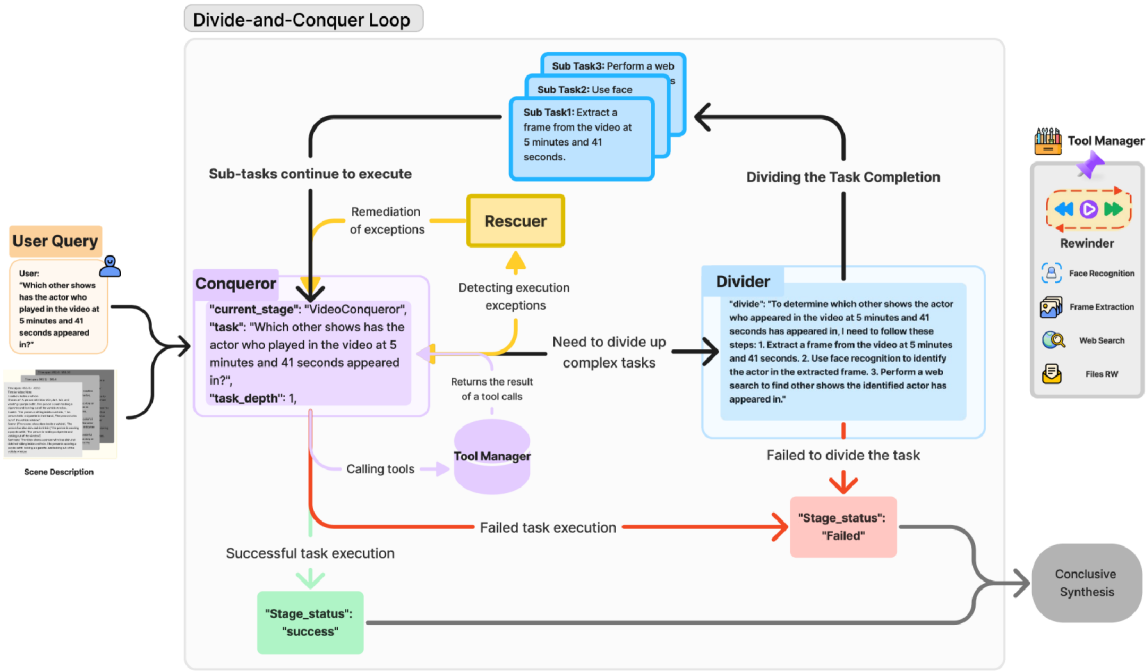

Figure 4.5: The original OmAgent's Divide-and-Conquer (DnC) loop [112, fig. 2].

With the release of version 0.2.0[12] in October 2024, OmAgent drastically changed by becoming an open-source agent framework designed for streamlining the development of on-device multimodal agents. The authors' new goal is to enable agents that can empower various hardware devices like smartphones, smart glasses, or even futuristic robots. Its design architecture now adheres to three fundamental principles: graph-based workflow orchestration, native multimodality, and device-centricity. As of February 2025, no updated research paper has been published, making OmAgent's official website[13] and Github repository[14] the primary sources of (unfortunately limited) information.

[12] https://github.com/om-ai-lab/OmAgent/releases/tag/v0.2.0
[13] https://www.om-agent.com/
[14] https://github.com/om-ai-lab/OmAgent

Chapter 5

Concept of the system

This chapter presents the concept of the system for searching and analyzing traffic footage, which is required to utilize LLMs in some capacity, and whose implementation will be described in chapter 6. This chapter begins by describing the thought processes behind the key idea that set the course for the entire work itself, which is then followed by a bit more detailed concept of the system's architecture.

Section 5.1, which is recommended to be read before chapters 3 and 4, explores multiple ideas about how the system could work that were considered (and justifies the final choice), and explains the roles of multimodal embedding models and MLLMs in the system. Section 5.2 then presents a more detailed concept of the system's architecture, which includes things like the processing and flow of data, and the user interface.

5.1 The main idea

The work started by exploring existing solutions along with current state-of-the-art technologies regarding video search and analysis using LLMs and related technologies. At first, the *Contextual Search Engine with LLM* system (see section 4.1) inspired this work by demonstrating how effective, efficient and scalable multimodal embedding models can be for creating an image search system, despite the fact that these models are much less complex than LLM-based models. On the other hand, the work assignment specifically required the utilization of LLMs, which lead to experimenting with various modern MLLM-based models for video understanding (some Github repositories[1] provide useful surveys of such models [88]).

At some point, it became clear that an optimal approach should utilize both of these types of models. A multimodal embedding model, combined with a vector database, can be used to build a very efficient search system capable of working with tons of data, but it lacks the ability to understand temporal relationships, and to further interact with the user. In constrast, video-focused MLLM-based models have the ability to communicate with the user in natural language, describe complex events, and understand temporal relationships in videos, but are orders of magnitude more computationally expensive, meaning that the amount of input data they can effectively process is limited. Therefore, the challenge in constructing an effective search-and-analysis pipeline lay in finding an optimal split in utilizing a multimodal embedding model for navigating in a large collection of video data, and utilizing an MLLM-based model for more detail-focused tasks and natural language

[1] https://github.com/yunlong10/Awesome-LLMs-for-Video-Understanding

user interaction. When also taking extremes into account, the possible approaches can be categorized into three categories, which differ in the amount of work delegated to the MLLM-based model (in decreasing order):

Option 1 – MLLM-based model for video analysis handles everything

When considering models that expect a video and some text on input, all videos uploaded to the system would be concatenated into a single video file (which also introduces a problem of combining videos with various resolutions, framerates, and other attributes). This approach wasn't even seriously considered, since these models can only sample a limited amount of frames at once, which inevitably forbids any reasonable scalability, and the inference also takes way too much time for it to be used as the search function (through TSGV).

Option 2 – Video-text embedding model handles video selection, MLLM-based model handles single-video-wise actions

This approach is much more reasonable and was seriously considered while experimenting with X-CLIP [55]. When a user enters a text query, the query embedding would be matched with embeddings that each correspond to a single uploaded video. Once the best matching video is found, an MLLM-based model would first have to perform a temporal sentence grounding (TSGV) task to find the specified moment inside the given video, and then further analyze the located moment. Very similar concept was implemented by VideoRAG (January 2025) [31], which utilizes video-text embeddings for selecting relevant videos, which are then used as input for a video-MLLM.

Obvious limitations of this approach include the compression of information introduced by generating embeddings for entire videos (especially in traffic videos, where important events occur rarely), and the time and resources required for comprehending an entire video by an MLLM. When experimenting with the original OmAgent [112] in October 2024 on a hour long traffic video, about 20 cents worth of OpenAI credits had to be spent just to fullfill a single request. These models usually uniformly sample and process a fixed amount of frames, which is not only costly, but mostly wasteful in traffic videos where nothing interesting happens most of the time. In defense of this approach, some recent models can sample even thousands of frames at once [22], and utilize mechanisms for a more curated frame selection [87]. Furthermore, the stored videos could be split into videos of a limited maximum length, to decrease the amount of work delegated to the MLLM, and to decrease the amount of information that has to be captured by a single embedding vector.

Option 3 – Image-text embedding model handles search, MLLM-based model handles analysis of a specified video segment

This idea seemed like the obvious optimal compromise. Although modern developments in the field of video-MLLMs seem to aim to make the second option properly reasonable, it will be very hard to reach the speed and cost-effectiveness of this approach. When a user uploads a video, it gets uniformly sampled at some reasonable framerate (let's say, 1 FPS), and each frame (image) gets converted to an embedding, which gets stored in a vector database along with the corresponding video name and timestamp. When searching for a specific event, since each result comes with the corresponding information, only the corresponding segment (let's say, from 5 seconds before the timetstamp to 5 seconds after the timestamp)

of the specified video has to be input to the MLLM-based model for a detailed analysis and chatting with the user.

Experimenting with this approach showed that it can be used to build a very fluid and responsive system, that runs entirely locally on a modern mid-range computer. The obvious limitation of this approach stems from the limited ability to use temporal information for search, as the search process itself is reduced to multimodal retrieval of independent images. When experimenting with the supplied footage from Božetěchova and various multimodal embedding models, queries like "stealing" or "theft" often resulted in retrieval of frames showing a person opening the doors of their own car, or a person handling a package.

5.2 System architecture concept

Now that the core idea of how the system should work is established, let's go through more details that should be considered before the system is implemented. It was established that the system will be centered around a two-step pipeline (see figure 5.1), where a user first searches for a specific moment in a collection of videos (multimodal embedding model), and secondly, once the desired moment is found, chats with the "analysis chatbot" about what's happening in the corresponding video segment (MLLM-based model). Since the roles of the two types of models are exactly defined, there should be an option to configure a different model of the same type as a drop-in replacement (in both cases).

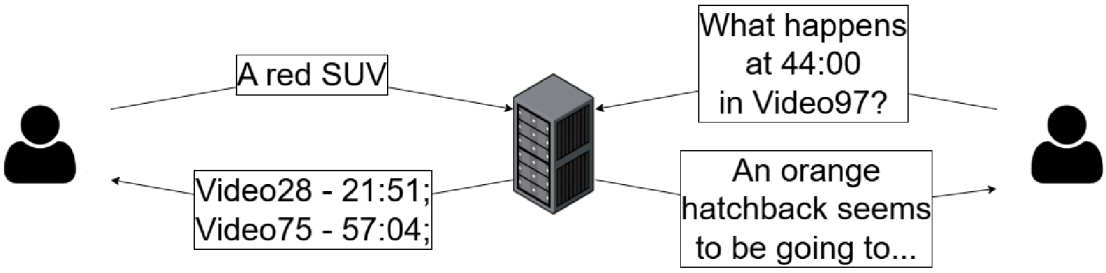

Figure 5.1: A diagram demonstrating the two-step pipeline through user interaction.

The system will require some application logic, that handles the user's collection of videos and proper interaction with the AI models. Based on the thesis' assignment, there should also be a web based user interface. Apart from the web application itself, the system will require some additional services—namely a vector database, that will store the embeddings and handle the embedding search. To interpret the search results and to provide the data for analysis to the MLLM-based model, the video data also have to be stored somewhere. The most convenient way would be to directly store entire video files, from which the requested frames would get sampled on demand, but if that turns out to be too computationally expensive, an alternative would be to always sample a video only once, and reuse the sampled frames both for interpreting search results and for analysis. Since most vector databases are optimized for storing and searching high-dimensional vector embeddings, and don't support storing video or image files, a cloud storage, a self hosted object storage (like MinIO[2]), or the local filesystem will additionally have to be used.

[2] https://min.io/

Application logic

When a user uploads a video, its frames have to be uniformly sampled at a specified sampling rate (one frame per second should probably suffice for traffic footage), then each converted to an embedding and stored in the vector database. The video is either uploaded to the storage before this preprocessing step, or the sampled frames get stored as images in between the sampling and the embedding generation. Each frame embedding gets stored along with the corresponding video name and timestamp (and possibly a reference to the corresponding image file). The user should obviously also be able to delete existing videos and the corresponding data, but since the record of a specific video refers to both its frame embeddings (and corresponding metadata) stored in the vector database and the file(s) stored in the file storage, the user should have access to some data management mechanism that allows him to keep track of all existing data corresponding to the individual videos, and to synchronize the states of the two separate services.

To search for a specific moment, the user enters a text query, which gets converted to an embedding using the same multimodal embedding model that generated the frame embeddings of the stored videos. The similarity search itself will be handled by the vector database, and cosine similarity will be probably used as the metric, since it's the most common similarity measure for real-valued vectors used in information retrieval, most vector databases support it efficiently via ANN search, and it's often the metric that gets optimized during training of models like CLIP [71]. It is also expected that not just one, but top-k results (for a user specified k), ranked by relevance, will be retrieved.

Once the user wants to analyze a specific video segment (which he located either through the embedding search, or manually), he should be able to specify (both automatically and manually) this interval by its start and end timestamps, the video name, and possibly the number of frames to be uniformly sampled from it for constructing the input for the MLLM-based model (unless the storage contains images instead of videos, and also, some of these models accept a video file on input and sample it themselves using an in-built preprocessor). The application doesn't have to support storing analysis conversations (as persistable sessions), but once the MLLM-based model is prompted, there should definitely be an option to continue the conversation.

User interface

The web based user interface can be simple and minimalistic, as it should primarily be easy to navigate. At least three separate views will be required—a search view for the embedding search, an analysis view for analyzing a video in the form of a conversation with an MLLM-based model, and some kind of data management view that let's the user manage the video data currently present in the system. The view for managing video data should display a structured list of uploaded videos with information about their corresponding data, and allow the user to perform certain actions on them, including uploading a new video.

The search view should contain a text input for inputting the search query, and an additional number input for choosing the number of retrieved results. Each search result should be displayed as a picture, along with some basic information (video name, timestamp, similarity), and a button that redirects the user to the analysis view with already predefined analysis parameters corresponding to the video segment centered around the timestamp of the retrieved frame. A simple mock of the search view is shown in figure 5.2.

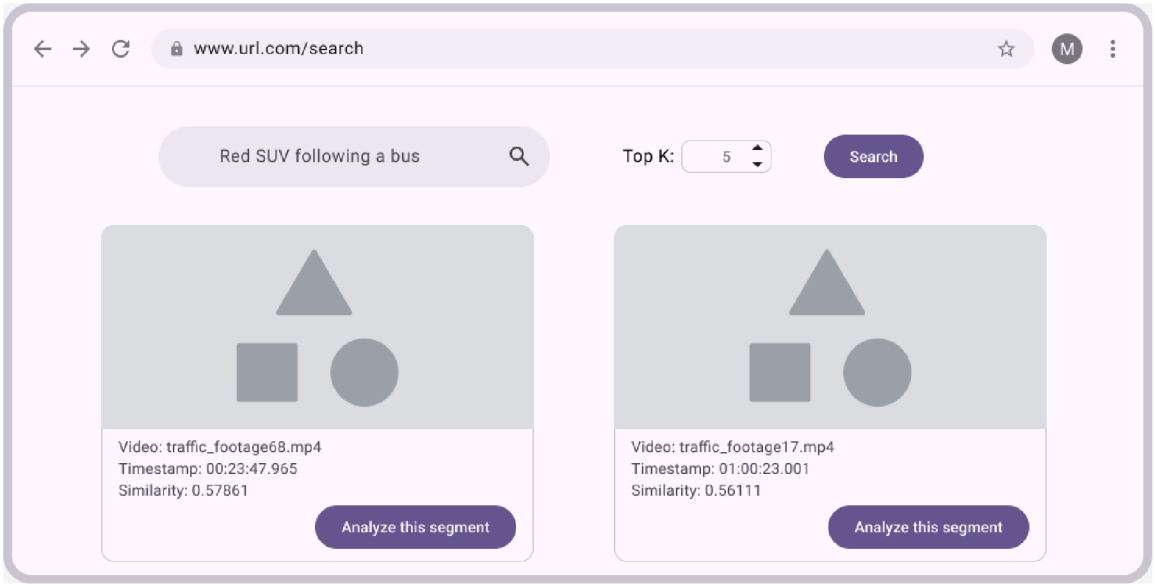

Figure 5.2: A simple mock of the search view created in Figma[3].

The analysis view should let the user specify the desired video segment (video name, start and end timestamps, sampling rate) to set up a new analysis. Once the analysis session is configured, the user should have access to a video player (which should automatically start at the specified start timestamp), and an interactive chat window for conversing with the MLLM-based model. A simple mock of the analysis view (during an active conversation and without the input fields for configuring a new analysis) is shown in figure 5.3.

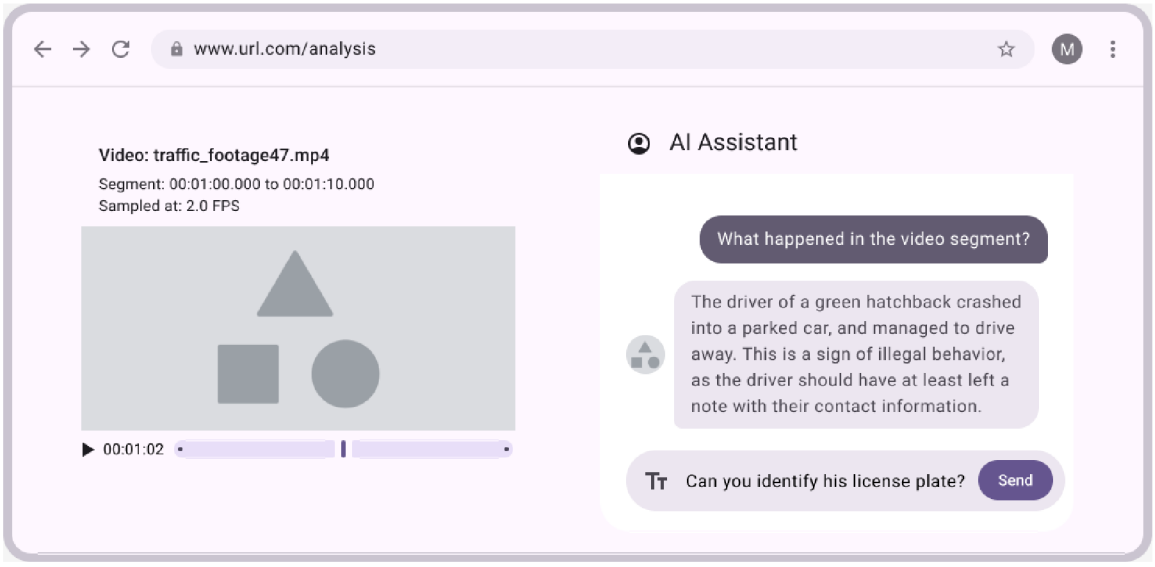

Figure 5.3: A simple mock of the analysis view (during an active analysis session) created in Figma[3].

[3] http://www.figma.com/

Chapter 6

Implementation of the system

This chapter describes the implementation of the system for searching and analyzing footage of traffic. It begins with an overview of its separate components (the web application and its additional services), followed by an overview of the available AI models, and a more detailed description of the application's core parts (both the backend and the frontend).

Section 6.1 describes the system's architecture in terms of its components and how they communicate with each other. Section 6.2 aims to explain why and how the available AI models, which were introduced in sections 4.2 and 4.3, were selected, specifies the exact model versions used, and mentions alternatives that were also considered. Sections 6.3 and 6.4 focus on the web application's backend and frontend, mainly by describing the available API endpoints, interesting or important implementation details, and the three separate graphical views.

6.1 Overview of system components

The system consists of a web application, installed and managed using Poetry[1], along with additional services running in containers managed by Docker Compose[2] (everything is located in the `/app` directory, which serves as the root directory of the Poetry project). The backend, which also integrates the AI models, is written in Python and uses FastAPI[3], a minimalistic yet powerful framework for building web applications using Python. FastAPI is often paired with Poetry, which automates the creation and management of virtual environments and simplifies dependency management. The frontend, written using Vue.js[4], is in the form of a single-page application (SPA), and is served directly by the backend as a "sub-application" using FastAPI's `StaticFiles`.

Milvus[5] was chosen as the vector database solution because it's open-source and optimized for high-performace vector similarity search, offers good scalability, and supports advanced indexing methods. The official `milvus-standalone-docker-compose.yml` Docker Compose file, allowing Milvus to run as a single instance as opposed to a distributed setup, was downloaded and saved as `/app/docker-compose.yml`. It manages three containers—`milvus-standalone` (the core service), `milvus-minio` (MinIO[6] object storage used by Mil-

[1] https://python-poetry.org/
[2] https://docs.docker.com/compose/
[3] https://fastapi.tiangolo.com/
[4] https://vuejs.org/
[5] https://milvus.io/
[6] https://min.io/

vus to persist large-scale files), and `milvus-etcd` (used to store Milvus metadata and service discovery). No additional service is required for storing uploaded video files[7], as the `milvus-minio` container was repurposed by creating an additional bucket[8] specifically for this purpose. Figure 6.1 illustrates a simplification of how the system components interact during three key scenarios—video upload, embedding search, and video analysis (although database and storage are usually considered part of an application's backend, they are referenced as separate components in this chapter for the sake of clarity).

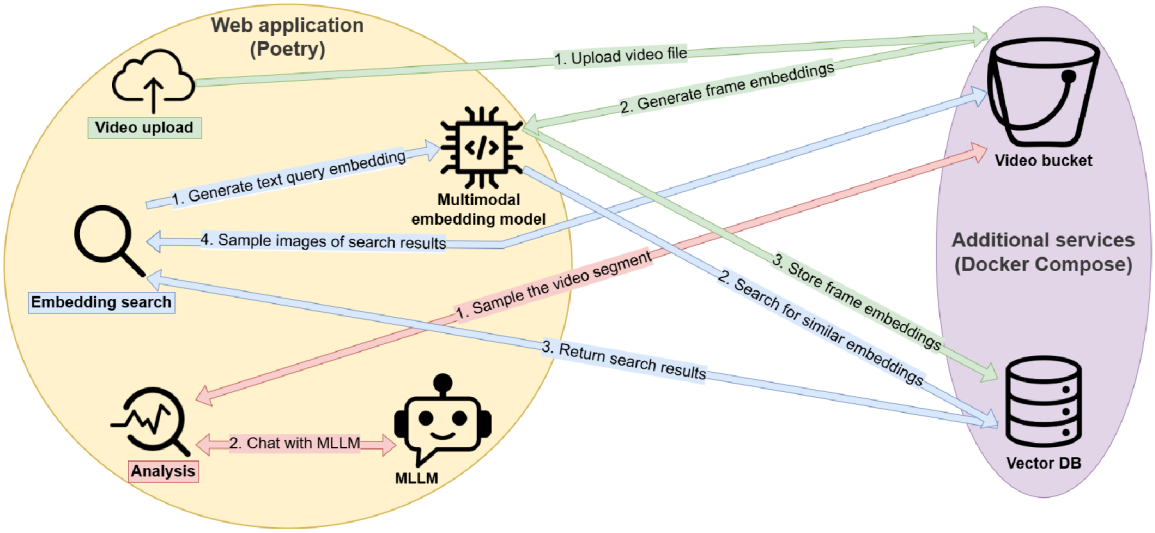

Figure 6.1: A simplified diagram of how the system components interact during three key scenarios. The diagram was created using draw.io[9].

6.2 The available AI models

As indicated in sections 4.2 and 4.3, the application allows users to select (configure) one of the four available multimodal embedding models (CLIP [71], ALIGN [33], SigLIP [107], BLIP [44]), and one of the four available MLLMs[10] (LLaVA-OneVision [42], GPT-4o [63], VideoLLaMA 3 [108], Qwen2.5-VL [4]). Benchmarks of both the available multimodal embeddings models and the available MLLMs, on datasets closely related to traffic footage, will be introduced and evaluated later in sections 7.1 and 7.2 respectively. The domain-specific experiments conducted with the models were not extensive, so their results may be considered anecdotal, which futher highlights the importance of the benchmarks.

All available models, except for GPT-4o, run locally to ensure that the system operates independently of (often paid) third-party services. The specific model versions (for example, Qwen2.5-VL comes in 3B, 7B, and 72B variants) were carefully chosen to match the hardware constraints of the reference machine equipped with 12-core AMD Ryzen 5900x

[7] On-demand frame sampling from stored videos proved itself to be sufficiently efficient, so the alternative approach of storing sampled images was not required.

[8] MinIO Object Storage uses buckets to organize objects. A bucket is similar to a folder or directory in a filesystem, where each bucket can hold an arbitrary number of objects.

[9] https://draw.io/

[10] Since all these models fit the standard definition of an MLLM, the broader term *MLLM-based model* will no longer be used.

CPU, NVIDIA GeForce RTX 4070 Super GPU with 12 GB of VRAM (the amount of video memory is definitely the most limiting factor, as both the multimodal embedding model and the MLLM need to be loaded there simultaneously), and 32 GB of RAM. All models, except for BLIP and GPT-4o, are utilized via Hugging Face's `transformers` Python package[11], which provides an interface for loading, fine-tuning, and running pre-trained models efficiently.

Available multimodal embedding models

Since most of the VRAM has to stay available to the MLLM, more "pure" multimodal embedding models (models whose abilities are limited to generating embeddings, measuring cross-modal similarity, etc., as opposed to complex models where embedding generation isn't the only main focus) were strongly preferred. It should be noted that when switching to a different embedding model, all entries in the Milvus vector database have to get replaced (the synchronization mechanism which automates this process will be described in section 6.3), as each of these models has its own embedding space, and the fact that their embedding vectors also have different lengths makes it easy to detect incompatible embedding entries. The following is a list of the available multimodal embedding models, with each entry accompanied by the specific version of the model, brief description of the process of how and why the model (and the version) was selected, and other relevant details:

- **CLIP** – This wasn't even really a choice, since CLIP is usually considered as the main reference point for this type of models. The specific version used is the ViT [19] based `openai/clip-vit-large-patch14`. At first, experiments were done using the smaller `openai/clip-vit-base-patch32` version, but the results were very unsatisfactory, as its traffic footage related abilities basically peaked at recognizing a bus or a person crossing the street. In contrast, ViT-L/14 was not only able to differentiate between different vehicle types (sedan, van, etc.) with reasonable accuracy, but also often recognized specific car brands or even specific models.

- **ALIGN** – Since CLIP's larger version performed so well, there was a strong focus on finding models which are very similar (basically drop-in replacements), but are made by different companies and were trained on different sets of data. The specific version of ALIGN that's being used is `kakaobrain/align-base`. The model demonstrated similar capabilities to CLIP ViT-L/14 in the context of traffic footage, although it didn't take much experimenting to notice that CLIP performs significantly better. But some trade-offs were expected, as ALIGN's base model only uses half the VRAM compared to CLIP ViT-L-14 (approximately 950 MiB versus approx. 1900 MiB) and its embedding vectors are also more compact (with a length of 640 versus 768).

- **SigLIP** – This model was chosen for similar reasons to ALIGN, but SigLIP is actually expected to outperform CLIP, since the `google/siglip-so400m-patch14-384` version (which is the most downloaded version on Hugging Face, is pre-trained on the WebLi dataset [9], and has the SoViT-400m architecture [1]) that's being used occupies over 4 GiB of VRAM and its embedding vectors have a length of 1152. Experiments have shown that although it usually outperforms CLIP ViT-L/14 (e.g., it is often better at identifying specific car models), the model can be quite inconsistent and unpredictable. For instance, when queried with "van", all of the retrieved

[11] https://huggingface.co/docs/transformers/index

images featured at least a few people but did not contain any vans. Only when the query was changed to a more specific "white van" did the model actually retrieve the expected video frames, whereas both of the previous models correctly processed the more general query.

- **BLIP** – BLIP (only its feature extractor) was chosen to fulfill the role of a model, which is still of the same type, but is less related to CLIP than the two previous models. Initially, the newer BLIP 2 [43] was planned to be used, but no way to separately access its feature extractor was found. Unlike the previous models, the BLIP feature extractor is imported from the `salesforce-lavis` Python Package. Since this package requires an older version of the `transformers` package, BLIP cannot be used simultaneously with any of the available local MLLMs (LLaVA-OneVision, VideoLLaMA 3, Qwen2.5-VL). Experiments have shown that although the model performs significantly better than the smaller version of CLIP (ViT-B/32), it also performs significantly worse than all three of the other used multimodal embedding models. Its embedding vectors have a lenght of only 256 (projected from 768-dimensional feature vectors) and the model uses about 1150 MiB of VRAM.

Other multimodal embeddings models that were at least briefly considered include OpenCLIP [11] (which was considered too closely related to CLIP), Meta's ImageBind [23] (which basically just extends CLIP to more modalities), ColPali [21] (which is too specialized for document retrieval), ImageBert [68], VisualBert [45], LXMERT [85], and UNITER [10].

Available MLLMs

It should be noted that getting different MLLMs (or MLLM-based models) to run was generally much more troublesome than running multimodal embedding models, usually due to long and often inconsistent lists of package dependencies and problems with CUDA. To meet the hardware restrictions (primarily the limited amount of VRAM) and improve performance, the local models (every available MLLM except GPT-4o) are configured to use several optimization techniques, such as 4-bit quantization (via either the `bitsandbytes` or the `autoawq` package) and the Flash Attention algorithm (via the `flash-attn` package). The following is a list of the available MLLMs, with each entry accompanied by the specific version of the model, brief description of the process of how and why the model (and the version) was selected, and other relevant details:

- **LLaVA-OneVision** – Since LLaVA [50] is one of the most well-known open-source MLLMs, incorporating one of its video-focused derivatives seemed very appropriate. LLaVA-OneVision (version `llava-hf/llava-onevision-qwen2-7b-ov-hf`) seemed to perform the best, by—for example—correctly counting the six cars parked next to a sidewalk and even correctly identifying the colors of some of them (see figure 6.2). Experiments were also done with Video-LLaVA [47] (version `LanguageBind/Video-LLaVA-7B-hf`), which is the oldest of these three models and also performed the worst (e.g., it recognized the situation in figure 6.2 as "an empty street with a long line of 10 red and blue cars parked on the side of the road"), and LLaVA-Next-Video [49] (version `llava-hf/LLaVA-NeXT-Video-7B-hf`), which still performed noticeably worse than LLaVA-OneVision (e.g., it only recognized three of the parked vehicles) and is also one of its direct predecessors [42, p. 34]. Additionally, a version of LLaVA-OneVision called `shashank23088/llava-onevision-qwen2-7b-traffic` was discov-

ered, which appears to be fine-tuned for traffic videos. However, attempts to run this version were unsuccessful.

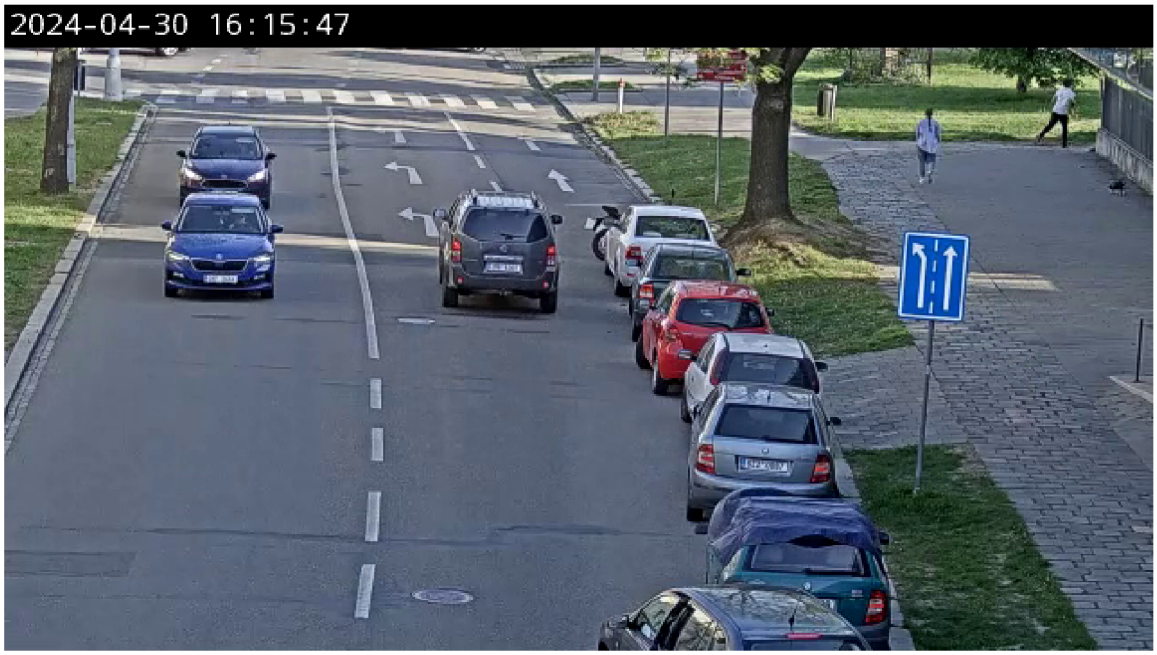

Figure 6.2: Frame from a traffic video captured on Božetěchova street, which is at the center of the video segment described by the tested LLaVA models.

- **GPT-4o** – This is the only model that doesn't run locally. Its inclusion was driven by a perfectly timed convenience—the company where I was employed during the writing of this thesis had a significant amount of unspent Azure credits expiring in June and granted me access to the API key. The inclusion of a full-size flagship model (all of the utilized local models are scaled-down versions) will help demonstrate the full capabilities of the system without the need to adhere to the hardware limitations. As expected, experiments have shown that its performance is significantly superior to the locally run models, providing more detailed scene descriptions and making fewer errors. Communication with the remote model is implemented using the `openai` Python package, and the application supports both OpenAI and Azure deployments.

- **VideoLLaMA 3** – This model wasn't specifically selected for a particular reason, but rather was one of the models that were quite easy to get running and was included mainly to just introduce more diversity. The specific version used is `DAMO-NLP-SG/VideoLLaMA3-7B-Image`, which accepts image inputs (instead of video files). Even though the model is more recent than all of the previously tested LLaVA models, experiments have shown that its traffic footage related performance is quite underwhelming and somewhat comparable to that of LLaVA-Next-Video.

- **Qwen2.5-VL** – Seeing how often various MLLM-based models use Qwen2 as the LLM component (e.g., the utilized version of LLaVA-OneVision) was the main initiative behind incorporating their latest open-source vision-language model—Qwen2.5-VL. The specific version used is `Qwen/Qwen2.5-VL-7B-Instruct-AWQ`, which utilizes AWQ (Activation-aware Weight Quantization for LLM Compression and Acceleration) [48].

Experiments have shown that the model significantly outperforms the other local models in most scenarios, often being basically comparable to GPT-4o.

MLLM-based models that were also considered, but weren't included in the end, include OmAgent—the original version [112], which was excluded not only for its core philosophy shift since version 2.0, but also because of the additional services needed to run the original model and the amount of OpenAI API calls it usually performed, NVILA by NVIDIA [53], excluded because all attempts to run the model were unsuccessful, and MiniGPT4-Video [2].

6.3 Backend of the web application

The backend of the web application, built with FastAPI, runs on Uvicorn[12]—an asynchronous server designed for high-performance web applications. The `python-dotenv` package is used to let users configure the application by changing the values of configuration variables found in the `/app/.env` file. The configuration variables mainly include the host and port to run the application on, configuration for the Milvus and Minio connections, names of the selected AI models (multimodal embedding model and MLLM, each of which can also be set to `None`, which will limit the application's capabilities correspondingly), and variables for communicating with OpenAI API (with support for Azure) when using GPT-4o. Users can also choose whether the frontend will be used (mounted at `'/'` via `StaticFiles`) or not (in which case, `'/'` serves as a redirect to the API documentation).

The backend's Python source code is divided into 7 files, all located in the `/app/src` folder. First, `__init__.py` is executed, loading the environment variables and the selected (configured) AI models. The `main.py` file contains the application setup, including API endpoints, and also executes `db_and_storage.py`, which manages database and storage setup along with related functions. Functions related to the application's core functionalities are distributed across `preprocess.py` (video frame embedding extraction), `search_embeddings` (embedding search), and `analyze_video.py` (video analysis using MLLMs). The `utils.py` file is meant to contain additional utilities, like converting timestamps from seconds to a more human-readable format (HH:MM:SS.sss).

The rest of this section consists of descriptions of the API endpoints from a more functional perspective, and descriptions of some important or interesting implementation details from a more technical perspective.

API endpoints

All backend endpoints share the `/api` prefix to avoid interference with the frontend routes. The first group of endpoints is related to the embedding search and the video analysis with an MLLM:

- `/api/search-embeddings` (GET) – Given an input query (`str`) and a number of top-k results to return (`int`), it returns a list of the k results found in the vector database. Each result consists of the corresponding video name (`str`), timestamp both in seconds (`float`) and in human-readable format (`str`), and the similarity score (`float`).

- `/api/image-from-video/{video_name}` (GET) – This endpoint primarily allows the frontend to directly display search results as images. Given a video name (`str`) and

[12] https://www.uvicorn.org/

a timestamp in seconds (`float`), it returns a JPEG image of the corresponding frame sampled from the corresponding video file stored in the Minio bucket (`bytes` object).

- `/api/start-video-analysis/{video_name}` (POST) – Allows the user to start a new analysis conversation with the currently configured MLLM. Given a video name (`str`), a user's initial query (`str`), the start end end timestamps in seconds (both `float`), and a number of frames to uniformly sample from the specified video segment (`int`), it samples the corresponding frames, creates the initial prompt with instructions for the MLLM, and prompts the model. It returns the MLLM's isolated text response (`str`) and the conversation history (`list` of `dict` entries, alternating between the user's and the MLLM's messages) with the user's initial prompt and the MLLM's response, which is required to continue the conversation.

- `/api/continue_video_analysis` (POST) – Allows the user to continue an existing analysis conversation with the currently configured MLLM. Given a user's query (`str`) and an existing conversation history (`list` of `dict` entries), it prompts the MLLM with the existing conversation history (updated with the user's query), and again, returns both the MLLM's isolated text response (`str`) and the up-to-date conversation history (`list` of `dict` entries). Since most of the available MLLMs cannot retain previously seen visual information—given that conversation history entries contain only references to the data provided in the initial prompt—the challenge of reintroducing the data to the model had to be addressed differently for each MLLM. The specific solutions to this issue, along with other implementation details, are discussed later in this section.

- `/api/video-url/{video_name}` (GET) – This endpoint primarily allows the frontend to load and display the specified video for analysis within a video player. Given a video name (`str`), it returns a URL of the video stored in the Minio bucket (`str`), which is valid for one hour.

The other group of endpoints, prefixed with `/api/data`, allows users to efficiently manage data stored in the system, which includes both the Milvus vector database and the Minio bucket, and the synchronization of their states:

- `/api/data/upload-video` (POST) – Given a video file (`UploadFile` object), specified sampling FPS (`float`), and `bucket_only` flag marking whether to only upload the video to the Minio bucket (`bool`), the video file is uploaded to the Minio bucket. If `bucket_only` is set to `False` (which is the default value), the video is also sampled and the frame embeddings are stored in the Milvus database.

- `/api/data/list-all` (GET) – This endpoint returns a list providing an overview of all data present in the system (`list` of `dict` entries). Each entry contains the video name (`str`), a flag marking whether the video file exists in the Minio bucket (`bool`), the number of corresponding embedding entries present in the Milvus database (`int`), and if the video file exists in the bucket, then also the video duration in both seconds (`float`) and in human-readable format (`str`).

- `/api/data/synchronize-video/{video_name}` (POST) – Given a video name (`str`) and specified sampling FPS (`float`), it deletes all corresponding embedding entries from the Milvus database and recreates them by re-sampling the entire video again. This is especially useful in situations when the multimodal embedding model has been

changed, or when the initial sampling process was interrupted and therefore didn't finish.

- `/api/data/synchronize-all` (POST) – This endpoint corresponds to the main synchronization mechanism. Given flag `force_bucket_mirror` marking whether to force mirroring of the Minio bucket's state (`bool`), and specified sampling FPS (`float`), if `force_bucket_mirror` is set to `False`, then all embedding entries without a corresponding video found in the Minio bucket are deleted from the Milvus database, and all videos in the bucket without any corresponding embedding entries get preprocessed (frame sampling and embedding extraction) again. If `force_bucket_mirror` is set to `True`, the entire collection of frame embedding entries is simply recreated, which is especially useful when the multimodal embedding model has been changed.

- `/api/data/delete-video` (DELETE) – Given a video name (`str`), it deletes all corresponding embedding entries from the Milvus database and removes the video file from the Minio bucket.

- `/api/data/delete-all` (DELETE) – Deletes all data, which includes all embedding entries stored in the Milvus database and all video files stored in the Minio bucket.

Implementation details

All operations regarding video files are implemented using the FFmpeg[13] multimedia framework, specifically via the `ffmpeg-python` package, which provides Python bindings. These operations include probing a video to obtain its metadata (e.g., duration), and extracting video frames either uniformly (for frame embedding extraction and video analysis preparation) or on demand (for interpreting search results as images).

A crucial implementation detail is how conversations with the configured MLLM are handled. It is based on the approach of passing a conversation object (popularized by OpenAI), where each message consists of a role ("user", "assistant", or "system") and some content (text or other data). The initial instructions for the LLM assistant (e.g., "You are a helpful assistant.") are usually contained in a "system" message, but since usage examples of the utilized MLLMs only contained "user" and "assistant" messages, the initial prompt gets injected into the first user message to ensure it isn't ignored by the model. When sending the first message to the MLLM, the sampled frames—processed by the model's processor—are either referenced as a video within the conversation object (LLaVA-OneVision), converted to Base64-encoded strings and inserted directly into the conversation object (GPT-4o), or temporarily stored in `/app/tmp` as JPEG images, with each image being individually referenced by the conversation object. Since conversations with all models, except for GPT-4o, only store references to the visual data and not the data itself, an additional "system" message is inserted at the beginning of the conversation, containing the metadata required to resample the frames when needed. An example of a conversation object obtained by initializing a new conversation with Qwen2.5-VL (with only 4 video frames being sampled) is shown in listing 6.1.

[13] https://www.ffmpeg.org/

```
"conversation": [
    {
      "role": "system",
      "content": [
        {
          "type": "metadata",
          "video_name": "traffic_footage.mp4",
          "start_timestamp": 1926,
          "end_timestamp": 1930,
          "num_frames": 4
        }
      ]
    },
    {
      "role": "user",
      "content": [
        {
          "type": "image",
          "image": "./tmp/traffic_footage.mp4_1926.000.jpg"
        },
        {
          "type": "image",
          "image": "./tmp/traffic_footage.mp4_1927.333.jpg"
        },
        {
          "type": "image",
          "image": "./tmp/traffic_footage.mp4_1928.667.jpg"
        },
        {
          "type": "image",
          "image": "./tmp/traffic_footage.mp4_1930.000.jpg"
        },
        {
          "type": "text",
          "text": "You are a video analysis assistant inside a specialized
              system for analyzing traffic footage.\nThis prompt includes 4
              images consecutively sampled from video named 'traffic_footage
              .mp4', each taken at a different timestamp. The timestamps (in
               the corresponding order and in HH:MM:SS.sss format) are:
              00:32:06.000, 00:32:07.333, 00:32:08.667, 00:32:10.000.\nYour
              goal is to have a conversation with the user about the
              situation in video section shown to you, don't talk about
              anything else or make up information about the video.\nDon't
              include any comments about your possibly limited video
              processing abilities, and do your best to interpret the images
               as a continous video.\nDon't blindly agree to everything the
              user says unless you saw proof in the footage. Try to be
```

```
            helpful and informative.\n\nThis is the user's query: What
            happens in the video segment?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "type": "text",
          "text": "The video segment shows a sequence of frames captured
              from a traffic camera. Here's what appears to happen:\n\n- At
              the beginning, there is a car driving towards the camera on
              the left side of the road.\n- As time progresses, this car
              continues to move forward, maintaining its lane.\n- There are
              no significant changes in the environment or other vehicles'
              behavior during these frames; the scene remains consistent
              with a steady flow of traffic.\n\nIf you need further details
              or analysis, please let me know!"
        }
      ]
    }
  ]
```

Listing 6.1: An example of a conversation history object obtained by starting a new analysis session with the Qwen2.5-VL MLLM.

The application is written in a way which makes it rather easy to add support for additional multimodal embedding models or MLLMs (refer to the application's README for more information). Due to the varying release dates of the AI models used, some of their package dependencies are incompatible with each other. For example, Qwen2.5-VL requires `transformers` version 4.49.0 or newer, LLaVA-OneVision requires a slightly older version (e.g., 4.47.0), and the `salesforce-lavis` package, which is necessary for running BLIP, requires a much older version (at least during the package installation process). To assist the user in managing these conflicts, `/app/pyproject.toml` contains comments that indicate which package requirements should be commented out when using certain models.

6.4 Frontend of the web application

The application's frontend, located in the `/app/frontend` folder, was developed using Vue.js—a progressive JavaScript framework for building user interfaces and single-page web applications (SPAs). While other frameworks, namely React, were also considered, Vue.js was finally chosen for its lightweightness and gentle learning curve. The goal was to make a simple, intuitive, and easy to use user interface without any unnecessary complexity. Additionally, Vite[14] is used as the build tool, while NPM[15] (Node Package Manager) manages the frontend's package dependencies (node modules). The code compiled by Vite

[14] https://vite.dev/
[15] https://www.npmjs.com/

(which has to be recompiled after any modifications) is stored in the `/app/frontend/dist` folder, which is then mounted to the backend via FastApi's `StaticFiles`.

A key part of the frontend are the three separate views, which define the three main ways a user interacts with the system, and whose source code files can be found in the `/app/frontend/src/views` folder. Routing between these views is managed using the `vue-router` package. The rest of this section aims to describe the separate views, mainly from the perspective of a user interacting with the system, while also mentioning certain technical details. All views share the same top bar, which contains buttons that allow the user to switch between them (see the top-left of figure 6.3).

Search view

The search view (see figure 6.3) allows the user to search for a specific moment within the entire collection of videos and instantly interpret the results. Once the user enters the inputs (the search query and the number of top-k results to show) and clicks the 'Search' button (or presses the `Enter` key), the search results are displayed in two columns, with the first two entries appearing in the first row. Each entry consists of the corresponding image (obtained by calling the `/api/image-from-video` endpoint, which works surprisingly quickly), the corresponding metadata, and a button that can redirect the user directly to the analysis view with already prefilled parameters for analyzing the given video segment.

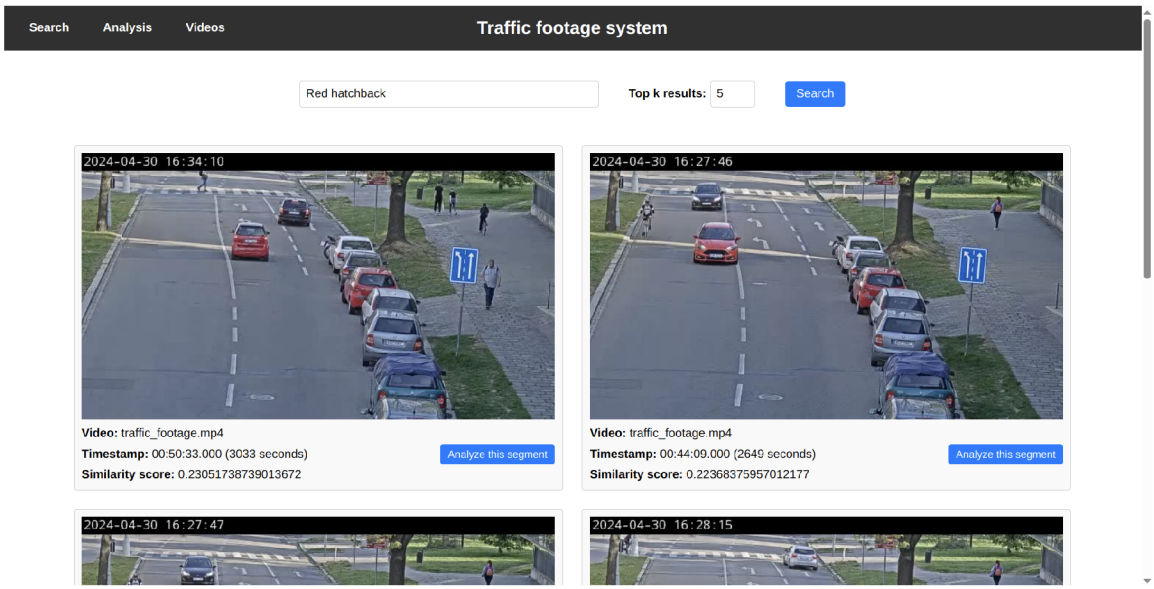

Figure 6.3: The application's search view, showing the results when the user enters the query 'Red hatchback'. All displayed images are sampled from a traffic video captured on Božetěchova street.

Analysis view

The analysis view (see figure 6.4) allows the user to analyze a specified video segment by interacting with the currently configured MLLM. Unless the user is redirected from the search view, he first enters the inputs at the top, which specify the analyzed video segment (the video name, the start and end timestamps, and the number of frames to

uniformly sample). For more intuitive usage, seconds are also automatically converted to structured timestamps, the sampling FPS is automatically calculated from the segment's parameters, and the video selector suggests existing videos whose names match the entered prefix (the `/api/data/list-all` endpoint is always called upon entering the analysis view). Once the 'New analysis' button is pressed, a video player (automatically set to the start timestamp), "frozen" analysis parameters (which are the ones specifying what the MLLM actually "sees"), and a chat window appear. The user can then play the video and chat with the model through the rather familiar chat interface with message bubbles.

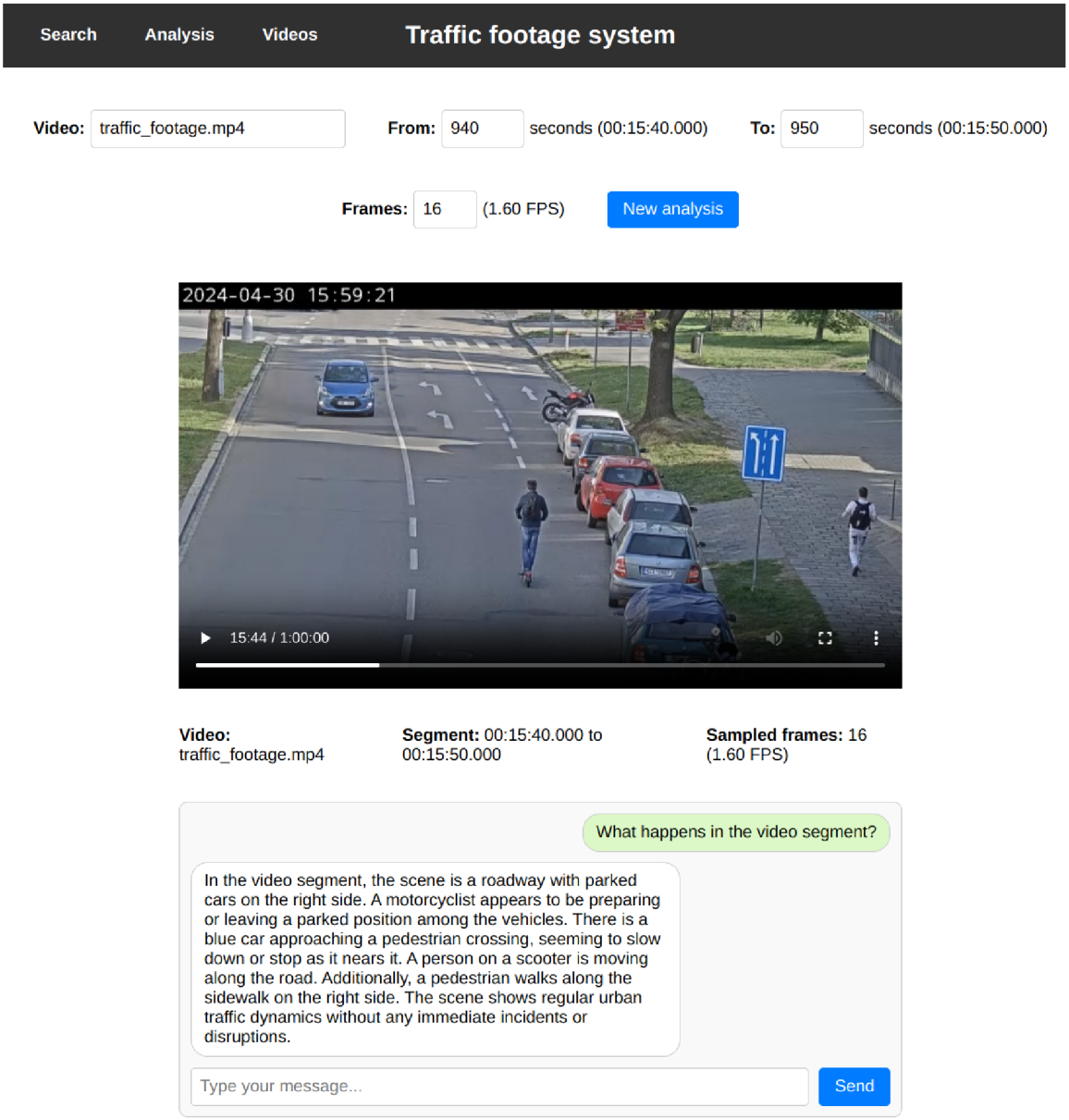

Figure 6.4: The application's analysis view, showing an exemplary video analysis with the GPT-4o MLLM. The analyzed video was captured on Božetěchova street.

Videos (data management) view

The videos view (see figure 6.5) allows the user to upload new videos and manage all existing data. At the top, the user can upload a new video file, choose whether to upload the video only to the Minio bucket, or, if not, select the sampling FPS for the embedding extraction (be aware that the selected value is also used for any synchronization actions that include embedding extraction). For each entry, the structured list of videos, which gets automatically updated after every action without the need to refresh the page, displays the corresponding information and buttons that allow the user to either synchronize or delete the video. The list is paginated in groups of up to 25 entries, which is not visible in figure 6.5 as there are only three videos shown. The bottom of the page contains buttons that allow the user to perform actions that affect all existing data ('Synchronize all data', Delete all data').

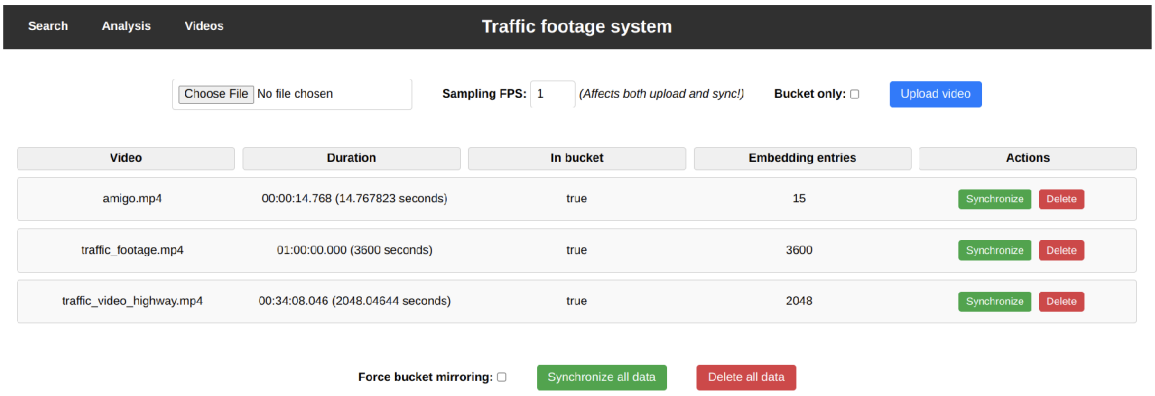

Figure 6.5: The application's videos (data management) view, showing all three videos present in the system at the moment.

Chapter 7

Benchmarks of the available models and evaluation of the system

This chapter guides the reader through the course of the performed benchmarks[1] of the AI models, all of which are available for use in the implemented system, on datasets very closely related to traffic footage. The importance of the benchmarks stems from the fact that none of the models were fine-tuned for this specific domain, so the results should validate whether the models were selected sensibly, and help the user choose the correct model for a specific use case. The system is then briefly evaluated as a whole, preferably with the best-performing models configured.

Section 7.1 describes two text-to-image retrieval benchmarks of the multimodal embedding models—first related to car brands and models, second related to Czech traffic signs and their categories. Section 7.2 describes a benchmark of the MLLMs on a traffic videos QA (Question Answering) dataset. Section 7.3 is about evaluating the whole system based on its usability in common use cases.

7.1 Benchmarks of the multimodal embedding models

The two benchmarks in this section are described "in parallel" to avoid repeating information that is common to both of them (the model specifications, the used metrics). In both benchmarks, the vector database contains a collection of image embeddings generated by the benchmarked model, and retrieves k images based on a text query (one for each class name) embedded by the same model.

In the first benchmark, the database always retrieves k images of a specific car model from a collection containing various images of 196 specific car models (there is also a "sub-benchmark" measuring whether at least the car brand was correct). In the second benchmark, the database always retrieves k images taken in the Czech Republic, each containing up to 10 various traffic signs (a correct image was retrieved when at least one sign in the image corresponds to the query, and there is also a "sub-benchmark" measuring whether at least the sign category was correct).

[1] Unlike the previous chapter, this chapter doesn't mention the specific files and folder structure. To replicate the performed benchmarks, see the README located in the `/model_benchmarks` folder.

Model specifications

To be able to draw any conclusions from the results of the benchmarks, it is crucial to specify the specific model versions and other variables that might affect the results (like the used version of the `transformers` package). It is also useful to list values related to the models' sizes (like the peak VRAM usage), to be able to assess the performance-per-cost of each model. These specifications are shown in table 7.1.

Model	Model version	Embedding dimensions	Version of `transformers`	Peak VRAM usage (MiB)
CLIP	`openai/clip-vit-large-patch14`	768	4.47.0	1920
ALIGN	`kakaobrain/align-base`	640	4.47.0	964
SigLIP	`google/siglip-so400m-patch14-384`	1152	4.47.0	4068
BLIP	`blip_feature_extractor`	256	4.26.1	1180

Table 7.1: The specifications of the multimodal embedding models used in the CARS196 and Czech traffic signs text-to-image retrieval benchmarks.

CARS196 dataset

The CARS196 dataset, often referred to as the Stanford Cars dataset [39], is a comprehensive collection comprising 16,185 images of cars spanning 196 different classes. Each class corresponds to a specific car model and comprises of the brand, model, and year (e.g., BMW M3 Coupe 2012). The dataset is further divided into 8,144 training images and 8,041 testing images, maintaning an approximate 50-50 split within each class. The dataset, created in 2013 by researchers at Stanford University and Max Planck Institute for Informatics, is often used in the context of fine-grained image classification and other computer vision tasks. An example of an image from the dataset is shown in figure 7.1.

Figure 7.1: An example image (BMW M3 Coupe 2012) from the train subset of the CARS196 dataset [39].

Only the training subset of CARS196 is used for the benchmark, as the test annotations could not be located online despite multiple attempts. Although preliminary testing indicated that the results should provide reasonable information value, all metrics will additionally be recalculated based on whether the retrieved images belong to the correct car brand (these results will always be, in principle, better than the original ones). Please note that the benchmark itself only runs once, and the different levels of granularity only refer to different interpretations of the same results. Each of the 196 embedded text queries will always correspond to the full class name (brand, model, year), which is very different than running the benchmark a second time with only 48 queries corresponding to the brand names. Important characteristics of the CARS196 train subset are shown in table 7.2 (there are originally 49 car brands, but Hummer and AM are treated as a single brand).

Granularity	**Classes**	**Most common class**	**Least common class**	**Avg. images per class**
Car models	196	GMC Savana Van 2012 (68 images)	Hyundai Accent Sedan 2012 (24 im.)	41.55
Car brands	48	Chevrolet (905 im.)	Maybach (29 images)	169.67

Table 7.2: Class count characteristics of the CARS196 dataset's train subset (8,144 images), based on different levels of granularity—car models (original) and car brands (generalized).

Czech traffic signs dataset

This dataset, which doesn't have an official name, was provided by the thesis' supervisor along with the videos from Božetěchova street. It consists of 3,040 images of traffic signs (only 2,294 of which are annotated and therefore will be used), which were extracted from panoramatic dashcam photos taken on Czech roads, mostly in the area of Trutnov. Each image contains one to ten (not necessarily unique) traffic signs, and the dataset contains 152 unique traffic signs. Each extracted image shows the traffic sign from two different angles, an example is shown in figure 7.2.

Figure 7.2: An example image from the Czech traffic signs dataset (B20a: Speed limit prohibitory traffic sign).

The traffic signs are labeled with corresponding alphanumeric codes (e.g., E13), so they had to be converted to English names to be used as the input queries. English names of certain Czech traffic signs identified by codes can be found on Wikipedia [97], and for the remaining signs, the Czech names were sourced from Decree No. 294/2015 Coll., which outlines the implementation rules for road traffic in the Czech Republic [62], and subsequently translated into English manually. In addition, the sign category names were appended to the sign names. Apart from the fact that a single image can correspond to multiple classes, the principle of this benchmark doesn't differ from the CARS196 one. Since the models were likely not trained on Czech data, the results are not expected to be optimal. However, they should still provide insight into which model handles this type of data most effectively. Important characteristics of the dataset are shown in table 7.3. Additionally, each annotated image contains on average 1.491 unique traffic signs (e.g., 3 stop signs in a single image count as 1 unique sign) and 1.357 unique traffic sign categories.

Granularity	Classes	Most common class	Least common class	Avg. images per class
Traffic signs	152	E13: Text or symbol supplementary plate traffic sign (329 im.)	20 different traffic signs (1 image each)	22.50
Traffic sign categories	10	Supplementary plate traffic sign (620 im.)	Mandatory traffic sign (27 images)	311.4

Table 7.3: Class count characteristics of the Czech traffic signs dataset (2,294 images), based on different levels of granularity—traffic signs (original) and traffic sign categories (generalized).

Used metrics

For an overview of common cross-modal retrieval metrics, refer back to the end of section 3.1. The metrics used for both benchmarks contain Precision at k (since it's the most basic and common metric), Average Precision at k (to also account for the order of the retrieved images), and in cases where it makes at least some sense—Recall at k (it is, by principle, only possible to achieve 100% recall when k is set to the number of occurrences of the most common class), all calculated for multiple k values. Normalized discounted cumulative gain at k isn't used, as there are no orderings of the retrieved samples based on their correlations to input queries. Instead, the metrics are also separately calculated for the generalized classes (car brands, traffic sign categories). Since the system is intended to work with images that are subsequently sampled from videos, the possibility of designing a hybrid metric—based on those used in TSGV (see section 2.3)—was considered (which would also require a video-focused dataset). However, this idea was later abandoned. Below is an exact overview of all metrics with specific k values used (and brief explanations of how the k values were chosen):

- **CARS196 – Car models**
  - P@1, P@5, P@10 (commonly used values), P@24 (least common class' count), P@42 (average number of images per class)

  - AP@1, AP@5, AP@10 (see above), AP@24 (see above), AP@42 (see above)
  - R@42 (average number of images per class), R@68 (most common class' count)

- **CARS196 – Car brands**
  - P@1, P@5, P@10 (commonly used values), P@29 (least common class' count), P@68 (the average number of images per class—169.67—is too high)
  - AP@1, AP@5, AP@10 (see above), AP@29 (see above), AP@68 (see above)

- **Czech traffic signs – Signs**
  - P@1, P@5, P@10 (commonly used values), P@23 (average number of images per class)
  - AP@1, AP@5, AP@10 (see above), AP@23 (see above)
  - R@23 (average number of images per class)

- **Czech traffic signs – Sign categories**
  - P@1, P@5, P@10 (commonly used values), P@27 (least common class' count)
  - AP@1, AP@5, AP@10 (see above), AP@27 (see above)

Each of the two benchmarks (CARS196, Czech traffic signs) only needs to be run once, with top-k set to the highest k value occurring in the corresponding metrics. To avoid ambiguity, let's give an example of how the metrics are calculated. Let's say the database is supposed to retrieve 5 images belonging to class 'a', class 'a' occurs in a total of 10 images, and the retrieved results (starting with the first retrieved) are ('a', 'b', 'a', 'c', 'a'). Precision@5 is then ($\frac{3}{5} = 0.6$), Recall@5 is ($\frac{3}{10} = 0.3$), and AP@5 is ($\frac{1}{3} \cdot (1 \cdot 1 + 0 \cdot \frac{1}{2} + 1 \cdot \frac{2}{3} + 0 \cdot \frac{2}{4} + 1 \cdot \frac{3}{5}) = \frac{34}{45} \approx 0.7555$). The mean of any *metric*@5 is then the sum of its values corresponding to individual input queries, divided by the total number of input queries.

Evaluation of CARS196 benchmark

The results related to specific car models are shown in table 7.4. Please note that there are certain metrics (mP@42, mAP@42, mR@42) for which it is, in principle, impossible to reach a value of 1.0. Ignoring these "@42" metrics, all values fall within a relatively narrow interval from 0.47655 to 0.85694, which suggests that all benchmarked models are reasonably suitable for retrieving specific car models from traffic footage. The largest model, SigLIP, performed the best across all metrics with the exception of mean Recall at 68, where it ranked second to CLIP. At the opposite end, BLIP performed the worst across all metrics without any exceptions. Taking model sizes into account, CLIP's performance is very close to SigLIP for less than half the price, and the smallest model—ALIGN—performs closer to the leading models than to BLIP.

Metric	Model			
	CLIP	**ALIGN**	**SigLIP**	**BLIP**
mP@1	0.79592	0.77041	0.81633	0.68367
mP@5	0.77653	0.74286	0.80612	0.69694
mP@10	0.75510	0.72806	0.77500	0.66122
mP@24	0.70897	0.66454	0.72428	0.57568
mP@42	0.62731	0.57981	0.63703	0.47655
mAP@1	0.79592	0.77041	0.81633	0.68367
mAP@5	0.85534	0.81938	0.85694	0.77727
mAP@10	0.82683	0.79780	0.84051	0.75197
mAP@24	0.78917	0.75446	0.80316	0.69659
mAP@42	0.75344	0.71681	0.77215	0.65221
mR@42	0.63697	0.58736	0.64758	0.48406
mR@68	0.78835	0.73625	0.76961	0.61991

Table 7.4: Results of the CARS196 benchmark related to car models. The values of all metrics are rounded to the fifth decimal place.

The results related to car brands are shown in table 7.5. Ignoring the *impossible* "@68" metrics, all values fall within an even shorter interval from 0.86013 to 0.98638, so the models seem to be really good at recognizing car brands. Again, SigLIP is always the best (now without an exception), and BLIP is always the worst, but interestingly, BLIP holds up to the other models very well this time. Also, ALIGN's performance is much closer to CLIP than previously, and it even outperforms it with its first retrieved results ("@1" metrics).

Metric	Model			
	CLIP	**ALIGN**	**SigLIP**	**BLIP**
mP@1	0.96939	0.97449	0.97959	0.96939
mP@5	0.97143	0.96939	0.98061	0.94388
mP@10	0.96429	0.96224	0.97194	0.92245
mP@29	0.94053	0.93139	0.94634	0.86013
mP@68	0.83891	0.83268	0.87695	0.74445
mAP@1	0.96939	0.97449	0.97959	0.96939
mAP@5	0.98493	0.97707	0.98638	0.97076
mAP@10	0.97778	0.97446	0.98625	0.95786
mAP@29	0.96688	0.96002	0.97048	0.92449
mAP@68	0.94317	0.93309	0.95199	0.87751

Table 7.5: Results of the CARS196 benchmark related to car brands. The values of all metrics are rounded to the fifth decimal place.

Figure 7.3 shows a plot of the mean Precision@5 metric for both car models and car brands. Since the values across most metrics lie within relatively narrow ranges, visualizing additional metrics would not provide much added intuitive value.

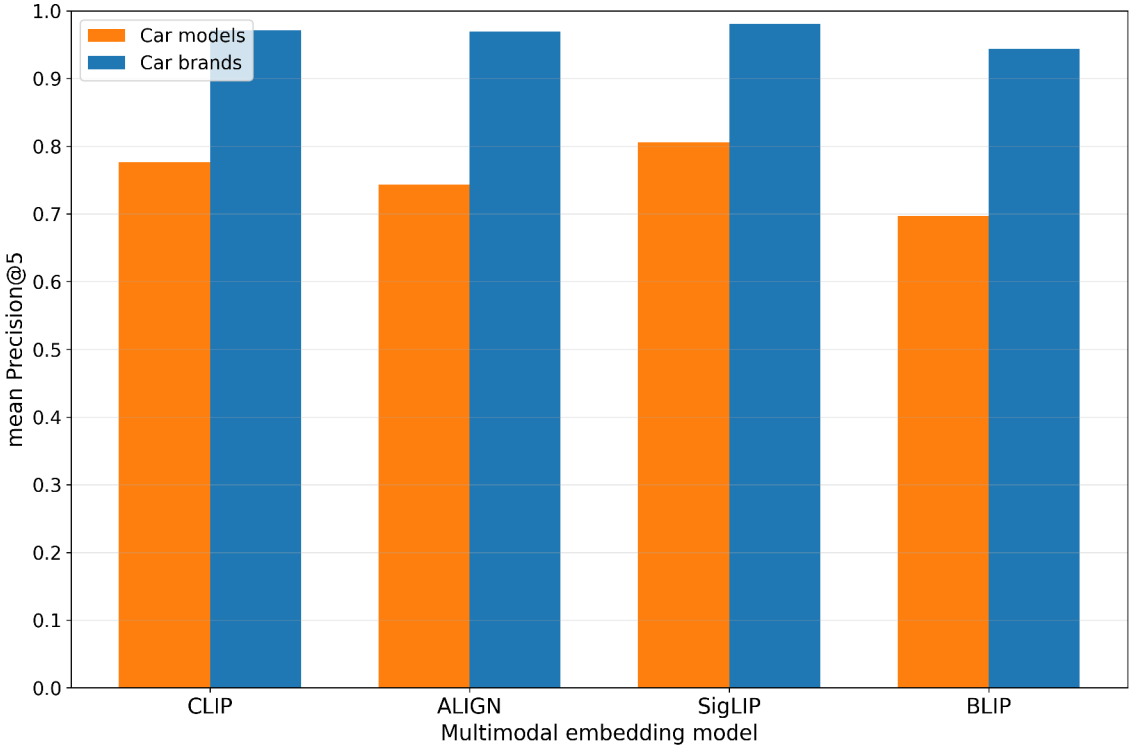

Figure 7.3: Plot of the mean Precision@5 metric for both car models and car brands.

Evaluation of Czech traffic signs benchmark

The results related to specific traffic signs are shown in table 7.6. Even at first glance, it is clear that the results are nowhere near the results of the previous benchmark, as all values fall within interval from 0.06322 to only 0.20099—one can wonder whether traffic sign data from the USA would yield significantly better results, since the models should probably have better general understanding of traffic signs than this benchmark suggests. Overall, the best-performing model definitely seems to be ALIGN, which is also the most lightweight one, possibly due to some higher correlation between its training data and the nature of this dataset. CLIP and SigLIP seem to be on around the same level, with CLIP slightly outperforming the latter, and the worst performing model is definitely BLIP, with the lowest values of all of the metrics.

Metric	Model			
	CLIP	**ALIGN**	**SigLIP**	**BLIP**
mP@1	0.13158	0.15789	0.13158	0.09211
mP@5	0.11053	0.12895	0.10263	0.09211
mP@10	0.10526	0.11842	0.09539	0.07829
mP@23	0.08896	0.09497	0.08982	0.06322
mAP@1	0.13158	0.15789	0.13158	0.09211
mAP@5	0.19181	0.19600	0.17386	0.14537
mAP@10	0.18860	0.20099	0.18102	0.14363
mAP@23	0.17446	0.19194	0.17983	0.12990
mR@23	0.16238	0.15457	0.16630	0.08249

Table 7.6: Results of the Czech traffic signs benchmark related to specific signs. The values of all metrics are rounded to the fifth decimal place.

The results related to Czech traffic sign categories (see table 7.7) instead of specific signs seem to be a lot better (all values fall within interval from 0.43031 to 0.69189), but this may be caused by class imbalance. This time, both ALIGN and SigLIP seem to be best performing models, often with one outperforming the other not only based on the metric type, but also the top-k parameter. CLIP performed significantly worse than these two models, and even though BLIP ended up as the worst-performing model again, its performance is very close to CLIP.

Metric	Model			
	CLIP	**ALIGN**	**SigLIP**	**BLIP**
mP@1	0.57237	0.60526	0.61184	0.48684
mP@5	0.48684	0.59605	0.55658	0.48158
mP@10	0.48289	0.58092	0.54079	0.45855
mP@27	0.43713	0.51852	0.51901	0.43031
mAP@1	0.57237	0.60526	0.61184	0.48684
mAP@5	0.63247	0.67606	0.69189	0.61689
mAP@10	0.60350	0.66247	0.65618	0.58567
mAP@27	0.54702	0.62224	0.60236	0.51939

Table 7.7: Results of the Czech traffic signs benchmark related to traffic sign categories. The values of all metrics are rounded to the fifth decimal place.

Figure 7.4 shows a plot of the mean AveragePrecision@10 metric for both traffic signs and traffic sign categories (note that the CARS196 plot was related to mean Precision@5, which is a different metric).

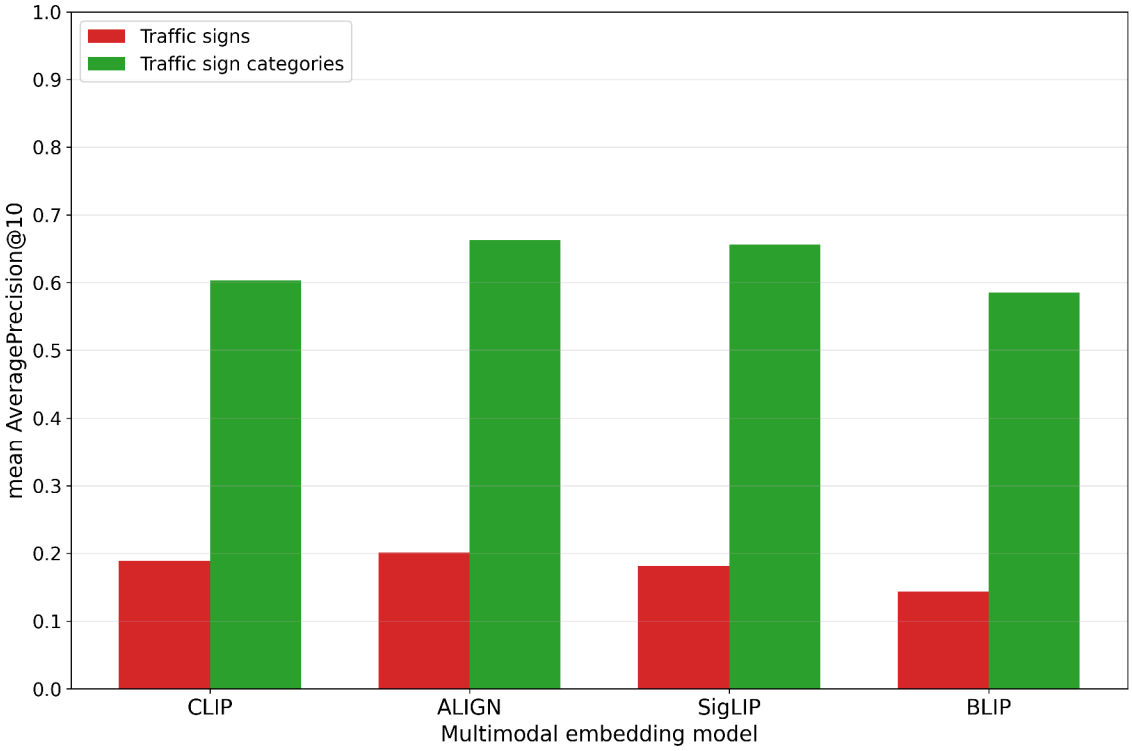

Figure 7.4: Plot of the mean AveragePrecision@10 metric for both car Czech traffic signs and Czech traffic sign categories.

Summary

In summary, all of the available multimodal embedding models are very good at distinguishing between car brands, and quite good at even differentiating between specific car models. However, their performance related to Czech traffic signs and their categories is rather poor, but that is to be expected given their training on broad, non-specialized datasets.

If a user has a lot of free VRAM available, he may want to use SigLIP for its (probably) best overall performance, although it uses as much VRAM as the three other models combined. On the other hand, ALIGN definitely provides the best performance-per-cost ratio and even seems to be the best at interpreting traffic sign data from the Czech Republic. CLIP emerges as the "jack of all trades" with its consistently strong performance across different domains, while consuming a reasonable amount of VRAM. Even though BLIP does not perform too poorly in absolute terms, there really seems to be no reason to use it instead of the other models.

7.2 Benchmark of the MLLMs

This section describes a video question answering (VQA) benchmark of the available MLLMs. In this benchmark, each model consecutively answers 2000 different questions about some event happening in a corresponding video of traffic, each with two to four available choices (only one option is correct; the model chooses exactly one option). Apart from the instructions and a question, each model is prompted with up to 16 frames uniformly sampled from a corresponding video.

Model specifications

Specifying the versions and configurations of the used models is even more important than in the multimodal embedding model benchmarks, since the local MLLMs are not the flagship versions of the models and they are also configured to use further optimizations (mainly to decrease VRAM usage). The following is a list of the benchmarked models with their corresponding configurations and other relevant information:

- **LLaVA-OneVision** – locally run open-source model; version: `llava-hf/llava-onevision-qwen2-7b-ov-hf`; `torch_dtype=torch.float16`; quantization: via `BitsAndBytesConfig` (4-bit, NF4, `torch.float16`); attention implementation: Flash-Attention-2; `transformers` version: 4.47.0; peak VRAM usage: 7,728 MiB
- **GPT-4o** – remotely run (Azure) closed-source model, `api_version="2024-02-01"`
- **VideoLLaMA 3** – locally run open-source model; version: `DAMO-NLP-SG/VideoLLaMA3-7B-Image`; `torch_dtype=torch.bfloat16`; quantization: via `BitsAndBytesConfig` (4-bit, NF4, `torch.float16`); attention implementation: Flash-Attention-2; `transformers` version: 4.47.0; peak VRAM usage: 11,364 MiB
- **Qwen2.5-VL** – locally run open-source model; version: `Qwen/Qwen2.5-VL-7B-Instruct-AWQ`; `torch_dtype=torch.float16`; quantization: 4-bit via AWQ; attention implementation: FlashAttention-2; processor config: `min_pixels=(256 * 28 * 28), max_pixels=(600 * 28 * 28)`; `transformers` version: 4.49.0; peak VRAM usage: 11,156 MiB

SUTD-TrafficQA dataset

The SUTD-TrafficQA dataset is a comprehensive video question answering (VQA) benchmark designed to evaluate models' reasoning capabilities in complex traffic scenarios [101]. Developed by researchers from the Singapore University of Technology and Design (SUTD) and Lancaster University, it consists of 10,080 real-world traffic videos and 62,535 annotated question-answer pairs. Each question has either 2, 3, or 4 available answers (choices), out of which only one is correct. Each question belongs to one of six types of reasoning tasks: basic understanding (understanding events or objects in the video), event forecasting (infering future events based on the observed video), reverse reasoning (reasoning about events that might have happened before the start of the video), counterfactual inference (hypothesizing what might have happened in the video if the conditions were different), introspection (providing preventive advice, which might have prevented the accident), and attribution (explaining the causes of traffic events).

Since inference with the MLLMs is quite time-consuming, only the first 2,000 samples from the test subset (which comprises 6,075 questions in total) were used. Some of these questions correspond to a corrupted video file, so these questions had to be replaced with subsequent questions. Given the set of 2,000 questions used for the benchmark, there are roughly 3.6 options per question on average (353 2-option, 95 3-option, and 1552 4-option questions) and the 2,000 questions correspond to 1738 unique video files with an average length of about 6.562 seconds (the longest video is 186 seconds long and the shortest one is only 0.708 seconds). Videos shorter than 8 seconds are uniformly sampled at 2 FPS, longer videos are uniformly sampled so that the total number of sampled frames is 16. Listing 7.1 shows the prompt that was universally used for this benchmark. After inference

with all MLLMs was run and the benchmark was evaluated, it was found that some of the models quite often answered 2 or 3-option questions with a choice that was not presented. Instead of rewriting the last two lines of the prompt and re-running the entire benchmark (which would be very time-consuming), additional metrics were proposed to extract further insights into the models' capabilities from this "accident".

```
Please answer the following question about the consecutive video frames: '
    What is the traffic condition on the road?'.
You must choose exactly one answer from the following options:
Option 0: The road is quite sparse with a few cars.
Option 1: The road is congested and jammed.
Option 2: The road is busy, but not congested.
Option 3: There is barely any car on the road.
Only answer either '0', '1', '2', or '3' (if there is an available option
    corresponding to the number).
Your answer will be parsed so that the first occurence of a number ('0', '1
    ', '2', '3') in your answer will be considered as the final answer.
```

Listing 7.1: The prompt used for all MLLMs and questions in the SUTD-TrafficQA benchmark. The variable parts of the prompt, which depend on the presented question, are displayed in red.

Used metrics

Since there is no ordering of the incorrect options for each question based on relevance or partial correctness, most non-basic metrics are either based on the number of available choices for each question, or based on the format of the MLLM's answer. Following is a list of all used metrics (excluding the **total number of answered questions**, and the **number of questions answered correctly**):

- **Accuracy** – The number of questions answered correctly, divided by the total number of answered questions. Q in equation (7.1) represents the set of all answered questions.

$$Q_c = \{q \in Q \,|\, correctly_answered(q)\},\ Accuracy = \frac{|Q_c|}{|Q|} \tag{7.1}$$

- **Weighted accuracy** – This metric assigns a higher reward for correctly answering questions with more answer options (as they are less probable to be answered correctly by randomly guessing): 1 point for binary (2-option) questions, 1.5 points for 3-option questions, and 2 points for 4-option questions. The entire sum is normalized to obtain a score between 0.0 and 1.0, as shown in equation (7.2), where both Q and Q_c have the same meaning as in equation (7.1).

$$WeightedAccuracy = \frac{\sum_{c \in Q_c} num_available_choices(c)}{\sum_{q \in Q} num_available_choices(q)} \tag{7.2}$$

- **Four-choice question accuracy** – Accuracy as shown in equation (7.1), but Q represents the set of all answered 4-choice questions (this metric doesn't "punish" models that tend to pick unavailable choices).

- **Number of incorrectly formatted answers** – The number of times the model answered with anything other than `'0'`, `'1'`, `'2'` or `'3'` (even if the picked choice was then successfully extracted).
- **Number of unavailable choices picked** – The number of times the model picked a choice (from `'0'` to `'3'`) that was not available for the given 2 or 3-choice question.

Evaluation of SUTD-TrafficQA benchmark

The results of the benchmark are shown in table 7.8 (GPT-4o was unable to answer certain questions related to traffic accidents, as they violated OpenAI policies). It is clear that the only flagship, remotely run model (GPT-4o) performed the best overall, but the margin is nowhere near what was expected. The worst performing model was VideoLLaMA 3, and its only redeeming quality is that it is the only model that formatted its answer correctly every time (but chose an unavailable option quite often). Qwen2.5-VL is the best performing locally run model, even with the additional optimization of lowering the `max_pixels` parameter in its processor (to fit in the VRAM limit). Surprisingly good is the performance of LLaVA-OneVision (which is half a year older than Qwen2.5-VL), whose accuracy was negatively influenced by the striking number of times it picked an unavailable choice, as suggested by its four-choice questions accuracy, which is on par with that of Qwen2.5-VL.

Metric	**Model**			
	LLaVA-OneVision	**GPT-4o**	**Video-LLaMA 3**	**Qwen2.5-VL**
Questions answered	2000	1987	2000	2000
Correct answers	871	998	737	918
Accuracy	0.43550	0.50226	0.36850	0.45900
Weighted accuracy	0.44006	0.50063	0.36172	0.45437
Four-choice accuracy	0.44974	0.50162	0.35374	0.45103
Incorrectly formatted	3	1	0	3
Unavailable choices	120	12	88	19

Table 7.8: Results of the SUTD-TrafficQA benchmark. The values of all real-valued metrics are rounded to the fifth decimal place.

Similarly to the benchmarks of the available multimodal embedding models, visualizing the values of the used metrics doesn't provide much added intuitive value. Figure 7.5 shows a plot of the models' (standard) accuracy.

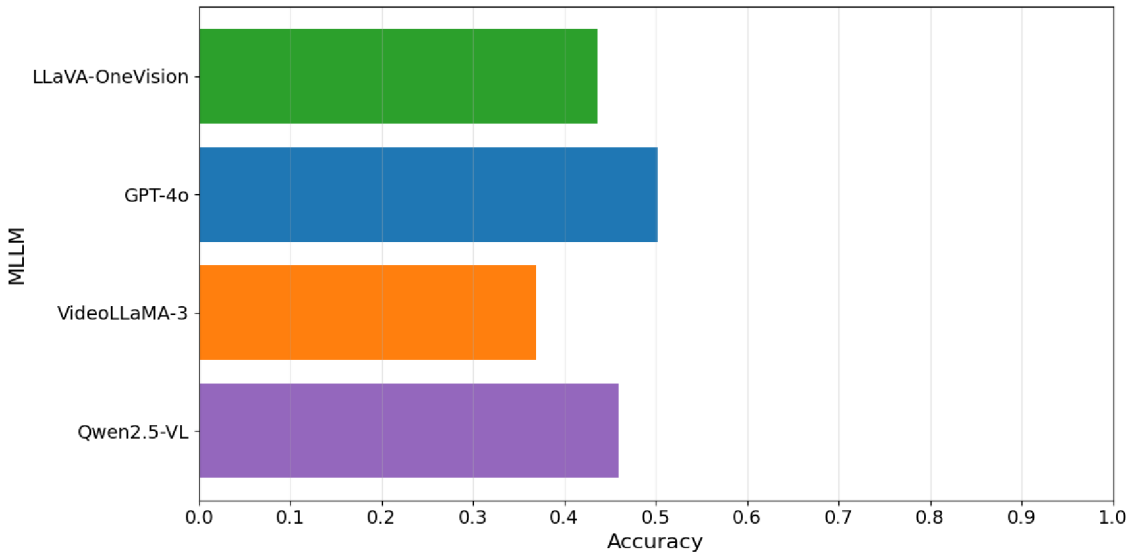

Figure 7.5: Plot of the MLLMs' accuracy in the SUTD-TrafficQA benchmark.

Summary

In summary, all of the available MLLMs are at least fairly good at understanding traffic footage. Even though the accuracy of all models ranged from about 36% to 50%, it's important to note that the dataset includes not only questions about interpreting current events in a given video, but also tasks involving future forecasting and other complex reasoning challenges (despite none of the models being a chain-of-thought model).

The best results are achieved with GPT-4o, but an average user probably doesn't want to spend money on OpenAI or Azure credits. Out of the local models, it may be tempting to recommend Qwen2.5-VL, but the one recommended model is actually LLaVA-OneVision for its much lower VRAM usage (which also doesn't seem to increase as drastically with additional images to process). The user just has to be very careful and specific when creating instructions for the model. There is probably no reason to use VideoLLaMA 3 in the system for anything other than just experimenting with the model.

7.3 Evaluation of the system

This section aims to briefly evaluate the system from a common user's perspective, by experimenting with the system in two different scenarios while using the two models that performed the best in the benchmarks (SigLIP, GPT-4o). Please note that the information in this section is mainly a summary of notes written during soft-testing of the system, so unlike the benchmark results, these results may be anecdotal and are intended to only complement the benchmarks, which remain the core focus of this chapter.

CCTV traffic footage from Božetěchova street

For the first experiment, about 18 hours of CCTV traffic footage (18 non-sequential hour-long videos, 6 of which were captured at night) from Božetěchova street (see figures 2.1, 2.2, 6.3, 6.4) was uploaded to the system and sampled at 1 FPS. Choosing a specific

moment in one of the videos and trying to locate it using the system was usually quite challenging. The system often succeeded in finding objects that are rather specific and occur rarely (new sports cars, a truck carrying rocks, etc.), but localizing more common events by trying to be very specific rarely worked (e.g., describing the unique combination of common economy cars present in the image). Since each embedding encodes an entire image, searching for something that is present in a non-center part of an image usually resulted in images where something different, but relevant to the query, is in the center. It is definitely recommended to include the general context in the search query—for example, the system is good at filtering the images based on the time of day and weather conditions (day, night, fog, etc.). SigLIP also demonstrated strong OCR capabilities, since the system reliably retrieved relevant results for queries like "bus number 30".

Searching for a (generic) query, without the aim to find a certain video moment chosen in advance, yielded very good results ("emergency services", "red car with open doors", "dog", etc.). Interestingly, certain images tended to be retrieved for various related queries, but very hard to search for specifically (e.g., images of a red, first-generation Skoda Fabia were almost always retrieved when searching for a red car, but directly searching for it resulted in images of a different red Fabia of the second generation). Based on previous experimentation with the multimodal embedding models, CLIP seemed a bit more consistent and predictable than SigLIP. For example, it was better at differentiating between car types (SUV, hatchback, sedan, etc.), and for the query "jaywalking", it retrieved images of people crossing the street, while SigLIP retrieved images of people walking on the sidewalk.

Video analyses with GPT-4o yielded better results than the benchmark suggests. The model was able to describe various situations in great detail, identify various car models and other car attributes with great precision, read the entire front destination sign of a bus, and understand and describe the temporal actions that occur in a video segment (e.g., a person opening the door of a car, then leaving the car while carrying a bag). Conversing with the model was also very natural—for example, when prompted with "what is the most interesting event that happened in the video segment", the model could often infer what the user had in mind.

Railroad video from a train's perspective

The second experiment was proposed by the thesis' supervisor, who provided a single 63 minutes long video of a train journey along tracks somewhere in the Czech Republic, filmed from the locomotive's perspective (see Figure 7.6). The single video uploaded to the system was sampled at 2 FPS, and the experiment focuses on the fact that the train frequently passes various traffic signs, objects (such as vehicles), and train stations (or rather just train stops). Although the system was primarily designed for analyzing CCTV footage of traffic, this experiment demonstrates its potential applicability to adjacent domains that still involve transportation and vehicle movement.

Figure 7.6: Example frame sampled from the railroad video, showing the train passing the Řitonice train stop.

Selecting a specific moment (object) in the video and locating it through the embedding search was quite easy (e.g., "white van on the left", „teepee"—see figure 7.6), but only as long as the key component wasn't located on the side of the image, far away from its center. Searching for level crossings, railway signals, people standing next to train tracks, etc. worked seamlessly, but the performance related to traffic signs was rather bad (the system couldn't locate a stop sign). The biggest positive surprise was SigLIP's OCR performance, as the system was able to locate any of the train stops based on the name written on its nameplate (see "Řitonice" in figure 7.6).

Since the train moves relatively slowly, there's not much to analyze in most segments of the video. When analyzing the segment shown in figure 7.6, GPT-4o also demonstrated strong OCR capabilities by correctly reading the entire train stop name, and described the entire scene in great detail. In another segment, featuring people standing next to the train tracks and waving at the passing train, the model was able to hypothesize plausible reasons for their presence based on visual cues——for example, noting that some of the people were wearing high-visibility vests.

Chapter 8

Conclusion

This thesis explored the application of multimodal large language models (MLLMs) and related technologies (multimodal embedding models) for understanding and searching through traffic surveillance videos. The primary objective was to design, implement, and evaluate a system capable of efficiently retrieving relevant moments from a collection of videos based on free-form textual queries and interactively analyzing specified moments with the assistance of a video-MLLM.

The problem domain was explored and analyzed, providing insights that may guide future efforts to develop similar systems. The mechanisms behind both types of models were introduced, and the current state of their development was examined. The research of existing solutions involved an experimental evaluation of state-of-the-art models to determine which are best suited for the given use cases.

A two-step pipeline was proposed, delegating search functionality to an image-text multimodal embedding model and a vector database, while analysis is handled by a video-MLLM, and the system was implemented as a web application with an easy-to-use user interface. The system allows users to choose between four different models of both types, most of which were carefully selected and optimized to enable the system to run locally on modern hardware, and the system is also implemented in a way that makes it rather easy to add support for additional models.

The available models of both types were benchmarked in the domain of traffic footage, demonstrating the strengths and limitations of each model for specific use cases. The image-text retrieval performance of the multimodal embedding models was evaluated through retrieving relevant images of car models (and additionally, at least selecting the correct car brands) and the same was also done with photos of traffic signs and their categories, which were taken in the Czech Republic—this most probably had never been done before. The performance of the MLLMs was assessed via video question answering (VQA) with various traffic videos and corresponding questions, often requiring the models to perform complex reasoning. The system was also briefly evaluated with the best-performing models configured.

There are two obvious possible ways of extending this work in the future. Since the capabilities of state-of-the-art MLLMs rapidly expand on a monthly basis, the first way would be a future rework of the system with newer models, potentially delegating more work to the MLLM (such as analyzing entire videos or even handling the search process). The second way of extending the work would involve either creating or fine-tuning models of both types specifically for the domain of traffic footage, reducing the

gap between the generality of the utilized models and the specialized nature of the system.

Bibliography

[1] Alabdulmohsin, I.; Zhai, X.; Kolesnikov, A. and Beyer, L. Getting ViT in shape: scaling laws for compute-optimal model design. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2023. NIPS '23.

[2] Ataallah, K.; Shen, X.; Abdelrahman, E.; Sleiman, E.; Zhu, D. et al. *MiniGPT4-Video: Advancing Multimodal LLMs for Video Understanding with Interleaved Visual-Textual Tokens*. 2024. Available at: https://arxiv.org/abs/2404.03413.

[3] Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S. et al. *Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond*. 2023. Available at: https://arxiv.org/abs/2308.12966.

[4] Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W. et al. *Qwen2.5-VL Technical Report*. 2025. Available at: https://arxiv.org/abs/2502.13923.

[5] Bertasius, G.; Wang, H. and Torresani, L. *Is Space-Time Attention All You Need for Video Understanding?* 2021. Available at: https://arxiv.org/abs/2102.05095.

[6] Boesch, G. *Large Language Models – Technical Overview* Viso.ai. 18. July 2024. Available at: https://viso.ai/deep-learning/large-language-models/. Accessed: 2025-01-20.

[7] Carreira, J. and Zisserman, A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2017, p. 4724–4733. ISSN 1063-6919. Available at: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.502.

[8] Chen, W.-G.; Spiridonova, I.; Yang, J.; Gao, J. and Li, C. *LLaVA-Interactive: An All-in-One Demo for Image Chat, Segmentation, Generation and Editing*. 2023. Available at: https://arxiv.org/abs/2311.00571.

[9] Chen, X.; Wang, X.; Changpinyo, S.; Piergiovanni, A.; Padlewski, P. et al. *PaLI: A Jointly-Scaled Multilingual Language-Image Model*. 2023. Available at: https://arxiv.org/abs/2209.06794.

[10] Chen, Y.-C.; Li, L.; Yu, L.; El Kholy, A.; Ahmed, F. et al. UNITER: UNiversal Image-TExt Representation Learning. In: *Computer Vision – ECCV 2020: 16th*

European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX. Berlin, Heidelberg: Springer-Verlag, 2020, p. 104–120. ISBN 978-3-030-58576-1. Available at: https://doi.org/10.1007/978-3-030-58577-8_7.

[11] Cherti, M.; Beaumont, R.; Wightman, R.; Wortsman, M.; Ilharco, G. et al. Reproducible Scaling Laws for Contrastive Language-Image Learning. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, p. 2818–2829.

[12] Choi, H.; Zhu, E.; Bangash, A. and Miller, R. J. VISE: vehicle image search engine with traffic camera. *Proc. VLDB Endow.* VLDB Endowment, august 2019, vol. 12, no. 12, p. 1842–1845. ISSN 2150-8097. Available at: https://doi.org/10.14778/3352063.3352080.

[13] Dai, J.; Li, Y.; He, K. and Sun, J. R-FCN: object detection via region-based fully convolutional networks. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 379–387. NIPS'16. ISBN 9781510838819.

[14] Damjanovski, V. *CCTV: from light to pixels*. Elsevier, 2013. 549 p. ISBN 9780124045576.

[15] DeepLearningAI. *Generative AI with Large Language Models* https://www.coursera.org/learn/generative-ai-large-language-models. 2023. Coursera Course.

[16] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, p. 248–255.

[17] Devlin, J.; Chang, M.-W.; Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J.; Doran, C. and Solorio, T., ed. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, p. 4171–4186. Available at: https://aclanthology.org/N19-1423/.

[18] Dinh, Q.; Ho, M.; Dang, A. and Tran, H. TrafficVLM: A Controllable Visual Language Model for Traffic Video Captioning. In:. June 2024, p. 7134–7143.

[19] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. Available at: https://arxiv.org/abs/2010.11929.

[20] Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazirbaş, C. et al. FlowNet: Learning Optical Flow with Convolutional Networks. In:. December 2015, p. 2758–2766.

[21] Faysse, M.; Sibille, H.; Wu, T.; Omrani, B.; Viaud, G. et al. *ColPali: Efficient Document Retrieval with Vision Language Models*. 2025. Available at: https://arxiv.org/abs/2407.01449.

[22] Gemini Team; Georgiev, P.; Lei, V. I.; Burnell, R.; Bai, L. et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.* 2024. Available at: https://arxiv.org/abs/2403.05530.

[23] Girdhar, R.; El Nouby, A.; Liu, Z.; Singh, M.; Alwala, K. et al. ImageBind One Embedding Space to Bind Them All. In:. June 2023, p. 15180–15190.

[24] Girshick, R.; Donahue, J.; Darrell, T. and Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016, vol. 38, no. 1, p. 142–158.

[25] Gurney, J. *Optical Flow.* 2019. Available at: https://gurneyjourney.blogspot.com/2019/12/optical-flow.html. Accessed: 2024-11-28.

[26] Hadsell, R.; Chopra, S. and Lecun, Y. Dimensionality Reduction by Learning an Invariant Mapping. In:. February 2006, p. 1735 – 1742. ISBN 0-7695-2597-0.

[27] He, K.; Zhang, X.; Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016, p. 770–778.

[28] Hendricks, L. A.; Wang, O.; Shechtman, E.; Sivic, J.; Darrell, T. et al. Localizing Moments in Video with Temporal Language. In: Riloff, E.; Chiang, D.; Hockenmaier, J. and Tsujii, J., ed. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.* Brussels, Belgium: Association for Computational Linguistics, October-november 2018, p. 1380–1390. Available at: https://aclanthology.org/D18-1168/.

[29] Hradiš, M. *Language models and autoregressive factorization. Lecture Notes in Convolutional Neural Networks.* Brno University of Technology, Faculty of Information Technology. 13. April 2021.

[30] Jaiswal, V.; Sharma, V. and Varma, S. Comparative Analysis of CCTV Video Image Processing Techniques and Application: A Survey, october 2018, p. 38–47.

[31] Jeong, S.; Kim, K.; Baek, J. and Hwang, S. J. *VideoRAG: Retrieval-Augmented Generation over Video Corpus.* 2025. Available at: https://arxiv.org/abs/2501.05874.

[32] @Jesus. *Contextual Search Engine with LLM* Medium. 3. August 2023. Available at: https://medium.com/@jchavezar/contextual-search-engine-with-llm-e8e0641772d2. Accessed: 2025-02-17.

[33] Jia, C.; Yang, Y.; Xia, Y.; Chen, Y.; Parekh, Z. et al. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In: Meila, M. and Zhang, T., ed. *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event.* PMLR, 2021, vol. 139, p. 4904–4916. Proceedings of Machine Learning Research. Available at: http://proceedings.mlr.press/v139/jia21b.html.

[34] John, A. *Can CCTV Record Audio?* online. CCTV Camera Vision, 19. July 2024. Available at: https://cctvcameravision.com/can-cctv-record-audio/. Accessed: 2024-11-21.

[35] Johnson, J.; Douze, M. and Jégou, H. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 2021, vol. 7, no. 3, p. 535–547.

[36] Kiros, R.; Salakhutdinov, R. and Zemel, R. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *31st International Conference on Machine Learning, ICML 2014*, november 2014, vol. 3.

[37] Kolbe, J.; Noyan, M. et al. *Computer vision course* online. GitHub, 2024. Available at: https://github.com/johko/computer-vision-course.

[38] Kong, Q.; Kawana, Y.; Saini, R.; Kumar, A.; Pan, J. et al. WTS: A Pedestrian-Centric Traffic Video Dataset for Fine-Grained Spatial-Temporal Understanding. In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXVI*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 1–18. ISBN 978-3-031-73115-0. Available at: https://doi.org/10.1007/978-3-031-73116-7_1.

[39] Krause, J.; Stark, M.; Deng, J. and Fei Fei, L. 3D Object Representations for Fine-Grained Categorization. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia: [b.n.], 2013.

[40] Kumar, V. and Svensson, J. *Promoting social change and democracy through information technology*. IGI Global, 2015. 75 p. ISBN 9781466685031.

[41] Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A. et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Jurafsky, D.; Chai, J.; Schluter, N. and Tetreault, J., ed. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, p. 7871–7880. Available at: https://aclanthology.org/2020.acl-main.703/.

[42] Li, B.; Zhang, Y.; Guo, D.; Zhang, R.; Li, F. et al. *LLaVA-OneVision: Easy Visual Task Transfer*. 2024. Available at: https://arxiv.org/abs/2408.03326.

[43] Li, J.; Li, D.; Savarese, S. and Hoi, S. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In: *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023. ICML'23.

[44] Li, J.; Li, D.; Xiong, C. and Hoi, S. C. H. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In: *International Conference on Machine Learning*. 2022. Available at: https://api.semanticscholar.org/CorpusID:246411402.

[45] Li, L. H.; Yatskar, M.; Yin, D.; Hsieh, C.-J. and Chang, K.-W. *VisualBERT: A Simple and Performant Baseline for Vision and Language*. 2019. Available at: https://arxiv.org/abs/1908.03557.

[46] Li, Y.; Chen, X.; Hu, B.; Wang, L.; Shi, H. et al. *VideoVista: A Versatile Benchmark for Video Understanding and Reasoning*. 2024. Available at: https://arxiv.org/abs/2406.11303.

[47] Lin, B.; Ye, Y.; Zhu, B.; Cui, J.; Ning, M. et al. Video-LLaVA: Learning United Visual Representation by Alignment Before Projection. In: Al Onaizan, Y.; Bansal, M. and Chen, Y.-N., ed. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics, November 2024, p. 5971–5984. Available at: https://aclanthology.org/2024.emnlp-main.342/.

[48] Lin, J.; Tang, J.; Tang, H.; Yang, S.; Xiao, G. et al. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *GetMobile: Mobile Comp. and Comm.* New York, NY, USA: Association for Computing Machinery, january 2025, vol. 28, no. 4, p. 12–17. ISSN 2375-0529. Available at: https://doi.org/10.1145/3714983.3714987.

[49] Liu, H.; Li, C.; Li, Y.; Li, B.; Zhang, Y. et al. *LLaVA-NeXT: Improved reasoning, OCR, and world knowledge*. January 2024. Available at: https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[50] Liu, H.; Li, C.; Wu, Q. and Lee, Y. J. Visual Instruction Tuning. In: Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023, vol. 36, p. 34892–34916. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

[51] Liu, Q.; Kusner, M. J. and Blunsom, P. *A Survey on Contextual Embeddings*. 2020. Available at: https://arxiv.org/abs/2003.07278.

[52] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S. et al. SSD: Single Shot MultiBox Detector. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, p. 21–37. ISBN 9783319464480. Available at: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[53] Liu, Z.; Zhu, L.; Shi, B.; Zhang, Z.; Lou, Y. et al. *NVILA: Efficient Frontier Visual Language Models*. 2024. Available at: https://arxiv.org/abs/2412.04468.

[54] Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision. In: *IJCAI'81: 7th international joint conference on Artificial intelligence*. 1981, vol. 2, p. 674–679.

[55] Ma, Y.; Xu, G.; Sun, X.; Yan, M.; Zhang, J. et al. X-CLIP: End-to-End Multi-grained Contrastive Learning for Video-Text Retrieval. In: *Proceedings of the 30th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2022, p. 638–647. MM '22. ISBN 9781450392037. Available at: https://doi.org/10.1145/3503161.3547910.

[56] Malkov, Y. A. and Yashunin, D. A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* USA: IEEE Computer Society, april 2020, vol. 42, no. 4,

p. 824–836. ISSN 0162-8828. Available at: https://doi.org/10.1109/TPAMI.2018.2889473.

[57] Mao, J.; Qian, Y.; Ye, J.; Zhao, H. and Wang, Y. *GPT-Driver: Learning to Drive with GPT*. 2023. Available at: https://arxiv.org/abs/2310.01415.

[58] Martinský, O. *Recognition of vehicle number plates*. Brno, CZ, 2007. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.vut.cz/studenti/zav-prace/detail/15185.

[59] Meng, G.; Dai, T.; Zhang, L.; Zhu, J.; Wang, J. et al. *Multimodal LLM-guided Query Optimization for Visual-Language Retrieval*. 2024. Available at: https://openreview.net/forum?id=sjGmiI49sd.

[60] Mikolov, T.; Chen, K.; Corrado, G. S. and Dean, J. Efficient Estimation of Word Representations in Vector Space. In: *International Conference on Learning Representations*. 2013. Available at: https://api.semanticscholar.org/CorpusID:5959482.

[61] Mindy Support. *What is Image Segmentation: The Basics and Key Techniques* online. 12. October 2022. Available at: https://mindy-support.com/news-post/what-is-image-segmentation-the-basics-and-key-techniques/. Accessed: 2024-11-27.

[62] Ministerstvo dopravy. *Vyhláška č. 294/2015 Sb., kterou se provádějí pravidla provozu na pozemních komunikacích*. Zákony pro lidi, 2015. Available at: https://www.zakonyprolidi.cz/cs/2015-294. Accessed: April 10, 2025.

[63] OpenAI; :; Hurst, A.; Lerer, A.; Goucher, A. P. et al. *GPT-4o System Card*. 2024. Available at: https://arxiv.org/abs/2410.21276.

[64] OpenAI. *ChatGPT [Large language model], model GPT-4o*. 2024. Available at: https://chat.openai.com/.

[65] OpenAI; Applin, . S.; Adesso, G.; Ashfaq, R.; Bai, M. et al. GPT-4V(ision) System Card, january 2023. Available at: https://opal.latrobe.edu.au/articles/report/GPT-4V_ision_System_Card/25479208.

[66] Prabhat, P.; Gupta, H. and Vishwakarma, A. K. *Face Detection: Present State and Research Directions*. 2024. Available at: https://arxiv.org/abs/2402.03796.

[67] Prabu, S. and Gnanasekar, J. A Study on Image Segmentation Method for Image Processing. In:. December 2021. ISBN 9781643682167.

[68] Qi, D.; Su, L.; Song, J.; Cui, E.; Bharti, T. et al. *ImageBERT: Cross-modal Pre-training with Large-scale Weak-supervised Image-Text Data*. 2020. Available at: https://arxiv.org/abs/2001.07966.

[69] Qwen; :; Yang, A.; Yang, B.; Zhang, B. et al. *Qwen2.5 Technical Report*. 2025. Available at: https://arxiv.org/abs/2412.15115.

[70] R., M.; Utkarsh and J., V. Violence Detection from CCTV Footage Using Optical Flow and Deep Learning in Inconsistent Weather and Lighting Conditions. In:. October 2021, p. 638–647. ISBN 978-3-030-81461-8.

[71] Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G. et al. Learning Transferable Visual Models From Natural Language Supervision. In: *International Conference on Machine Learning*. 2021. Available at: https://api.semanticscholar.org/CorpusID:231591445.

[72] Radford, A. and Narasimhan, K. Improving Language Understanding by Generative Pre-Training. In:. 2018. Available at: https://api.semanticscholar.org/CorpusID:49313245.

[73] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S. et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* JMLR.org, january 2020, vol. 21, no. 1. ISSN 1532-4435.

[74] Raschka, S. *Understanding Multimodal Large Language Models (LLMs)* https://sebastianraschka.com/blog/2024/understanding-multimodal-llms.html. 3. November 2024. Accessed: 2025-01-22.

[75] Redmon, J.; Divvala, S.; Girshick, R. and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection . In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2016, p. 779–788. ISSN 1063-6919. Available at: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91.

[76] Regneri, M.; Rohrbach, M.; Wetzel, D.; Thater, S.; Schiele, B. et al. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2013, vol. 1, p. 25–36.

[77] Ren, S.; He, K.; Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M. and Garnett, R., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28. Available at: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[78] Ren, Y. and Li, Y. *On the Importance of Contrastive Loss in Multimodal Learning*. 2023. Available at: https://arxiv.org/abs/2304.03717.

[79] Saba, W. S. *LLMs' Understanding of Natural Language Revealed*. 2024. Available at: https://arxiv.org/abs/2407.19630.

[80] Senel, L. K.; Utlu, I.; Yucesoy, V.; Koc, A. and Cukur, T. Semantic Structure and Interpretability of Word Embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. Institute of Electrical and Electronics Engineers (IEEE), october 2018, vol. 26, no. 10, p. 1769–1779. ISSN 2329-9304. Available at: http://dx.doi.org/10.1109/TASLP.2018.2837384.

[81] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In:. Computational and Biological Learning Society, 2015, p. 1–14.

[82] Snoek, C. and Worring, M. Multimodal Video Indexing: A Review of the State-Of-The-Art. *Multimedia Tools and Applications*, january 2005, vol. 25, p. 5–35.

[83] Soldan, M.; Pardo, A.; Alcazar, J.; Heilbron, F.; Zhao, C. et al. MAD: A Scalable Dataset for Language Grounding in Videos from Movie Audio Descriptions. In: *Proceedings - 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*. United States: IEEE Computer Society, 2022, p. 5016–5025. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[84] Song, E.; Chai, W.; Wang, G.; Zhang, Y.; Zhou, H. et al. MovieChat: From Dense Token to Sparse Memory for Long Video Understanding. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, p. 18221–18232. Available at: https://api.semanticscholar.org/CorpusID:260333927.

[85] Tan, H. H. and Bansal, M. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In: *Conference on Empirical Methods in Natural Language Processing*. 2019. Available at: https://api.semanticscholar.org/CorpusID:201103729.

[86] Tan, M. and Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: Chaudhuri, K. and Salakhutdinov, R., ed. *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. PMLR, 2019, vol. 97, p. 6105–6114. Proceedings of Machine Learning Research. Available at: http://proceedings.mlr.press/v97/tan19a.html.

[87] Tang, X.; Qiu, J.; Xie, L.; Tian, Y.; Jiao, J. et al. *Adaptive Keyframe Sampling for Long Video Understanding*. 2025. Available at: https://arxiv.org/abs/2502.21271.

[88] Tang, Y.; Bi, J.; Xu, S.; Song, L.; Liang, S. et al. Video Understanding with Large Language Models: A Survey. *ArXiv preprint arXiv:2312.17432*, 2023.

[89] Trabelsi, E. Comprehensive Guide To Approximate Nearest Neighbors Algorithms. *Towards Data Science*, 2020. Available at: https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6.

[90] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L. et al. Attention is All you Need. In: Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[91] Wang, J.; Yi, X.; Guo, R.; Jin, H.; Xu, P. et al. Milvus: A Purpose-Built Vector Data Management System. In: *Proceedings of the 2021 International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2021, p. 2614–2627. SIGMOD '21. ISBN 9781450383431. Available at: https://doi.org/10.1145/3448016.3457550.

[92] Wang, P.; Bai, S.; Tan, S.; Wang, S.; Fan, Z. et al. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *ArXiv preprint arXiv:2409.12191*, 2024.

[93] Wang, S.; Anastasiu, D. C.; Tang, Z.; Chang, M.; Yao, Y. et al. The 8th AI City Challenge. *CoRR*, 2024, abs/2404.09432. Available at: https://doi.org/10.48550/arXiv.2404.09432.

[94] Wang, T.; Li, F.; Zhu, L.; Li, J.; Zhang, Z. et al. *Cross-Modal Retrieval: A Systematic Review of Methods and Future Directions*. 2024. Available at: https://arxiv.org/abs/2308.14263.

[95] Wang, X.; Zhang, Y.; Zohar, O. and Yeung Levy, S. VideoAgent: Long-Form Video Understanding with Large Language Model as Agent. In:. October 2024, p. 58–76. ISBN 978-3-031-72988-1.

[96] Watson, A. *Deep Learning Techniques for Super-Resolution in Video Games*. 2020. Available at: https://arxiv.org/abs/2012.09810.

[97] Wikipedia contributors. *Road signs in the Czech Republic* Wikipedia, The Free Encyclopedia. 2025. Available at: https://en.wikipedia.org/wiki/Road_signs_in_the_Czech_Republic. Accessed: April 10, 2025.

[98] Wu, J.; Gan, W.; Chen, Z.; Shicheng, W. and Yu, P. Multimodal Large Language Models: A Survey. In:. November 2023.

[99] Wu, Y.; Li, B.; Cao, J.; Zhu, W.; Lu, Y. et al. *Zero-Shot Long-Form Video Understanding through Screenplay*. 2024. Available at: https://arxiv.org/abs/2406.17309.

[100] Xie, J.; Chen, Z.; Zhang, R.; Wan, X. and Li, G. *Large Multimodal Agents: A Survey*. 2024. Available at: https://arxiv.org/abs/2402.15116.

[101] Xu, L.; Huang, H. and Liu, J. SUTD-TrafficQA: A Question Answering Benchmark and an Efficient Network for Video Reasoning Over Traffic Events. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, p. 9878–9888.

[102] Xu, Z. *Dimensionality of Word Embeddings* Baeldung. 18. march 2024. Available at: https://www.baeldung.com/cs/dimensionality-word-embeddings. Reviewed by Michal Aibin. Accessed: January 14, 2025.

[103] Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B. et al. *Qwen2 Technical Report*. 2024. Available at: https://arxiv.org/abs/2407.10671.

[104] Yawale, N.; Sahu, N. and Khalsa, N. Design of a high-density bio-inspired feature analysis deep learning model for sub-classification of natural & synthetic imagery. *Multimedia Tools and Applications*, august 2023, vol. 83, p. 1–26.

[105] Yu, C.-C.; Wen, M.-G.; Fan, K.-C. and Lue, H.-T. Preprocessing for Images Captured by Cameras. In: Ding, X., ed. *Advances in Character Recognition*. Rijeka: IntechOpen, 2012, chap. 5. Available at: https://doi.org/10.5772/52110.

[106] Zargar, S. *Introduction to Sequence Learning Models: RNN, LSTM, GRU*. April 2021.

[107] Zhai, X.; Mustafa, B.; Kolesnikov, A. and Beyer, L. Sigmoid Loss for Language Image Pre-Training . In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, October 2023, p. 11941–11952. Available at: https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01100.

[108] Zhang, B.; Li, K.; Cheng, Z.; Hu, Z.; Yuan, Y. et al. *VideoLLaMA 3: Frontier Multimodal Foundation Models for Image and Video Understanding*. 2025. Available at: https://arxiv.org/abs/2501.13106.

[109] Zhang, H.; Li, X. and Bing, L. Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding. In: Feng, Y. and Lefever, E., ed. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Singapore: Association for Computational Linguistics, December 2023, p. 543–553. Available at: https://aclanthology.org/2023.emnlp-demo.49/.

[110] Zhang, H.; Sun, A.; Jing, W. and Zhou, J. T. Temporal Sentence Grounding in Videos: A Survey and Future Directions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Institute of Electrical and Electronics Engineers (IEEE), august 2023, vol. 45, no. 8, p. 10443–10465. ISSN 1939-3539. Available at: http://dx.doi.org/10.1109/TPAMI.2023.3258628.

[111] Zhang, L.; Li, P. and Wang, B. A survey of multimodal retrieval: from traditional analytics to deep learning. *Proceedings of SPIE*, 2023, vol. 13214, p. 3033335. Available at: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13214/3033335/A-survey-of-multimodal-retrieval--from-traditional-analytics-to/10.1117/12.3033335.full.

[112] Zhang, L.; Zhao, T.; Ying, H.; Ma, Y. and Lee, K. OmAgent: A Multi-modal Agent Framework for Complex Video Understanding with Task Divide-and-Conquer. In: Al Onaizan, Y.; Bansal, M. and Chen, Y.-N., ed. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics, November 2024, p. 10031–10045. Available at: https://aclanthology.org/2024.emnlp-main.559/.

Appendix A

Contents of the external attachment

- `/app` – The application (the implemented system), the folder contains its own `README.md`.
- `/model_bechmarks` – The benchmarks of the available AI models, the folder contains its own `README.md`.
- `/text` – The LaTeX source code of the thesis' text (including figures, etc.).
- `/excel_at_fit` – Materials used to present the work at the Excel@FIT conference (poster, short paper, video, preview).
- `README.md` – The main README file.
- `thesis.pdf` – The text (technical report) of the thesis in PDF format.

Appendix B

Poster