Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Integrating Text and Image Models for Question Answering

Master thesis

Bc. Jan Čuhel

Study programme: Open Informatics
Branch of study: Computer Vision and Image Processing
Supervisor: Ing. Jan Šedivý, CSc.

Prague, May 2024

Thesis Supervisor:
Ing. Jan Šedivý, CSc.
Czech Institute of Informatics, Robotics and Cybernetics (CIIRC)
Czech Technical University in Prague
Jugoslávských partyzánů 1580
160 00 Prague 6
Czech Republic

Copyright © May 2024 Bc. Jan Čuhel

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Čuhel Jan** Personal ID number: **483634**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Cybernetics**
Study program: **Open Informatics**
Specialisation: **Computer Vision and Image Processing**

II. Master’s thesis details

Master’s thesis title in English:

Integrating Text and Image Models for Question Answering

Master’s thesis title in Czech:

Integrace textových a obrazových modelů pro odpovídání na otázky

Guidelines:

This assignment is to develop algorithms for a system that simplifies the use of user's manuals for service and maintenance of complicated machines.The technical user’s manuals for complicated machines are usually in PDF format and contain both textual information and images that complement the explanation. Explore methods for answering questions from technical manuals, with a specific focus on Retrieval-Augmented Generation (RAG). The task involves finding the most effective combination of Large Language Models (LLMs) for generating accurate textual answers along with relevant descriptive images. Study and review the state-of-the-art models and propose an optimal fusion of text and image models. Subsequently, select models and design and implement a working system. Select benchmarks to estimate accuracy, with special attention to alignment of textual workflows and tags in images, ensuring comprehensive understanding and accurate representation.

Bibliography / sources:

[1] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry et al. "Learning transferable visual models from natural language supervision." In International conference on machine learning, pp. 8748-8763. PMLR, 2021.
[2] Li, Junnan, Dongxu Li, Silvio Savarese, and Steven Hoi. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models." arXiv preprint arXiv:2301.12597 (2023).
[3] Chen, Jun, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. "Minigpt-v2: large language model as a unified interface for vision-language multi-task learning." arXiv preprint arXiv:2310.09478 (2023).
[4] Liu, Haotian, Chunyuan Li, Yuheng Li, and Yong Jae Lee. "Improved baselines with visual instruction tuning." arXiv preprint arXiv:2310.03744 (2023).
[5] Wang, Jianfeng, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. "Git: A generative image-to-text transformer for vision and language." arXiv preprint arXiv:2205.14100 (2022)
[6] Wang, Weihan, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji et al. "Cogvlm: Visual expert for pretrained language models." arXiv preprint arXiv:2311.03079 (2023).

Name and workplace of master’s thesis supervisor:

Ing. Jan Šedivý, CSc. Big Data and Cloud Computing CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **03.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Šedivý, CSc.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt

Student’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date... ....
Bc. Jan Čuhel

Declaration of Artificial Intelligence Usage

I hereby declare that while working on this thesis, I have used the artificial intelligence tools as an auxiliary tool while writing the report and implementation in the form of GitHub Copilot[1] and ChatGPT/GPT–4[2] for the following types of tasks; generating code for LaTeX, small text reformulations, generating docstrings of functions and methods or generating auxiliary functions (e.g. to draw a plot). The generated outputs were used as a starting point and edited (if needed) for this work.

---

[1] `https://github.com/features/copilot`

[2] `https://chatgpt.com/`

Abstract

Abstract (en)

This work focuses on enhancing Retrieval–Augmented Generation (RAG) systems by incorporating knowledge from images. The goal of this work was to find an optimal fusion of text and image models to integrate knowledge from images into Question–Answering (QA) systems capable of answering questions about technical manuals. The work introduces the problem and reviews the related work and the current state–of–the–art Visual–Language models (VLMs). We then propose a multi–modal RAG pipeline and describe its crucial components. We created several data sets to train and evaluate the proposed pipeline's components. The created data sets and trained models were published. The selected models show promising results on the created data sets. As part of the work, we proposed an algorithm for localizing an icon's position in a text based on its coordinates on a page. Ultimately, we implemented the proposed multi–modal RAG pipeline into a working QA application. All experiments and models were programmed in the Python programming language.

Keywords: Computer Vision (CV), Natural Language Processing (NLP), Large Language Models (LLMs), Vision–Language Models (VLMs), Retrieval–Augmented Generation (RAG), Question–Answering (QA).

Abstrakt (cs)

Tato práce se zaměřuje na vylepšení systémů Retrieval–Augmented Generation (RAG) začleněním znalostí z obrázků. Cílem této práce bylo najít optimální fúzi textových a obrazových modelů pro začlenění znalostí z obrázků do systémů pro zodpovídání otázek (QA), které jsou schopny odpovídat na otázky týkající se technických příruček. Práce začíná představením problému a podáním přehledu o souvisejících pracích a současně nejlepších vizuálně jazykových modelů (Visual–Language models, VLM). Poté je navržen multimodální RAG systém a jsou popsány jeho klíčové komponenty. Pro natrénování a ohodnocení jednotlivých komponent jsme vytvořili několik datových sad. Tyto datové sady a natrénované modely byly zveřejněny. Vybrané modely vykazují na vytvořených datových sadách slibné výsledky. V rámci práce jsme také navrhli algoritmus pro lokalizaci pozice ikony v textu na základě jejích souřadnic na stránce. Nakonec jsme implementovali navržený multimodální RAG systém do funkční aplikace. Všechny experimenty a modely byly naprogramovány v programovacím jazyce Python.

Klíčová slova: Počítačové vidění, Zpracování přirozeného jazyka, Velké jazykové modely, Vizuálně–jazykové modely, Rozšířená generace pomocí vyhledávání, Problém odpovídání otázek.

Acknowledgements

First, I would like to express my enormous gratitude to my thesis supervisor, Ing. Jan Šedivý, CSc. He has been a constant source of encouragement and insight during my research and helped me with numerous problems. Furthermore, I am grateful to Bc. Adam Černý and Ing. Ondřej Kobza for help with creating the icon captioning dataset. Special thanks go to the staff of the Department of Cybernetics, CIIRC, and Team Alquist, who gave me access to the GPT–4 model and maintained a pleasant and flexible environment for my research. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

Contents

Abstract vii

Acknowledgements viii

List of Tables xi

List of Figures xiii

List of Acronyms xiv

1 Introduction 1
- 1.1 Structure of the thesis 2
- 1.2 Problem Description 2

2 Related Work 4
- 2.1 Theoretical Background 4
- 2.2 Vision–Language Models (VLMs) 6
  - 2.2.1 State–of–the–art VLMs 6
  - 2.2.2 Evaluating VLMs 9
- 2.3 Retrieval–Augmented Generation (RAG) 14
  - 2.3.1 RAG components 15
  - 2.3.2 Multi–modal RAG 16

3 Our Approach 17
- 3.1 Proposed RAG Pipeline 17
  - 3.1.1 Pre–Processing Phase 17
  - 3.1.2 Inference Phase 19
  - 3.1.3 Extensions 20
- 3.2 Data 21
- 3.3 Created Data Sets 23
  - 3.3.1 Image Text Relevance Data Set 23
  - 3.3.2 Icon Captioning Data Set 29
  - 3.3.3 Plot Recognition Data Set 31

4 Results and Discussion 32
- 4.1 Used Libraries and Tools 32
- 4.2 Used Hardware 33
- 4.3 Used Models 33
- 4.4 Image Text Relevance Task 33

4.4.1 Baseline Results 33
4.4.2 Using LLM to shorten the text 35
4.4.3 Fine–tuning of CLIP and SigLIP 36
4.4.4 Fine–tuning of Idefics2 38
4.4.5 Final Results 38
4.5 Icon Captioning 39
4.6 Icon Localization 40
4.7 Plots 40
4.7.1 Plots Recognition 40
4.7.2 Plot to Text Translation 41
4.8 Multi–modal RAG Pipeline Implementation 41

5 Conclusion 44
5.1 Future Work 44

A Model Overview 46

B Image Text Relevance Task 47
B.1 Semi–automatically Curated Data Set: Utilized Questions 47
B.2 Utilized Prompts 49
B.3 Fine–tuning CLIP 50
B.4 Fine–tuning SigLIP 50
B.5 Fine–tuning Idefics2 50

C Icon Captioning Task 52
C.1 Utilized Prompts 52

Bibliography 57

List of Tables

2.1 A table with results of state–of–the–art VLMs. (*: model was trained on this task) . . . 13

3.1 Analysis of the selected manuals . . . 22
3.2 Analysis of the resolutions of the images from the selected manuals . . . 23
3.3 Analysis of the semi-automatically curated image–text–relevance data set . . . 24
3.4 Analysis of the manually curated image–text–relevance data set . . . 25
3.5 Sizes of each data set split . . . 25
3.6 Analysis of the icon captioning data set . . . 30

4.1 Used hardware . . . 33
4.2 Utilized model variants, their size and the tasks they were utilized for (*ITR*: stands for image–text–relevance task, *IC*: stands for icon captioning, *PD*: stands for plot detection, and *PT*: stands for plot translation, the names in the parentheses represent the tasks onto which the models were fine–tuned, *patch**X/Y*** means that the given model was pre–trained on images with $Y \times Y$ resolution and it splits input images into patches of size $X \times X$) . . . 34
4.3 A table showing the results of the baseline VLMs on the image–text–relevance data set (for *CLIP* and *SigLIP* models, the number in the parentheses in the column *mode* represents the value of the automatically found threshold) . . . 35
4.4 Comparison of the CLIP results when using summaries vs. not (please note that *CLIP B/32* refers to *CLIP ViT–B/32*, and *CLIP B/16* refers to *CLIP ViT–B/16*) 36
4.5 Comparison of the SigLIP results when using summaries and when not (please note that *siglip-so-p14* refers to *SigLIP so400m (patch14/384)* and *SigLIP L/16* to *SigLIP large (patch16/384)*) . . . 37
4.6 Table showing the contribution of training the CLIP models in comparison with the non–trained versions (please note that +T stands for training, *CLIP B/32* refers to *CLIP ViT–B/32*, and *CLIP B/16* refers to *CLIP ViT–B/16*) . . . 37
4.7 Table showing the contribution of training the CLIP models in comparison with the non–trained versions (please note that +T stands for training, and that *siglip-so-p14* refers to *SigLIP so400m (patch14/384)* and *SigLIP L/16* to *SigLIP large (patch16/384)*) . . . 37
4.8 A table showing the contribution of training the Idefics2 model in comparison with the non–trained version . . . 38
4.9 A table showing the final results on the image–text–relevance data set . . . 39
4.10 A table showing the results on the icon captioning data set (*BLEU N* means that the overlap is computed across geven N-grams, e.g. $N = 2$ means bigrams are considerated) . . . 39
4.11 Results of the icon localization algorithm1 . . . 40

4.12 Model results for plot recognition 41

A.1 Used pre–trained models, link to their weights and their latency on a batch with a single sample measured across around a hundred samples (except for the DePlot model) on a device with one NVIDIA Tesla A100 GPU with 40GB of memory 46

B.1 Hyperparameters used for training the CLIP models 50
B.2 Hyperparameters used for training the SigLIP models 51
B.3 Hyperparameters used for training the Idefics2 model 51

List of Figures

2.1 An example from the paper [41] of the *Flickr30K* data set 10
2.2 Two examples from the paper [42] of the *NoCaps* data set. Each image is annotated with 10 reference captions 11
2.3 Two examples from the data set website [45] of the *VQAv2* data set 11
2.4 Examples from the data set website [46] of the *VizWiz* VQA data set 12
2.5 The Vision Arena leaderboard 13
2.6 The OpenVLM leaderboard 14

3.1 Pre–processing phase of the proposed multi–modal RAG pipeline 18
3.2 The potential icon problem 19
3.3 Inference phase of the proposed multi–modal RAG pipeline 19
3.4 Updated pre–processing phase of the proposed Multi–modal RAG pipeline with plots 21
3.5 Histogram of the 25 most common words in the selected manuals 22
3.6 Example of the created image–text–relevance data set with the following question: *"How do I turn on the cruise control?"* and the following answer: *"To turn on cruise control, press the R button on the control stalk. Then, select CC using the rotary knob on the steering wheel and press to confirm. Cruise control is switched on and passive.The operating status appears grey in the status display. There is no speed stored."*. 24
3.7 Label frequencies of the combined image–text-relevance data set 26
3.8 Icon captioning data set preview (only showing the icons' function and their apperance) 30
3.9 Examples from the curated plot recognition data set 31

4.1 Example of a plot with a correct translation that is following: *TITLE | <0x0A> Emotions | Number of samples <0x0A> Anger | 1079 <0x0A> Disgust | 1057 <0x0A> Fear | 1080 <0x0A> Joy | 1084 <0x0A> Sadness | 1078 <0x0A> Shame | 1057 <0x0A> Guilt | 1045* 42

List of Acronyms

AI Artificial intelligence. 1

CV Computer vision. 1

GIT Generative Image–to–text Transformer. 7

IR Information retrieval. 15

LLMs Large Language models. 1
LMs Language models. 1
LoRA Low–Rank Adaptation. 8
LSTM Long Short–Term Memory. 5

MAP mean average precision. 15
MRR mean reciprocal rank. 15

NLP Natural language processing. 1

PEFT Parameter Efficient Fine–tuning method. 8

QA Question–Answering. 2

RAG Retrieval–Augmented Generation. 1
RNNs Recurrent neural networks. 5

tf–idf Term Frequency-Inverse Document Frequency. 11

ViT Vision Transformer. 5
VLMs Visual–Language models. 2
VQA Visual Question Answering. 6

Chapter 1

Introduction

In recent years, Artificial intelligence (AI) has witnessed remarkable advancements, particularly in the domains of Natural language processing (NLP) and Computer vision (CV), with the introduction of Self–Attention and Transformers architecture in the paper *Attention is all you need, Vaswani, Ashish, et al.* [1]. The Transformer architecture became very popular in NLP and CV, and it is nowadays used to solve various NLP and CV problems. One notable area of research that has garnered significant popularity is multi–modal AI, which takes the best of what each modal field has to offer. Throughout the years, as the Language models (LMs) powered by the Transformer architecture kept growing, we started to talk about them as Large Language models (LLMs). However, even though the LLMs were getting more impressive and capable of storing much more pieces of information about the world, they still suffered from several issues, such as generating responses with outdated knowledge or responses that were factually incorrect, a phenomenon called hallucination [2]. One solution to reduce hallucination is to use a Retrieval–Augmented Generation (RAG) [3] technique. RAG attempts to reduce hallucination by fetching knowledge from external sources (e.g., web searches and manuals) and utilizing this fetched knowledge during the process of generating a response. This ensures that the response contains factual knowledge and decreases the possibility of hallucinating. Despite the popularity of the multi–modal AI, little work is being done in the domain of multi–modal RAG. This thesis focuses on enhancing RAG systems dealing with technical manuals by integrating images into the process. Typically, these manuals are presented in PDF format and are rich with illustrative images. Employing the knowledge from these images could, therefore, boost the performance of RAG systems.

The core objective of this thesis is to explore and develop algorithms that enhance RAG systems by incorporating images. The main focus is on finding relevant images for a generated answer and the input question and utilizing icons. This thesis will contribute to the field by providing a detailed analysis and practical solution to a real-world problem, potentially setting a new standard for how technical manuals are interacted with in the industry. This thesis can also be considered as sort of an overview of the current state–of–the–art Visual–Language models (VLMs).

1.1 Structure of the thesis

This thesis is organized into x chapters as follows:

1. *Introduction*: Explain the motivation behind our efforts, set the goals, and describe the problem.
2. *Related Work*: Introduces the necessary theoretical background and state–of–the-art related work to the reader.
3. *Our Approach*: Describes our approach to achieving the set goals.
4. *Results and Discussion*: Describes the experiments that we have conducted and shows their results.
5. *Conclusion*: Summarizes the thesis results, suggests possible topics for further research, and concludes the work.

1.2 Problem Description

Throughout history, humanity has invented many astonishing inventions that have helped us prolong and improve the quality of our lives, saved us time by automating some routine tasks, or enabled us to connect with people more easily and travel not only around the globe but also into the space and so much more. We are living in truly technologically unique times. However, as the devices we use get more and more advanced, they also get more complex. This increased the required knowledge for us to operate these devices successfully. Fortunately, manuals come to the rescue.

Manuals contain valuable pieces of information that help people understand how to use and operate various devices and products. They contain product descriptions, instructions on how to use them and repair if necessary. They also warn about possible dangers that may arise from the devices' usage. These manuals come in the form of a structured text document (usually a PDF file). The textual instructions are usually accompanied by images that further explain some procedure or help point to some feature. There is a famous saying: ”A picture is worth a thousand words,” which translates to the fact that sometimes images can explain some idea better than a thousand words. However, in today's classical Question–Answering (QA) applications, images pose a considerable challenge that many QA systems solve by ignoring them and only working with the textual modality. In this work, we will utilize images in a QA application focused on user manuals for the service and maintenance of complicated machines such as vehicles.

Our problem is notably straightforward; we are given a manual as a PDF file(s). Our task is, given a user question, to find relevant images to the topic of the question and the corresponding answer. We assume that the input manuals are in English.

Chapter 2

Related Work

2.1 Theoretical Background

These days, machine learning models in the NLP domain are standardly based on the Transformer architecture [1]. CV models have been using Convolutional layers [4] since the arrival of *AlexNet* [5], but nowadays, they are also utilizing Transformer layers. The following passages will briefly go through some basic necessary concepts.

Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNS) [4] is a specific type of neural network that was inspired by the visual Cortex in the human brain. CNNs are generally composed of convolution, pooling, and fully connected layers. Each convolutional layer performs an operation called convolution, which represents an application of a filter kernel to an input matrix using a sliding window mechanism. Each convolutional layer contains a set of these filter kernels. Each filter is trained to recognize a specific pattern in the image, such as an edge [6].

Encoder Decoder

The encoder and decoder are two components of so-called Encoder–decoder networks, sometimes called sequence–to–sequence networks. Their principle is simple: the encoder accepts an input sequence and returns a corresponding contextual representation of the input sequence. In other words, the encoder takes input information and encodes it into a vector representation. A decoder accepts a contextual representation and generates an output sequence specific for a specified task [2]. These two components can stand apart and be used separately, which is commonly done.

Language Models (LMs)

Language models, often abbreviated as LMs, assign probabilities to forthcoming words or sequences of words based on a specified context. The *n–gram* LM is an example of a straightforward LM. An *n–gram* is just a sequence of n words [2].

Large Language Models (LLMs)

The term Large language models (LLMs) refers to scaled–up language models with billions of parameters such as Google's *PaLM* [7] with 540 billion parameters or OpenAI's *GPT–3* [8] with 175 billion parameters. Please note that the perception of what is 'large' shifts throughout time.

Transformer Architecture

Transformers [1] are a deep learning architecture that powers the most recent NLP solutions. They are the successors of *Recurrent neural networks (RNNs)* [9] and *Long Short–Term Memory (LSTM)* [10]. Transformer–based models are made out of transformer blocks. Each block receives a sequence of vectors $(\mathbf{x}_1, ..., \mathbf{x}_n)$ and outputs a sequence of vectors $(\mathbf{z}_1, ..., \mathbf{z}_n)$ of the same length [2]. Each block consists of feed–forward networks and multi–head attention layers. Multi–head attention layer is just a combination of many self-attention mechanisms, which is the key innovation of the Transformer architecture. The intuition behind self–attention is finding relationships within the vectors of a sequence and their proper weighting.

LMs based on Transformer architecture are trained in a self–supervised fashion, either by Masked generation [11] or Next token generation (Causal language modeling) [12] training objectives. In the Masked generation objective, a specified number of randomly chosen input tokens are masked, and the task for the model is to unmask them (predict the probability of masked tokens). In the Next token generation, the model iteratively predicts the next token conditioned on the previous tokens. Such models are called autoregressive. The difference between the Masked generation and the Next token generation is that in the Masked generation, the model also has access to the future tokens that follow a specific token. The conditioning is bi-directional, whereas the Next token generation model has only access to the previous tokens.

Vision Transformer (ViT)

Transformer architecture was initially invented for the NLP domain; however, in work *An image is worth 16x16 words: Transformers for image recognition at scale, Dosovitskiy, Alexey, et al.* [13] it was successfully adapted to the CV domain. The authors in this paper presented the *Vision Transformer (ViT)* model, a pure transformer–based classification model. The model works in the following way: first, an image is split into fixed–sized patches, flattened, and linearly mapped into an embedding space. Position embeddings are added to the sequence alongside a *classification token*. This updated sequence is fed into the Transformer encoder, to which a classification head is attached.

2.2 Vision–Language Models (VLMs)

Vision–Language models (VLMs) are machine learning models that combine the vision and language modalities. In other words, they are capable of ingesting images and natural texts. The exact details depend on the specific architecture and the task these models are trained on. VLMs solve tasks such as image captioning, image retrieval, Visual Question Answering (VQA), and image classification. Image retrieval is a traditional CV task that involves finding similar images to a given query from a database of images [14]. In VQA, models are given an image and a question and are tasked to answer the question using the knowledge from the image. In image classification, VLMs classify input images based on provided textual prompts [15].

2.2.1 State–of–the–art VLMs

In the following subsections, we will introduce some of the state–of–the–art VLMs.

CLIP

The *CLIP* model [16], an acronym for Contrastive Language–Image pre–training by OpenAI, represents the first significant advancement in multi–modal image–text understanding. Its primary objective is to assess the degree of similarity between a provided image and a provided textual caption. During the training process, *CLIP* strives to optimize the cosine similarity metric between pairs of vectors that correspond to correctly associated image–caption pairs. Simultaneously, it minimizes the similarity scores between vectors representing incorrect associations. This is done using the *CLIP loss* (equation 2.1). During inference, *CLIP* leverages its trained model to compute similarity scores between the vector representation of a single image and a collection of potential caption vectors. Subsequently, it selects the caption with the highest similarity score, indicating the most contextually relevant textual description for the particular image. *CLIP* model is widely used in Image retrieval.

The ***CLIP loss***, incorporating the definition of cross–entropy loss, can be expressed as:

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^{N} \left(\log \frac{e^{t\mathbf{x}_i \cdot \mathbf{y}_i}}{\sum_{j=1}^{N} e^{t\mathbf{x}_i \cdot \mathbf{y}_j}} + \log \frac{e^{t\mathbf{x}_i \cdot \mathbf{y}_i}}{\sum_{j=1}^{N} e^{t\mathbf{x}_j \cdot \mathbf{y}_i})} \right) \tag{2.1}$$

where t are learnable parameters, $\mathbf{x}_i$ normalized image features, $\mathbf{y}_j$ normalized text features, and N is the size of a particular mini–batch.

SigLIP

The *SigLIP* model [17] is another multi–modal embedding model. The paper's authors propose replacing the *CLIP loss* with a simple pairwise sigmoid loss (equation 2.2). In contrast to the *CLIP loss*, the proposed sigmoid loss does not require global normalization factors. It processes each image–text pair independently, resulting in a more efficient parallel implementation than *CLIP loss*.

The ***Sigmoid loss*** is defined as

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\log\frac{1}{1+e^{z_{ij}(tx_i\cdot y_j+b)}} \tag{2.2}$$

where b and t are learnable parameters, $\mathbf{x}_i$ normalized image features, $\mathbf{y}_j$ normalized text features, N is the size of a specified mini–batch, and z_{ij} is defined as:

$$z_{ij} = \begin{cases} 1, & \text{for positive pairs} \\ -1, & \text{for negative pairs.} \end{cases}$$

GIT

In work *Git: A generative image-to-text transformer for vision and language, Wang, Jianfeng, et al.* [18], the authors present an image–to–text *Generative Image–to–text Transformer (GIT)* model consisting of a pre–trained image encoder and transformer–based text decoder to generate text from images. The image encoder transforms raw images into a compact 2D feature map, subsequently projected into D dimensions through an additional linear layer and a normalization layer. The text decoder employs a Transformer module for text description prediction, consisting of multiple Transformer blocks. Text is tokenized and embedded into D dimensions. Image features and text embeddings are concatenated and serve as input to the Transformer module. Text generation commences with the $[BOS]$ token, proceeding autoregressively until the $[EOS]$ token or maximum steps are reached.

BLIP–2

BLIP–2 [19] introduces a method that leverages frozen vision and language models to efficiently bridge the modality gap between visual and textual features. It minimizes computational requirements by training fewer parameters than prior end–to–end approaches. A proposed, lightweight Querying Transformer (or *Q–Former*) acts as an intermediary between the frozen image and text encoders. Visual features are extracted by the image encoder and then processed by the language model for comprehension. For the frozen image encoder, the authors explored two pre–trained Vision Transformer models: *ViT–L/14* from *CLIP* [16] and *ViT–g/14* from *EVA–CLIP* [20]. For the frozen language model, the authors experimented with the unsupervised–trained *OPT* model family [21] and the instruction–trained *FlanT5* model family [22] LLM.

InstructBLIP

InstructBLIP [23] is an improved version of *BLIP–2*. Specifically, it is an instruction–tuned version of *BLIP–2*. The authors trained the *InstructBLIP* model in three stages (the first two are identical as in *BLIP–2*, and the third is the instruction–tuning). As a frozen LM backbone, the authors decided to use instruction–trained *FlanT5* model and *Vicuna* [24][1] (a decoder–only LLM finetuned from *LLaMA* [25]).

MiniGPT4

The authors of *MiniGPT4* [26] were inspired by the *BLIP–2* model and employed the same pre–trained vision components of *BLIP–2* (*ViT–G/14* from *EVA–CLIP* and a *Q–Former* network). As an LLM, they used *Vicuna*. *MiniGPT4* adds a single projection layer trained in two–stage settings to align the encoded visual features with the Vicuna language model. The rest of the parameters are frozen.

MiniGPT–v2 [27] is built upon *MiniGPT4* model. *MiniGPT–v2* aims to build a unified interface for completing many vision–language tasks, including image description, VQA, and visual grounding, using intensive multi–modal instruction tuning. *MiniGPT–v2* adopts the open–sourced *LLaMA2-chat* (7B) [25] as the LM backbone that is further fine–tuned using *Low–Rank Adaptation (LoRA)* [28], one of the *Parameter Efficient Fine–tuning method (PEFT)*. *LoRA* works on the principle of updating the weights of carefully selected layers while keeping the original model frozen. The weights of the selected layers to be updated are cloned and initialized to zeros. The key idea is that the weight matrices are not updated using full rank but by product of two matrices of lower rank.

LLaVA

LLaVA [29], *LLaVA–1.5* [30], and *LLaVA–NeXT* [31] are similar to *MiniGPT4* as they are using *Vicuna* as an LLM and pre–trained *CLIP* visual encoder *ViT–L/14*. The critical difference is that instead of using *Q–Former* from *BLIP–2*, *LLaVA* uses a linear projection, and *LLaVA–1.5* uses a two–layer MLP [32], [33]. *LLaVA-NeXT* further builds upon *LLaVA–1.5* by increasing the

---

[1] `https://github.com/lm-sys/FastChat`

input image resolution, improving the data sets' variability, and scaling up the LLM backbone. In addition to *Vicuna 1.5* (in 7 and 13 billion variants), the authors also considered *Mistral 7B* [34] and *Nous-Hermes–2–Yi–34B* [35] LLMs.

Idefics2

In work *What matters when building vision-language models?, Laurençon, Hugo, et al.* [36], the authors sought to find out, as the name suggests, which aspects matter when creating a VLM. Based on their findings, they built the *Idefics2* model, a VLM similar to *LLaVA*. The model takes an input image and feeds it into a vision encoder (*SigLIP–SO400M* [17]). The outputted visual features are then mapped to the LM input space to obtain visual tokens. Visual tokens are then concatenated with the input text embedding sequence. Finally, the resulting sequence of vectors is passed into the LM (*Mistral 7B*) that predicts the textual output.

DePlot

Reasoning about plots and charts is a challenging task. The authors of *DePlot* [37] divided this challenge of visual language reasoning into two steps: plot–to–text translation using the proposed *DePlot* model and reasoning over the translated text using a LLM (*FlanPaLM* [38]). *DePlot* is a image–to–text encoder–decoder Transformer model based on *MatCha* [39] VLM trained to map plots to their corresponding linearized data tables.

GPT–4

OpenAI released in March 2023 its multi–modal LM, *GPT–4* [40], which can take images and other modalities as an input with a textual instruction describing the task. In May 2024 an updated version *GPT–4o*[2] ('o' for 'omni') was released. It is the OpenAI's latest multi–modal model capable of reasoning across audio, text, and vision. It improves over the previous versions of *GPT–4*. Unfortunately, all *GPT–4* variants are proprietary, and their source code is not published. Therefore, we do not know anything about the architecture of this model.

2.2.2 Evaluating VLMs

The current state–of–the–art VLMs were described in the previous parts. The models are trained and evaluated on numerous data sets across various tasks, including image captioning, VQA, visual reasoning, visual conversational QA, image question generation, video question answering, and image classification. However, the issue with the current VLMs is that each model is trained and assessed on slightly different tasks and data sets.

---

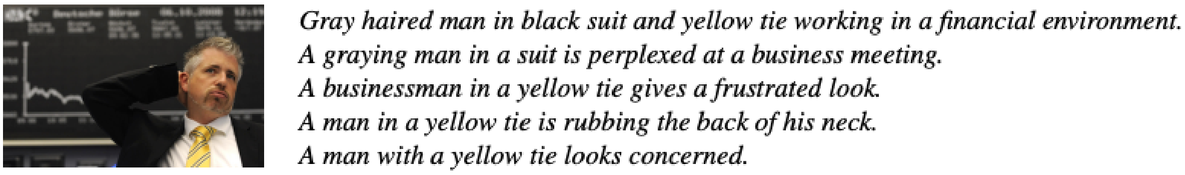
[2] `https://openai.com/index/hello-gpt-4o/`

Image captioning and VQA are popular tasks on which many VLMs are evaluated. To compare the current state–of–the–art VLMs we will choose instance data sets from these tasks and compare the models' results on them. *Flickr30K* [41] and *NoCaps* [42] are two image captioning data sets commonly used. *VQAv2* [43] and *Vizwiz* [44] are popular VQA data sets. We describe each task and metric utilized to measure the models' performance, then depict the presented models' results on these four tasks.

As an alternative option for benchmarking VLMs, we also included a short section of available VLM leaderboards that we found.

Image Captioning Data Sets

Flickr30K *Flickr30K* is an image captioning data set that contains 31 thousand images collected from Flickr[3]. Each image has five ground truth captions. Figure 2.1 displays an example from this data set.



Gray haired man in black suit and yellow tie working in a financial environment.
A graying man in a suit is perplexed at a business meeting.
A businessman in a yellow tie gives a frustrated look.
A man in a yellow tie is rubbing the back of his neck.
A man with a yellow tie looks concerned.

Figure 2.1: An example from the paper [41] of the *Flickr30K* data set

NoCaps *NoCaps* is an image captioning data set. It contains over 15 thousand images with 166 thousand human–written captions for novel object image captioning. Figure 2.2 displays an example from this data set.

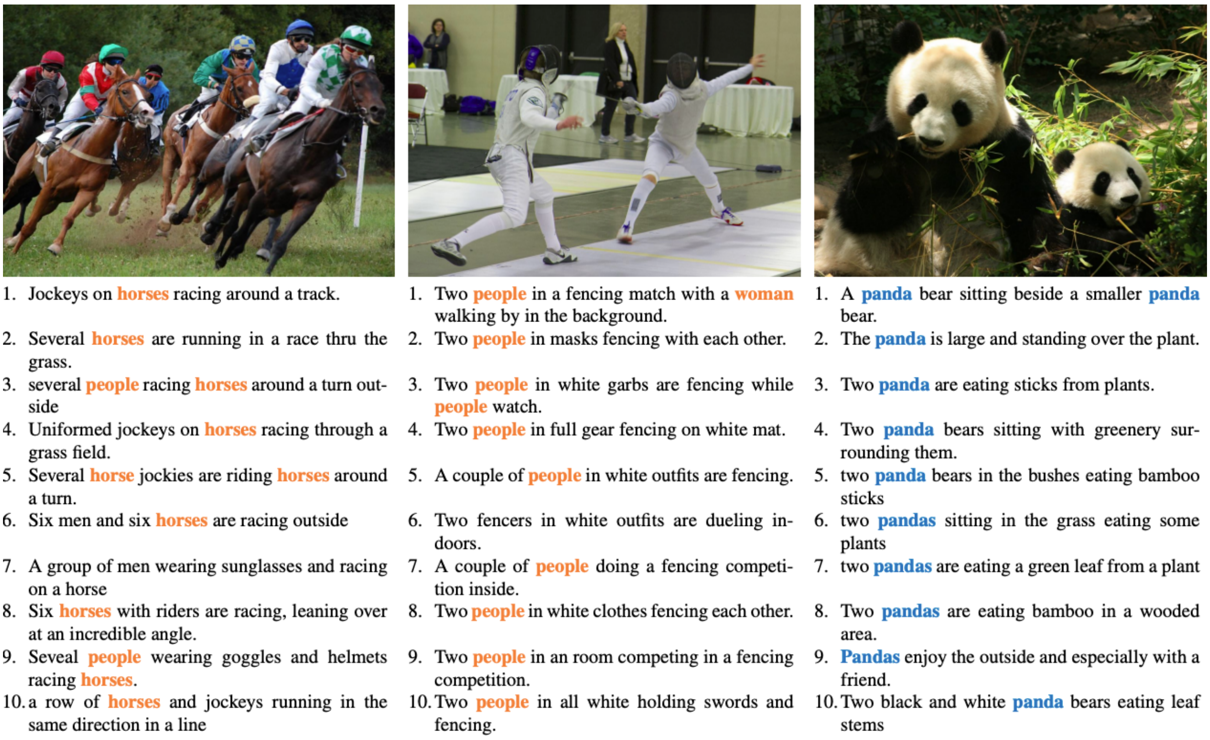
VQA Data Sets

VQAv2 *VQAv2* is a data set for open–ended image QA. The training set contains 82 thousand samples, the validation set 40 thousand, and the test set 81 thousand samples. Figure 2.3 displays an example from this data set.

Vizwiz The *VizWiz* VQA data set originates from a natural VQA setting where blind people took an image and recorded a spoken question, together with ten crowd–sourced answers per visual question. The proposed challenge addresses the following two tasks for this data set: predict the answer to a visual question and predict whether a visual question cannot be answered. Figure 2.4 displays examples from this data set.

---

[3] `https://www.flickr.com`



1. Jockeys on horses racing around a track.
2. Several horses are running in a race thru the grass.
3. several people racing horses around a turn outside
4. Uniformed jockeys on horses racing through a grass field.
5. Several horse jockies are riding horses around a turn.
6. Six men and six horses are racing outside
7. A group of men wearing sunglasses and racing on a horse
8. Six horses with riders are racing, leaning over at an incredible angle.
9. Seveal people wearing goggles and helmets racing horses.
10. a row of horses and jockeys running in the same direction in a line

1. Two people in a fencing match with a woman walking by in the background.
2. Two people in masks fencing with each other.
3. Two people in white garbs are fencing while people watch.
4. Two people in full gear fencing on white mat.
5. A couple of people in white outfits are fencing.
6. Two fencers in white outfits are dueling indoors.
7. A couple of people doing a fencing competition inside.
8. Two people in white clothes fencing each other.
9. Two people in an room competing in a fencing competition.
10. Two people in all white holding swords and fencing.

1. A panda bear sitting beside a smaller panda bear.
2. The panda is large and standing over the plant.
3. Two panda are eating sticks from plants.
4. Two panda bears sitting with greenery surrounding them.
5. two panda bears in the bushes eating bamboo sticks
6. two pandas sitting in the grass eating some plants
7. two pandas are eating a green leaf from a plant
8. Two pandas are eating bamboo in a wooded area.
9. Pandas enjoy the outside and especially with a friend.
10. Two black and white panda bears eating leaf stems

Figure 2.2: Two examples from the paper [42] of the *NoCaps* data set. Each image is annotated with 10 reference captions

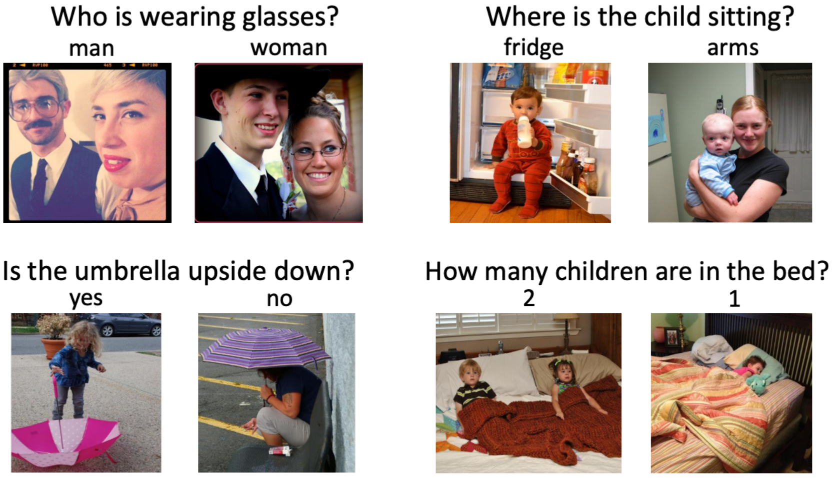
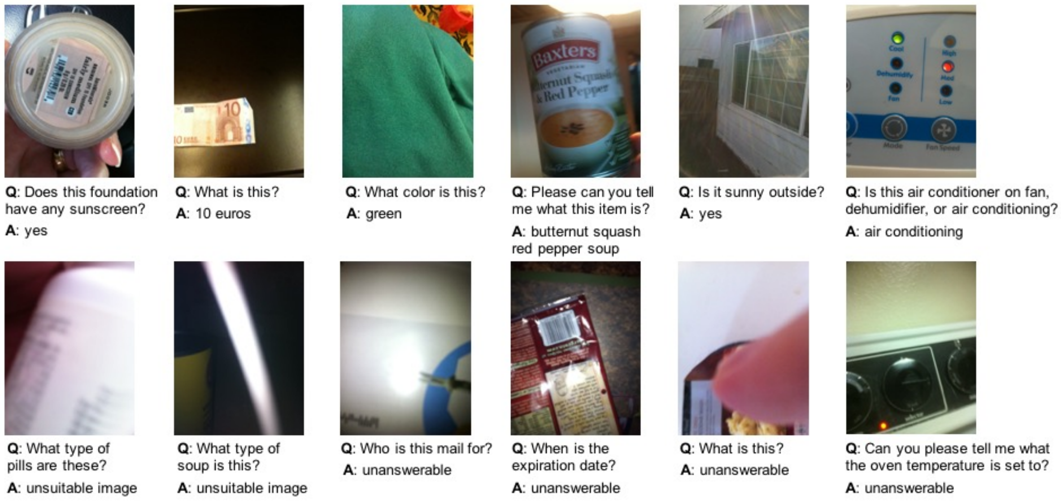

Who is wearing glasses?
man
woman
Where is the child sitting?
fridge
arms
Is the umbrella upside down?
yes
no
How many children are in the bed?
2
1

Figure 2.3: Two examples from the data set website [45] of the *VQAv2* data set

Evaluation Metrics

CIDEr score The *CIDEr score* [47] measures the similarity between a generated caption and the reference captions. It is based on the concept of consensus: the idea that good captions should not only be similar to reference captions in terms of word choice and grammar but also in terms of meaning and content. The similarity is measured using *Term Frequency-Inverse*



Q: Does this foundation have any sunscreen?
A: yes

Q: What is this?
A: 10 euros

Q: What color is this?
A: green

Q: Please can you tell me what this item is?
A: butternut squash red pepper soup

Q: Is it sunny outside?
A: yes

Q: Is this air conditioner on fan, dehumidifier, or air conditioning?
A: air conditioning

Q: What type of pills are these?
A: unsuitable image

Q: What type of soup is this?
A: unsuitable image

Q: Who is this mail for?
A: unanswerable

Q: When is the expiration date?
A: unanswerable

Q: What is this?
A: unanswerable

Q: Can you please tell me what the oven temperature is set to?
A: unanswerable

Figure 2.4: Examples from the data set website [46] of the *VizWiz* VQA data set

Document Frequency (tf–idf) [48]. *tf–idf* is a statistical method for assessing the significance of a word within a document in a corpus. The *CIDEr score* is used to measure the quality of image captioning models.

top-1 accuracy *Top–1 accuracy* measures the proportion of examples for which the generated text matches the target label. It is commonly used to evaluate models trained on the VQA tasks.

Results

The table 2.1 shows the results of the presented models (except *DePlot* which was not trained or evaluated on the tasks of image captioning and VQA) on the *Flickr30K*, *NoCaps*, *VQAv2*, and *Vizwiz* data sets reported by their authors. *InstructBLIP FlanT5XL* performs very well on the image captioning tasks. *GIT2* model performs well on *NoCaps* image captioning task and after fine–tuning also very well on VQA tasks. *LlaVA* and *Idefics2* models were only evaluated on the two VQA tasks on which they performed well.

VLMs Leaderboard

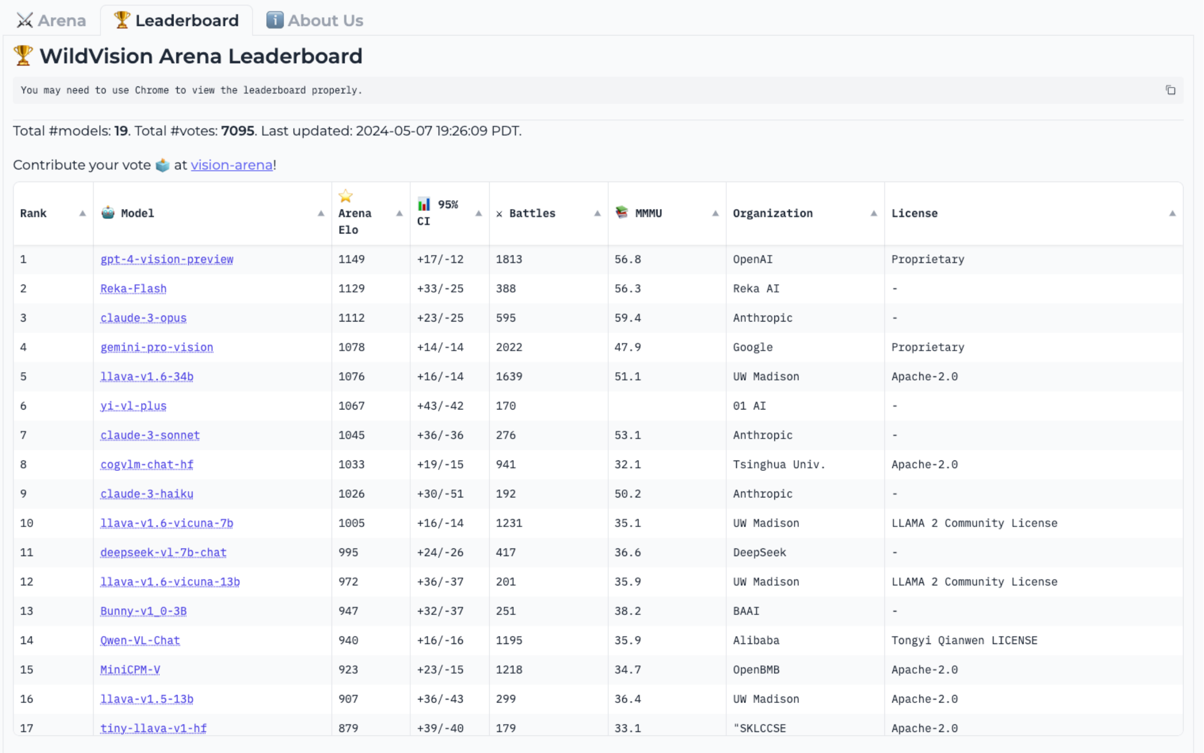
Another option to compare VLMs is via leaderboards. To the best of our knowledge, two leaderboards exist to compare VLMs.

Vision Arena [49] is a leaderboard based on user–anonymous voting of model outputs. In this arena, the users choose an image and a prompt, or they can be sampled. After that, outputs from two different random models are sampled anonymously, and the user can judge the

Model Name	**Image Captioning [CIDEr]**		**VQA [top-1 accuracy %]**	
	Flickr30K [↑]	NoCaps [↑]	VQAv2 [↑]	Vizwiz [↑]
BLIP-2 FlanT5XL	76.10	104.50	63.00	29.80
BLIP-2 FlanT5XXL	73.70	98.40	65.00	29.40
BLIP-2 Vicuna 7B	74.90	107.50	–	25.30
BLIP-2 Vicuna 13B	71.60	103.90	–	19.60
InstructBLIP FlanT5XL	**84.50**	119.90	Training	32.70
InstructBLIP Vicuna 7B	82.40	123.10	Training	34.50
MiniGPT–v2 7B	Training	-	Training	32.90
MiniGPT–v2 7B Chat	Training	-	Training	53.60
LLaVA 1.5 Vicuna 7B	–	–	78.50*	50.00
LLaVA 1.5 Vicuna 13B	–	–	80.00*	53.60
LLaVA-NeXT Mistral 7B	–	–	82.20*	60.00
LLaVA-NeXT 34B	–	–	**83.70***	63.80
Idefics2 8B	–	–	81.20	–
GIT2	50.70	**124.80**	81.74*	**70.97***
GPT–4 (0–shot)	–	–	77.20	–

Table 2.1: A table with results of state–of–the–art VLMs. (*: model was trained on this task)

models' responses. The leaderboard is available here. Figure 2.5 shows a screenshot of the leaderboard.



WildVision Arena Leaderboard

You may need to use Chrome to view the leaderboard properly.

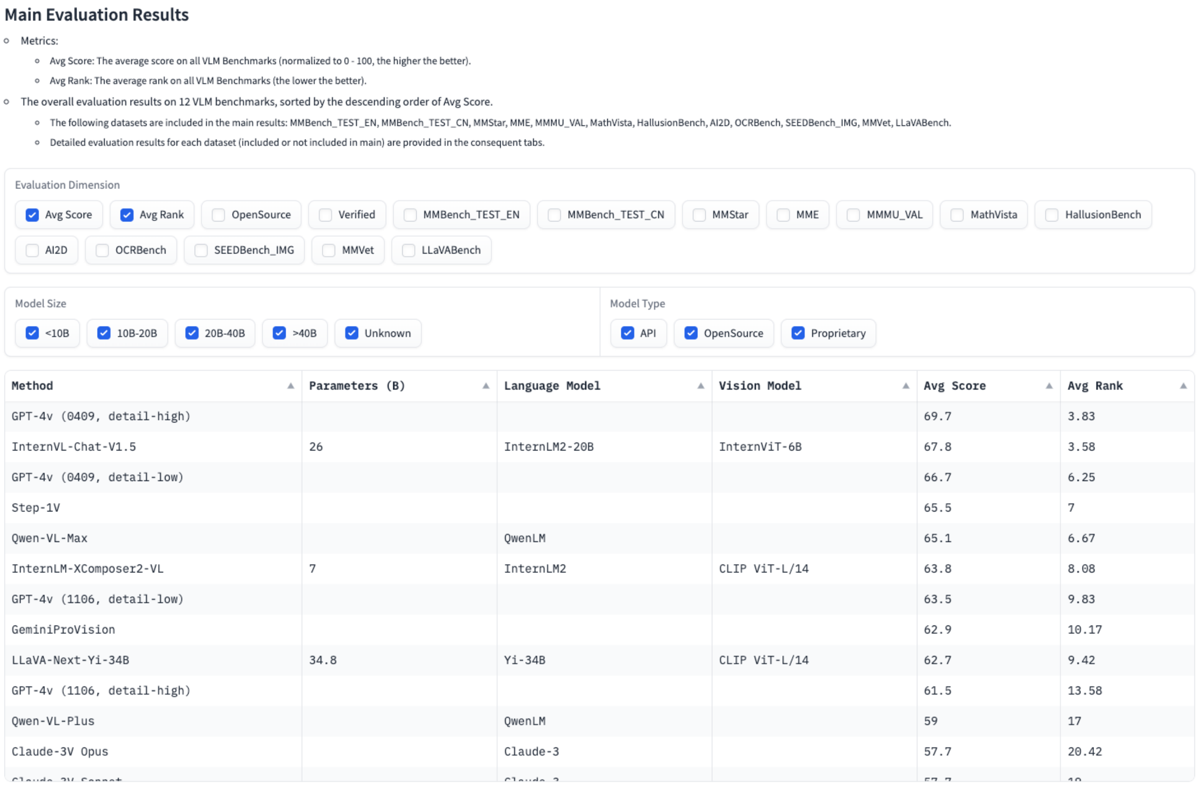
Total #models: **19**. Total #votes: **7095**. Last updated: 2024-05-07 19:26:09 PDT.

Contribute your vote at vision-arena!

Rank	Model	Arena Elo	95% CI	× Battles	MMMU	Organization	License
1	gpt-4-vision-preview	1149	+17/-12	1813	56.8	OpenAI	Proprietary
2	Reka-Flash	1129	+33/-25	388	56.3	Reka AI	-
3	claude-3-opus	1112	+23/-25	595	59.4	Anthropic	-
4	gemini-pro-vision	1078	+14/-14	2022	47.9	Google	Proprietary
5	llava-v1.6-34b	1076	+16/-14	1639	51.1	UW Madison	Apache-2.0
6	yi-vl-plus	1067	+43/-42	170		01 AI	-
7	claude-3-sonnet	1045	+36/-36	276	53.1	Anthropic	-
8	cogvlm-chat-hf	1033	+19/-15	941	32.1	Tsinghua Univ.	Apache-2.0
9	claude-3-haiku	1026	+30/-51	192	50.2	Anthropic	-
10	llava-v1.6-vicuna-7b	1005	+16/-14	1231	35.1	UW Madison	LLAMA 2 Community License
11	deepseek-vl-7b-chat	995	+24/-26	417	36.6	DeepSeek	-
12	llava-v1.6-vicuna-13b	972	+36/-37	201	35.9	UW Madison	LLAMA 2 Community License
13	Bunny-v1_0-3B	947	+32/-37	251	38.2	BAAI	-
14	Qwen-VL-Chat	940	+16/-16	1195	35.9	Alibaba	Tongyi Qianwen LICENSE
15	MiniCPM-V	923	+23/-15	1218	34.7	OpenBMB	Apache-2.0
16	llava-v1.5-13b	907	+36/-43	299	36.4	UW Madison	Apache-2.0
17	tiny-llava-v1-hf	879	+39/-40	179	33.1	"SKLCCSE	Apache-2.0

Figure 2.5: The Vision Arena leaderboard

OpenVLM Leaderboard [50] is a leaderboard where various VLMs are ranked based on their performance on various benchmarks. As of the date of writing this text (May 2024), it covers 56 different VLMs and 22 different multi–modal benchmarks. The leaderboard is available here. Figure 2.6 shows a screenshot of the leaderboard.



Main Evaluation Results

- Metrics:
  - Avg Score: The average score on all VLM Benchmarks (normalized to 0 - 100, the higher the better).
  - Avg Rank: The average rank on all VLM Benchmarks (the lower the better).
- The overall evaluation results on 12 VLM benchmarks, sorted by the descending order of Avg Score.
  - The following datasets are included in the main results: MMBench_TEST_EN, MMBench_TEST_CN, MMStar, MME, MMMU_VAL, MathVista, HallusionBench, AI2D, OCRBench, SEEDBench_IMG, MMVet, LLaVABench.
  - Detailed evaluation results for each dataset (included or not included in main) are provided in the consequent tabs.

Evaluation Dimension

☑ Avg Score ☑ Avg Rank ☐ OpenSource ☐ Verified ☐ MMBench_TEST_EN ☐ MMBench_TEST_CN ☐ MMStar ☐ MME ☐ MMMU_VAL ☐ MathVista ☐ HallusionBench ☐ AI2D ☐ OCRBench ☐ SEEDBench_IMG ☐ MMVet ☐ LLaVABench

Model Size

☑ <10B ☑ 10B-20B ☑ 20B-40B ☑ >40B ☑ Unknown

Model Type

☑ API ☑ OpenSource ☑ Proprietary

Method	Parameters (B)	Language Model	Vision Model	Avg Score	Avg Rank
GPT-4v (0409, detail-high)				69.7	3.83
InternVL-Chat-V1.5	26	InternLM2-20B	InternViT-6B	67.8	3.58
GPT-4v (0409, detail-low)				66.7	6.25
Step-1V				65.5	7
Qwen-VL-Max		QwenLM		65.1	6.67
InternLM-XComposer2-VL	7	InternLM2	CLIP ViT-L/14	63.8	8.08
GPT-4v (1106, detail-low)				63.5	9.83
GeminiProVision				62.9	10.17
LLaVA-Next-Yi-34B	34.8	Yi-34B	CLIP ViT-L/14	62.7	9.42
GPT-4v (1106, detail-high)				61.5	13.58
Qwen-VL-Plus		QwenLM		59	17
Claude-3V Opus		Claude-3		57.7	20.42

Figure 2.6: The OpenVLM leaderboard

2.3 Retrieval–Augmented Generation (RAG)

As time passed and the LM grew, this increased size enabled them to store more world knowledge they could use during generation. However, this did not allow LLMs to remember every single information available. Moreover, the world is constantly changing, so pieces of information can get outdated, or new pieces of information can arise daily. Therefore, LLMs can generate very confident responses with proper grammatical features, which are factually wrong, a phenomenon called hallucination [2]. Retrieval–Augmented Generation (RAG) [3] is addressing this problem by retrieving knowledge from external sources (e.g., Wikipedia[4]) and passing the acquired passages to the prompt. The prompt is an input piece of text used to condition the generated response; in other words, to steer the LM in the right direction.

---

[4] `https://en.wikipedia.org/`

RAG consists of two components: a retriever and a generator. A retriever is a system that retrieves relevant documents given an input query from a collection of documents. These documents are then fed to the generator (a LLM) that takes the input query with the retrieved documents and, conditioned on them, generates a response. RAG is a well–suited for the task of open domain QA. QA is a task of answering questions. Open domain QA, also known as retrieval–based QA, is a task of finding or generating an answer to a user question based on the information available on the web or in some collection of documents [2].

2.3.1 RAG components

Retriever

Retriever (also called search engine) is a resulting system of Information retrieval (IR). IR is the task of retrieving relevant documents from a set of documents based on a provided user query. A document refers to any segment of text that the system retrieves (news articles, websites, messages). IR is solved by mapping the query and documents to their vector representation (embeddings). It uses the cosine similarity between the query and documents' embeddings to rank the documents based on their similarity with the user query [2]. To save time, the document embeddings are pre–calculated and stored so that only the query embedding is computed during the search. Historically, *tf–idf* [48] or *BM25* [51] were a popular choice for weighting words. Modern IR systems use transformer–based encoders such as *BERT* [52] for computing embeddings. However, since the context window (the number of tokens a language model can process during generation) is limited, usually between 512 to $32K$ tokens, the documents must be split into several chunks.

Evaluation of Retrievers Popular metrics to evaluate retrievers are *precision*, *recall*, *mean average precision (MAP)*, and *mean reciprocal rank (MRR)* [53]. *MAP* measures the average precision across multiple text passages, considering *precision* only at those points where a relevant document has been encountered. *MRR* evaluates the average reciprocal rank of the first relevant result across multiple queries [2].

Generator

Any generative LM can be used as a generator in RAG. The original paper utilized a transformer–based model called *BART* [54]. Following today's trend, a popular choice is using one of the latest instruction–tuned LLM such as *GPT–4* [40], *Mistral* [34], *Llama 3* [55] or *Phi–3* [56]. An instruction–tuned LLM is a fine–tuned version of the specific LLM on a labeled dataset of instructional prompts and corresponding outputs [57].

2.3.2 Multi–modal RAG

Standard RAG systems work only with textual knowledge, neglecting the knowledge hidden in other modalities, such as images. Multi–modal RAG, on the other hand, also incorporates knowledge from other modalities such as images.

We found two possible approaches for multi–modal retrieval in the literature. The first option is to embed all modalities into the same vector space using multi–modal embedding models like *CLIP* or *SigLIP*. The second option is to ground all modalities into one primary modality, for example, using a VLM such as *LlaVA* or *GPT–4* to generate images' captions and embed them using text embedding models.

The first approach was used in work *Murag: Multimodal retrieval-augmented generator for open question answering over images and text, Chen, Wenhu, et al.* [58], in which the authors proposed *MuRAG*, a multi–modal transformer–based model used for open domain QA. Both the retriever and the generator use a multi–modal backbone, which is capable of encoding image–text pairs in such a way that they are suitable for both retrieval and answer generation. During generation, the retrieved image–text pairs are combined with the query, fed to the multi–modal backbone encoder, and passed to a decoder to generate a textual response. Another example of the first approach in [59], where authors used *CLIP* model (*ViT-L/14*) [16] for generating embeddings.

The second approach was used in [60], where authors use *DePlot* model to translate the plots and *neva-22b* [61] from NVIDIA to generate captions from images and decide whether an image is a plot or chart.

Chapter 3

Our Approach

This section will discuss our approach to incorporating images into the RAG system, which is focused on technical manuals. We will propose a multi–modal RAG pipeline and describe each component. Then, we will describe the data and metrics used in our work.

3.1 Proposed RAG Pipeline

RAG and its components were described in the previous chapter. As our focus in this work is on image integration into a RAG pipeline, we will not conduct any experiments regarding selecting the best text chunkenization method, text embedding model, and LLM responsible for generating answers. Instead, we will propose a general multi–modal RAG pipeline that can be used with arbitrary vector storage, text embedding, and answering models. We will divide the RAG pipeline into two phases; pre–processing and an inference phase. During the pre–processing phase, the input manual(s) is(are) processed, and the resulting chunk embeddings and extracted images are stored to be easily used during the inference phase, which happens whenever a user asks a question. The pre–processing phase is executed only when the QA application is created, or a new or updated manual is added.

3.1.1 Pre–Processing Phase

The pre–processing phase of the proposed multi–modal RAG pipeline is depicted in the figure 3.1. Text and all images are extracted from each PDF manual. An image is classified as an icon or a standard image based on its resolution. In our work, an image is considered an icon when both the width and height of the image are below $100px$. Standard images are stored for inference into folders based on what pages they are in the manual. Icons are localized in the text and fed into a VLM that generates the icon's caption. If the icon's location is detected, then the generated icon caption is inserted at the detected location.

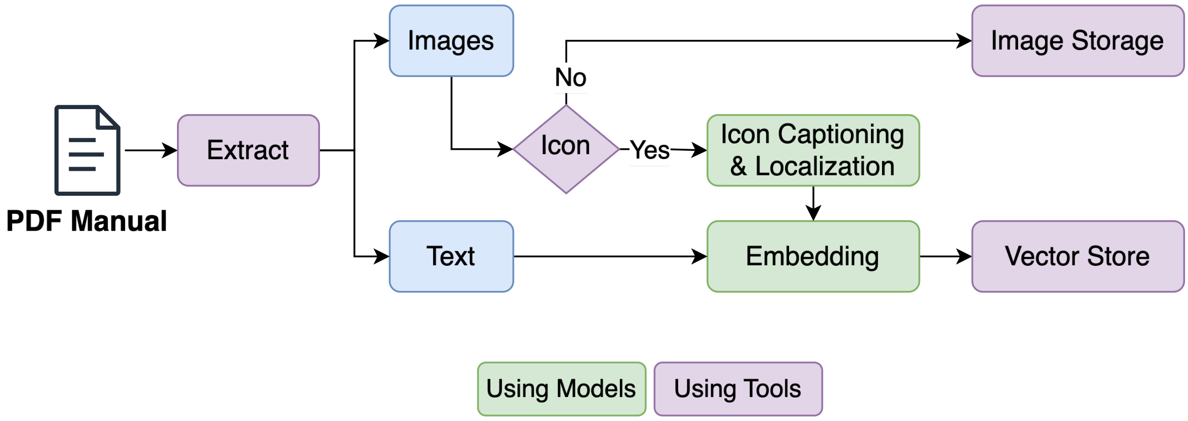


Figure 3.1: Pre–processing phase of the proposed multi–modal RAG pipeline

The icon extraction pipeline's pseudo code is written in 1. To extract icons from the PDF file and to extract text around specified coordinates on a page in the PDF file, we utilize a Python package *PyMuPDF*[1]. Because the PDF parsing can create mistakes (wrongly parsed letters or even words), we use a Levenshtein distance to compare the text segments parsed around the icon's coordinates on a page with the extracted text of the whole page and sort the matches based on this distance. The icon localization does not work perfectly; it can make mistakes or even not localize an icon. There are several reasons for that. First, there does not have to be any text around the icon, so the icon cannot be detected if no text is parsed around the icon. The text in the manuals can be structured into more than one column, so the text around the specific icon can very well be from a different text section, and the icon can be localized incorrectly or not at all. Lastly, the detected text around an icon can be present in the text several times, and therefore, the icon can be mistakenly localized. The first two mentioned issues can be reduced by carefully picking the Levenshtein distance threshold and the bounding box size used for extracting text around an icon's coordinates. Choosing a big threshold will result in a higher *recall* but smaller *precision*. The selection of the optimal values will be a trade–off between *precision* and *recall*. However, wrong localization is more problematic in our case, so we prioritized *precision* more. The third problem can be reduced by sorting the icons based on their coordinates from top to bottom and left to right. One of our experiments was designed to measure the *precision* of the algorithm's localization.

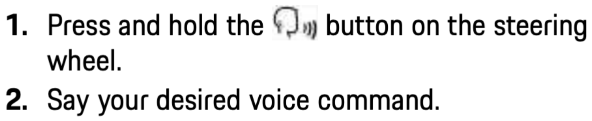
We decided to process icons differently from other images because the icons often represent some buttons or symbols that help to make the text more transparent. Furthermore, icons have tiny resolutions and are, therefore, usually placed right between words, meaning that during text parsing, the extracted text segments around are disconnected because they are missing the information the icons represent. An example of such an icon can be seen in the figure 3.2.

---

[1] `https://pymupdf.readthedocs.io/en/latest/`

Algorithm 1: Icon Extraction Pipeline

```
Data: Manual file in .pdf
Result: Extracted text from the manual with injected textual description of the icons.
Function findTemplate(template, text, threshold):
    matches ← []
    for i ← 0 to (text.length - template.length) do
        candidate ← text[i:i+templateLength]
        dist ← Levenshtein(candidate, template)
        if dist $\leq$ threshold then
            save the candidate to matches with dist
    return sorted matches based on dist;
Function extractIcons(manual):
    extractedText ← ""
    foreach page in manual do
        textPage ← extract text from the page
        icons ← extract icons on page
        foreach icon in icons do
            textAround ← extract text around the icon
            candidates ← findTemplate(textAround, textPage, threshold)
            if found match in candidates then
                inject description of the icon into the textPage
        extractedText ← extractedText + textPage
    return extractedText
```



1. Press and hold the button on the steering wheel.
2. Say your desired voice command.

Figure 3.2: The potential icon problem

3.1.2 Inference Phase

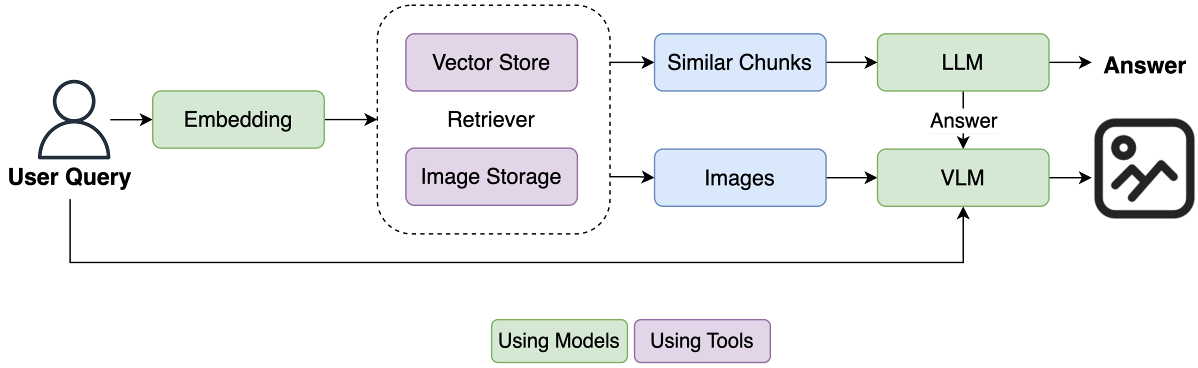


Figure 3.3: Inference phase of the proposed multi–modal RAG pipeline

The inference phase of the proposed multi–modal RAG pipeline is depicted in the figure

3.3. An input user query is embedded using a text embedding model, and the resulting vector is passed to the retriever, which retrieves the k–most similar chunks to the query with all the images from the pages where the most similar chunks originate. Retrieving images from particular pages is solved by storing the source page(s) from which a specific chunk originates. The retrieved similar text chunks are passed into a LLM that, conditioned on the chunks, generates an answer. The generated response and/or the user query are alongside the retrieved images passed into a VLM to determine which image(s) is(are) relevant, if any. The image retrieval in the pipeline utilizes the idea that an image on a page is relevant to some part of the page. By assuming this, we can easily filter out most images in the storage and focus only on a few selected. The advantage of this approach is that the images are retrieved without the necessity to compute anything. The disadvantage is that the information contained in the retrieved images is not considered when generating the answer.

3.1.3 Extensions

We can further improve the proposed RAG pipeline by translating plots into linearized text using a specialized model or supporting languages other than English.

Plots

Inspired by [60], we can improve our proposed pipeline by translating plots into text using the *DePlot* model. Doing this should increase the performance of the QA system in situations when the user question is referring to a chart or plot because, as the work [37] suggests, visual language reasoning is a challenging task, and splitting it into two parts helps.

To utilize the *DePlot* model, it is crucial to differentiate the plots and charts from regular images. For this differentiation, we can employ one of the pre-trained instruction-based VLM. This update requires modifying only the proposed pipeline's pre–processing phase. The updated pre–processing phase with plots is depicted in figure 3.4. The translated textual representations of icons are embedded using the selected text embedding model.

Other Languages Support

In our work, we assume the data is in English. However, using the proposed multi–modal RAG pipeline with other languages is possible. This requires replacing each model in the proposed RAG pipeline with a multilingual one. Here are suggestions for the components:

- **Text embedding model**: *Multilingual MiniLM–L12 v2*[2] [62]
- **Icon captioning model**: *GPT–4*

---

[2] `https://tinyurl.com/yc5tt5mk`

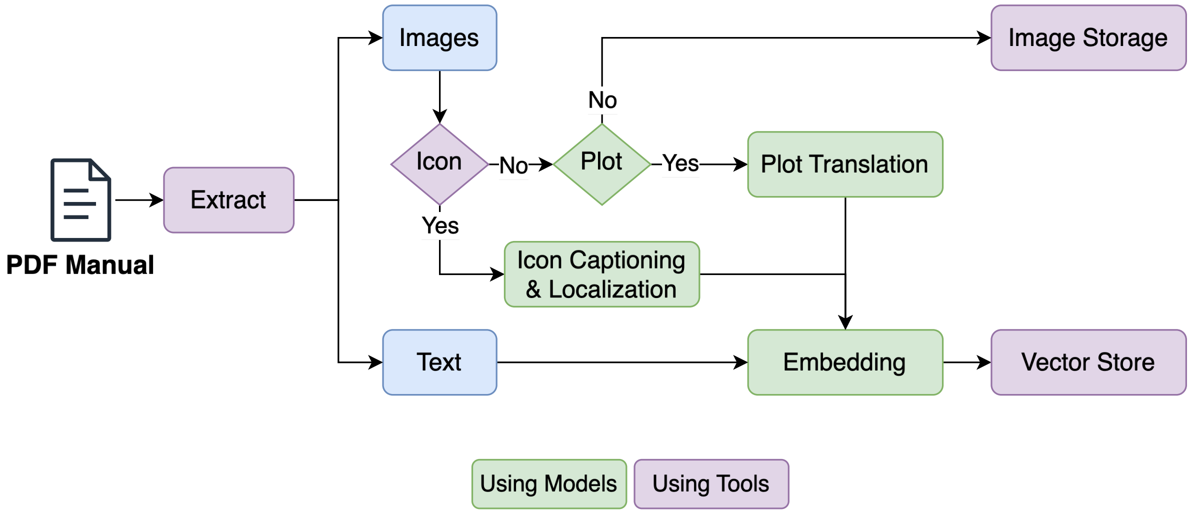


Figure 3.4: Updated pre–processing phase of the proposed Multi–modal RAG pipeline with plots

- **LLM for generating answers**: *C4AI Command-R*[3]
- **VLM**: *Multilingual SigLIP base (Patch16/256)*[4] [17]

3.2 Data

For our work, we selected five manuals about electric vehicles. All these manuals are publicly available on the internet. All manuals are in English. They all describe how to operate and use the vehicle and all of their features. Here is the list of the selected manuals:

1. Porsche Taycan[5]
2. BMW I4[6]
3. Tesla Model S[7]
4. Volvo[8]
5. Ford Mustang Mach E[9]

Table 3.1 shows some essential characteristics of the selected manuals (such as the number of pages, words, and images). The table shows that manuals about vehicles are rich in images,

---

[3] `https://huggingface.co/CohereForAI/c4ai-command-r-v01-4bit`
[4] `https://huggingface.co/google/siglip-base-patch16-256-multilingual`
[5] `https://tinyurl.com/436u54s6`
[6] `https://tinyurl.com/puvfzbjf`
[7] `https://tinyurl.com/4v6u9k6j`
[8] `https://tinyurl.com/3z4n2ppu`
[9] `https://tinyurl.com/5n89ht3y`

as could be suspected, and ignoring them could result in losing much knowledge. Please note that the icons in these manuals are usually displayed multiple times even though it is visible that the manuals contain many icons, which only proves their importance.

Manual	**# of Pages**	**# of Images**	**# of Icons**	**# of Words**	**# of Unique Words**
Porsche Taycan	327	823	528	108 128	5 999
BMW I4	404	1 080	735	115 707	5 104
Tesla Model S	292	467	299	124 431	5 121
Volvo	921	2 345	1 373	198 564	7 592
Ford Mach E	742	506	234	115 325	5 912
Avg	476.6	1 319.6	909.4	132 431	5 945.6
Total	2 383	6 598	4 547	662 155	13 046

Table 3.1: Analysis of the selected manuals

Generally, the most frequent words are words like *the* and *a*. These words are called stop words. Figure 3.5 shows the 25 most frequent words occurring in the manuals after the stop words were removed. By far, the most frequent word is *vehicle* with over nine thousand occurrences, which is around 2.5 times more than the second most frequent word *system*.

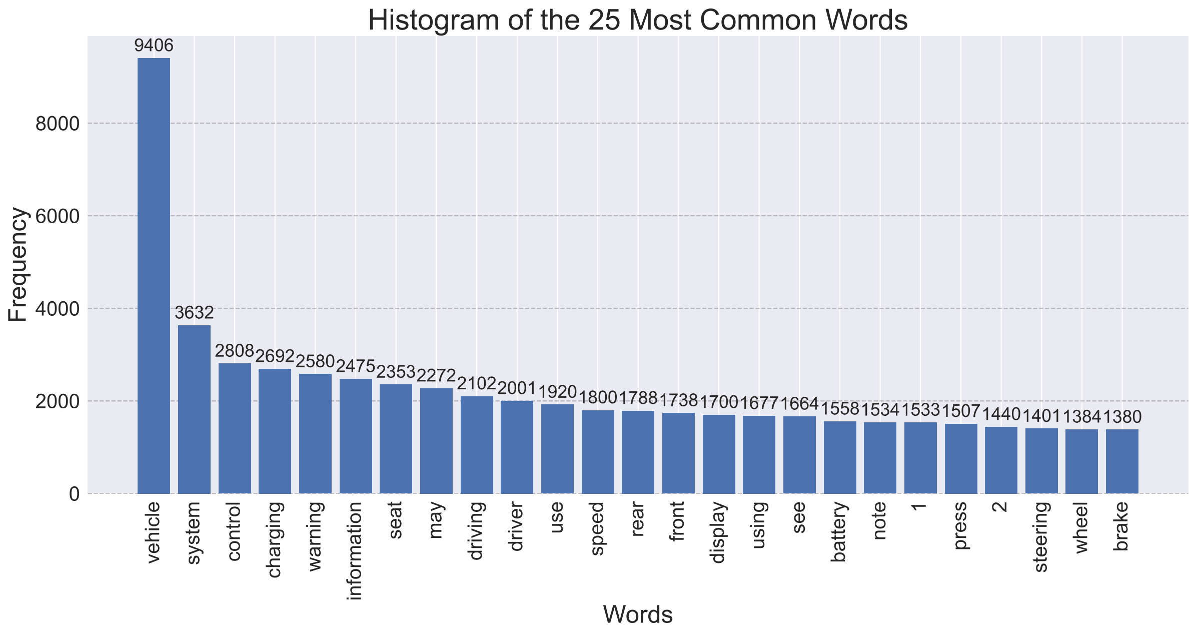


Figure 3.5: Histogram of the 25 most common words in the selected manuals

Table 3.2 displays image resolution statistics of the manuals' images. The table tells us that the Porsche Taycan manual contains the smallest images. The smallest image from the manuals has only a resolution of $9x12px$.

Manual	**Image Resolution [px]**			
	Mean ± Std	**Min**	**Median**	**Max**
Porsche Taycan	146.65±171.17 x 116.6±138.7	9x12	29x19	1181x779
BMW I4	192.73±382.6 x 107.94±107.12	8x17	56x60	3818x1247
Tesla Model S	175.71±206.69 x 129.56±162.36	24x18	70x70	1654x2100
Volvo	117.86±148.11 x 76.99±72.89	13x13	38x38	2637x359
Ford Mach E	200.44±194.17 x 151.17±145.46	18x19	138x98	1396x1123
Avg	166.68±220.55 x 116.45±125.3	14.4x15.8	66.2x57.0	2137.2x1121.6

Table 3.2: Analysis of the resolutions of the images from the selected manuals

3.3 Created Data Sets

Since this work focuses on integrating images and icons into a RAG pipeline, we needed data sets to evaluate the components of the proposed multi–modal RAG pipeline that are responsible for the image integration. These components are an icon captioning VLM, a plot detector, and, most importantly, a VLM used for finding relevant images based on a text. We created data sets for all these components, which we will describe in the following sections.

3.3.1 Image Text Relevance Data Set

The purpose of the image–text–relevance task is to determine whether the particular image is relevant to the provided text (answer, question, or both). This data set represents an image retrieval task or can also be seen as a specific type of VQA task. Each sample in the dataset is a quadruplet (I, Q, A, L) of an image, a question, an answer, and a corresponding label stating whether the image is relevant to the question and answer. The labels can be either relevant ($label = 1$) or irrelevant ($label = 0$). Figure 3.6 shows an example from this data set.

We created two versions of this data set depending on the creation method. Each version is described in the following subsections. Please note that when working with this task, we first concatenated the splits from both versions and that the number of retrieved images differs for each question.

Semi-automatically Curated Data Set

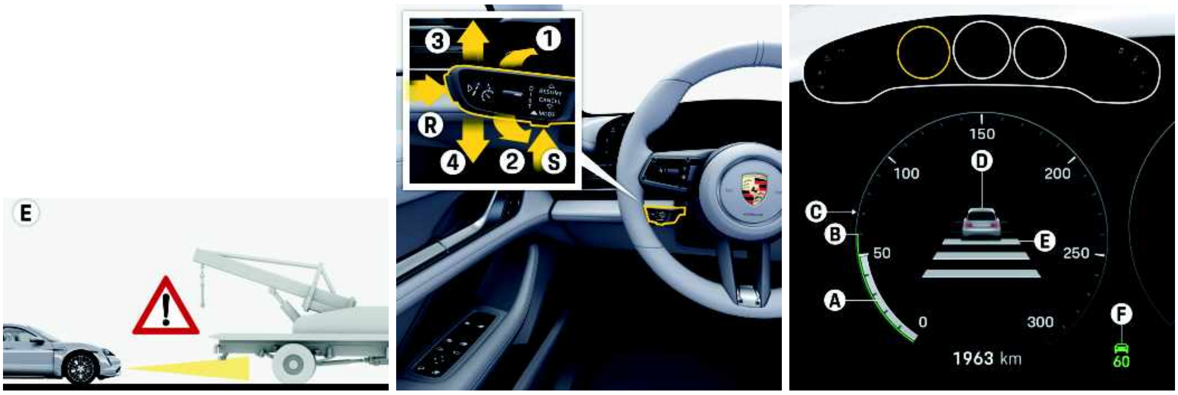
To create this data set, we used a self–service[10][11] conversational QA tool. This tool lets users upload a PDF manual; it proceeds to parse the manual, extract images, generate embeddings, and then answer any question a user asks in a chat. It is powered by *ChatGPT*[12]. The answer

---

[10] `https://slf-service.web.app/index.html`

[11] During the work on this thesis, we participated in the development of this service, where we were in charge of the image integration.

[12] `https://openai.com/index/chatgpt/`



(a) **Label 0** (not relevant image). (b) **Label 1** (relevant image). (c) **Label 1** (relevant image).

Figure 3.6: Example of the created image–text–relevance data set with the following question: *"How do I turn on the cruise control?"* and the following answer: *"To turn on cruise control, press the R button on the control stalk. Then, select CC using the rotary knob on the steering wheel and press to confirm. Cruise control is switched on and passive.The operating status appears grey in the status display. There is no speed stored."*.

contains not only the textual answer but also images from the retrieved most relevant pages. We uploaded the selected five manuals to this utility and asked 42 questions about the cars (to see the full list of the questions, refer to Appendix B.1). Some of these 42 questions were created manually; the rest was generated using *ChatGPT*. We saved all the questions, the generated answers, and the retrieved images. We then labeled whether each retrieved image was relevant to the question and answer. Please note that because the questions were general, sometimes, no relevant images were returned due to missing images on the topic. The dataset is available here at HuggingFace. The table 3.3 shows data analysis of this version of the data set. Please note that the average values are rounded to integers except for the column with the average number of words. The table shows that simply returning images from the retrieved most relevant pages is insufficient, and only a few are relevant. This is most likely the questions we ask about a specific detail that only a small portion of data deals with.

Manual	**# of Samples**		**# of Words**		**Image Resolution [px]**		
	Positive	**Negative**	**Avg**	**Max**	**Min**	**Median**	**Max**
Porsche Taycan	21	79	29.1±23.5	119	322x153	345x345	637x345
BMW I4	9	118	39.5±32.2	105	120x126	448x241	3818x519
Tesla Model S	8	53	52.3±44.7	146	150x126	380x281	760x527
Volvo	23	81	44.3±45.9	154	219x113	219x151	369x265
Ford Mach E	12	54	30.5±34.1	138	114x106	330x218	1396x777
Avg	15	77	39.1±36.1	132	185x125	344x247	1396x487

Table 3.3: Analysis of the semi-automatically curated image–text–relevance data set

Manually Curated Data Set

As the name suggests, this data set was created entirely manually. The goal was to create a high–value data set. The creation process was straightforward. We skimmed through each of the five selected manuals, and whenever we saw an image about which we could ask something, we noted the question and the answer and saved the image(s). Because the creation process was manual, it was more time–consuming and therefore, this version is smaller than the semi-automatically curated one. The dataset is available here at HuggingFace. The table 3.4 shows data analysis of this version of the data set. Please note that the average values are rounded to integers except for the column with the average number of words.

Manual	**# of Samples**		**# of Words**		**Image Resolution [px]**		
	Positive	**Negative**	**Avg**	**Max**	**Min**	**Median**	**Max**
Porsche Taycan	10	20	34.3±23.4	87	166x83	166x99	166x166
BMW I4	26	67	24.2±23.6	74	448x241	448x241	448x241
Tesla Model S	17	36	26.3±21.3	87	379x237	380x263	910x573
Volvo	31	46	26.4±23.8	85	219x151	219x151	703x189
Ford Mach E	19	36	29.9±17.3	61	146x111	343x231	986x583
Avg	21	41	28.2±21.9	79	272x165	311x187	643x350

Table 3.4: Analysis of the manually curated image–text–relevance data set

Data Set Splits

As it is custom, we decided to split the samples into training, validation, and test parts in both data set versions. Samples from the Porsche Taycan, the BMW I4, and the Volvo manuals were chosen as a training split, samples from the Tesla Model S manual as a validation split, and samples from the Mustang Mach E manual as a test split. We decided to split the samples this way, as this will ensure the objectivity of the results. Furthermore, it ensures that no images from the training set will be used in either validation or test splits and vice versa. Table 3.5 shows the split sizes. As shown in figure 3.7, the task is imbalanced, and the relevant images form only a minority of the combined data set.

Data Set	**Train Size**	**Validation Size**	**Test Size**
Semi-automatical	331 (72.3%)	61 (13.3%)	66 (14.4%)
Manual	200 (64.9%)	53 (17.2%)	55 (17.9%)
Total	**531 (69.3%)**	**114 (14.9%)**	**121 (15.8%)**

Table 3.5: Sizes of each data set split

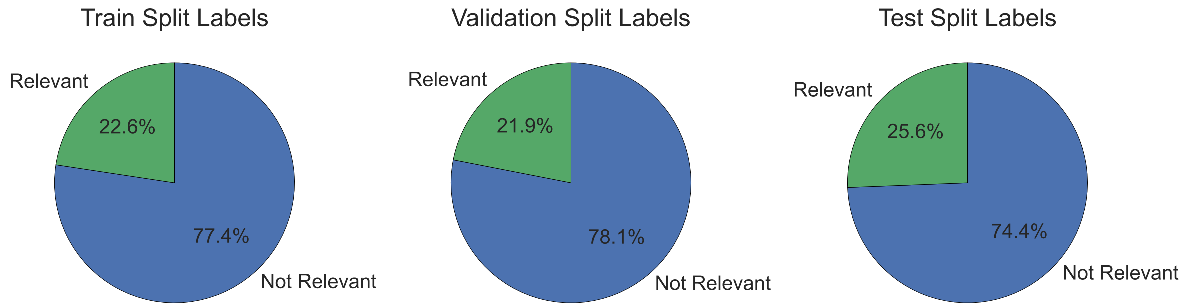


Figure 3.7: Label frequencies of the combined image–text-relevance data set

Used Metric

Since our task is a particular case of image retrieval or VQA, we used standard metrics such as accuracy, *F1–score*, *precision*, and *recall*. Because the data set is imbalanced, we chose the *F1–score* as the primary indicator of the quality of a model.

Modes

Our image–text–relevance data sets provide user questions and the corresponding answers. This creates several combinations of how to exploit these textual pieces of information. We created the following modes: *answer only* (denoted as *A*), *question only* (denoted as *Q*), *question answer* (denoted as *QA*), *question answer max* (denoted as *maxQA*), *question answer mean* (denoted as *meanQA*).

- **answer only** – Only the answer to a user question is used to determine the relevance of the input image.
- **question only** – Only the user question is used to determine the relevance of the input image.
- **question answer** – Both the user question and the answer are used in the prompt to determine the relevance of the input image. This option is used for open–generation VLMs such as *Idefics2*.
- **question answer max/mean** – For both the user query and the corresponding answer, the relevance scores are computed separately, and then, depending on the aggregation function (max/mean), the final relevance is computed. This option is used for multi–modal embedding models such as *CLIP* and *SigLip* models.

Evaluated Models

Models we chose to evaluate on the image–text–relevance task are the following: *CLIP*, *SigLIP*, *GIT*, *BLIP*, *InstructBLIP*, *LlaVA–NeXT*, *Idefics2* and as a reference model, we chose to use *GPT–4o*.

CLIP and SigLIP inference Both *CLIP* and *SigLIP* are a multi–modal embeddings models. They project images and tokenized texts into a shared vector space in which an image with content similar to a text's meaning should be close to each other. The similarity is assessed through cosine similarity. We needed to find a threshold to classify whether a given text is relevant to the provided image to utilize these two models in our image–text–relevance task. Finding such a threshold is depicted in 2.

Algorithm 2: Process of finding the best threshold

```
Data: Validation set
Result: Threshold for cosine similarity
Function findThreshold(validationSet, model):
    bestThreshold ← 0 bestF1 ← 0
    for candidateThreshold ← 0.1 to 0.3 + [0.32, 0.35, 0.4, 0.5] do
        f1Score ← compute F1–score with model, candidateThreshold on validationSet
        if f1Score ≥ bestF1 then
            bestThreshold ← candidateThreshold bestF1 ← f1Score
    return threshold with the best F1–score on validation set;
```

Instruction VLMs inference

Besides *CLIP* and *SigLIP*, all the other models selected for experimenting are generative VLMs. For every mode, we created a shared instruction for all the models. The specific instruction was inserted into a model's prompt. Each model uses a different prompt format, which depends on the model's pre–training. To see the prompts used for all the selected models, please refer to Appendix B.2. Here are the utilized instructions for each mode:

- **answer only** and **question only**
  Is this image relevant to the following text or not?
  Text: answer
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'.

- **question answer**
  Is this image relevant to either text1 or text2 or not?
  Text1: question

Text2: answer

Reply 'Yes' only if this image is relevant to either of the texts, otherwise reply 'No'.

After the generation, we check whether the generated response contains either the word 'yes' or the phrase 'is relevant'; if so, we predict the input image as relevant; in all other cases, we classify the input image as irrelevant.

Using LLM to shorten the text

One of the disadvantages of the *CLIP* and *SigLIP* models is their context window size. For *SigLIP*, the context window size is 64; for *CLIP*, it is 77. This means that many answers could be truncated, resulting in missing information; this is especially true for the answers from the semi–automatically curated data set (the generated answers from *ChatGPT* tend to be wordier). Therefore, we decided to conduct an experiment in which we shortened the answers using an LLM in case the answers had more tokens than allowed. As a shortening LLM we used *Phi–3*[63] (specifically the instruction–tuned variant *microsoft/Phi-3-mini-4k-instruct*[13] with 3.8B parameters). Here is an example of the utilized input prompt (including the generation tokens and an answer) and the corresponding shortened answer:

- **Input prompt**
  <s><|user|> Shorten the following text while keeping the important information. Don't write comments of any kind. Write only the shorter text.
  Text: The driver and front passenger airbags deploy during significant frontal and near frontal crashes. The airbags are a supplemental restraint system and are designed to work with the seatbelts to help protect the driver and right front passenger from certain upper body injuries. Airbags do not inflate slowly; there is a risk of injury from a deploying airbag. Note: You will hear a loud bang and see a cloud of harmless powdery residue if an airbag deploys. This is normal. The airbags inflate and deflate rapidly upon activation. After airbag deployment, it is normal to notice a smoke–like, powdery residue or smell the burnt propellant.<|end|><|assistant|>

- **Response shorten answer**
  Front and near frontal crashes trigger driver and front passenger airbags, which complement seatbelts to protect against upper body injuries. Airbags inflate quickly, posing a risk of injury. A loud bang and harmless residue are normal upon deployment, with subsequent smoke–like residue or burnt propellant smell.

The advantage of shortening longer answers is that we can include more information in the *CLIP* and *SigLIP* context window. Using the generated summaries, we can drastically reduce

---

[13] `https://huggingface.co/microsoft/Phi-3-mini-4k-instruct`

the number of texts that need to be truncated during pre–processing, concretely from 294 to 29 in case of *CLIP* (that is a reducement of 90%), and from 311 to 7 in case of *SigLIP* (that is a reducement of almost 98%). On the other hand, the summarization can omit information that is necessary for answering the provided user question.

Trained models

We chose to attempt to train *CLIP*, *SigLIP*, and *Idefics2* models to improve their performance on the task further. Since both the *CLIP* and *SigLIP* have too small context window, we could experiment with increasing the context size and pre–training the models with longer context window. However, we opted not to do this as we believed we needed more data, so we chose to do full fine–tuning of all weights as the models are relatively small. We utilized the *CLIP loss* function described in the related work chapter to fine–tune both models. For the *SigLIP* model, we initially tried the *Sigmoid loss*. However, the results were worse than when using *CLIP loss*. We used a training set for training the fine–tuning and the validation set for finding the threshold used during inference. For fine–tuning the of *Idefics2* model, we decided to utilize *LoRA*[28] since the combined data set is still relatively small, and more importantly, the model is large (8 billion parameters).

3.3.2 Icon Captioning Data Set

To evaluate state–of–the–art VLMs on the task of icon captioning, we curated a data set from the manuals, which we refer to as an icon captioning data set. We extracted icons from all the manuals and annotated 300 of them[14]. We annotated the function and appearance of each icon. Examples from this data set are shown in figure 3.8.

Used Metric

Since our icon captioning task is an instance of an image captioning task, we used the *CIDEr score* as the primary indicator of a model's performance. Additionally, we also computed classical statistical metric *BLEU* [64], *METEOR* [65] and *ROUGE L* [66].

BLEU *BLEU* is a statistical metric measuring an n-gram overlap between a reference and candidate strings. It is widely used for the automatic evaluation of machine translation models [2].

---

[14] Cooperative work with fellow students Bc. Adam Černý and Ing. Ondřej Kobza

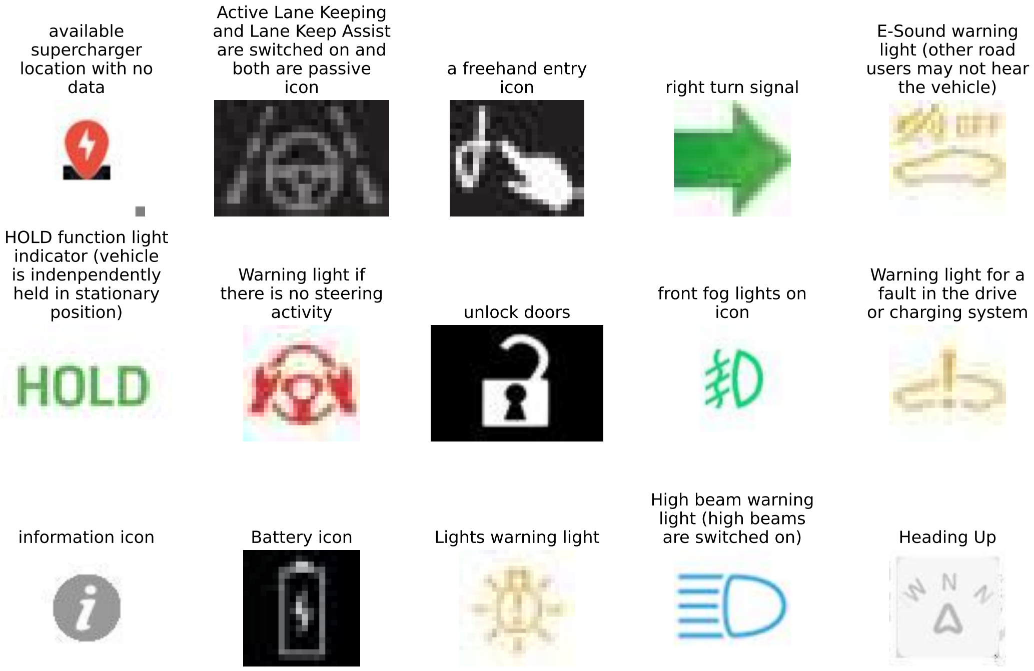

available supercharger location with no data
Active Lane Keeping and Lane Keep Assist are switched on and both are passive icon
a freehand entry icon
right turn signal
E-Sound warning light (other road users may not hear the vehicle)
OFF
HOLD function light indicator (vehicle is indenpendently held in stationary position)
HOLD
Warning light if there is no steering activity
unlock doors
front fog lights on icon
Warning light for a fault in the drive or charging system
information icon
Battery icon
Lights warning light
High beam warning light (high beams are switched on)
Heading Up
N

Figure 3.8: Icon captioning data set preview (only showing the icons' function and their appearance)

METEOR *METEOR* is another metric for automatically evaluating machine translation models. It is based on computing a harmonic mean of *precision* and *recall* of words (unigrams) alignment.

ROUGE L *ROUGE L* scores two text segments based on the longest common subsequence.

Data Set Splits

We randomly divided the samples from the icon captioning data set into training, validation, and test sets. Table 3.6 shows the split sizes with statistics about the resolutions of the icons.

Split	**# of Samples**	**Image Resolution [px]**		
		Min	**Median**	**Max**
Train	200	10x10	28x28	90x90
Validation	40	14x7	28x27	80x80
Test	60	11x10	30x29	85x99

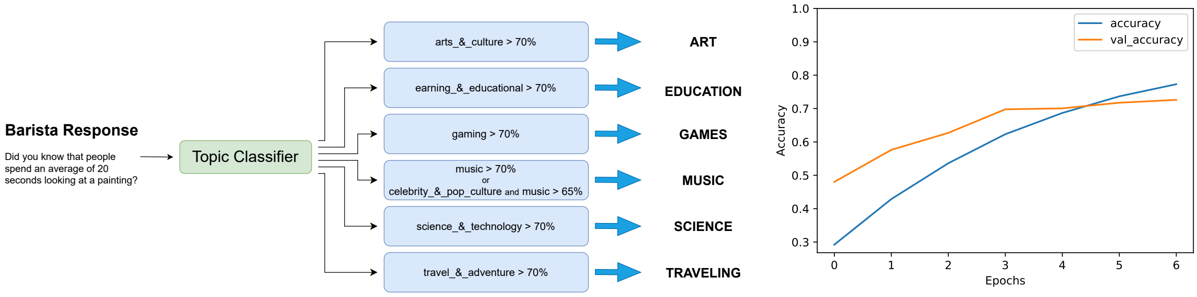
Table 3.6: Analysis of the icon captioning data set

Evaluated Models

As baseline models, we chose the generative VLMs *GIT*, *BLIP*, *InstructBLIP*, *LlaVA–NeXT*, *Idefics2* and as a reference model, we again used *GPT–4o*. For *GIT* and *BLIP* models, no prompt was necessary, as this is how they were trained for image captioning. To see the prompts used for all the other VLMs on this task, please refer to Appendix C.1.

3.3.3 Plot Recognition Data Set

In order for us to experiment with plots, we gathered 160 images from which 60 are plots. These images and plots were taken from the selected manuals, our solutions for school assignments, as well as from these works [67], [68] and this repository [69]. Figure 3.9 shows examples of a plot and a non-plot image.



(a) An example of a non–plot image. (b) An example of a plot image.

Figure 3.9: Examples from the curated plot recognition data set

Evaluated Models

To be able to use the *DePlot* model to translate plots into text, we first needed to determine which image is a plot and which is not. For this task, we evaluated *LlaVA–NeXT* together with *Idefics2* models as both models showcased strong performance out of the open–sourced VLMs on the VQA tasks (table 2.1). The instruction for both these models was the following:

- *Decide whether this image is a chart, plot or something different (e.g. a diagram or flowchart). In case this image is either plot or chart, reply 'Yes', if the image is a diagram, flowchart or something else reply 'No'. Do not reply anything else.*

Chapter 4

Results and Discussion

This chapter presents and discusses the results of the experiments proposed in the previous chapter.

4.1 Used Libraries and Tools

Libraries

We developed this thesis in Python programming language in version 3[1]. Some of the most important employed Python packages include *PyTorch*[2], *Transformers*[3], *SentenceTransformers*[4] [70], *Datasets*[5], *Accelerate*[6], *Numpy*[7], *TRL*[8], *PEFT*[9], and *langchain*[10].

SentenceTransformers is a popular Python package for state–of–the–art text and image embeddings. However, they only supported the *CLIP* model, so while working on this thesis, we have added the support for the *SigLIP* models[11].

Tools

All the scripts were implemented in the Visual Studio Code[12] and Google Colab[13]. For tracking of the training processes, we utilized the wandb platform[14].

---

[1] `https://www.python.org`
[2] `https://pytorch.org`
[3] `https://huggingface.co/docs/transformers/index`
[4] `https://sbert.net`
[5] `https://huggingface.co/docs/datasets/index`
[6] `https://huggingface.co/docs/accelerate/index`
[7] `https://numpy.org`
[8] `https://huggingface.co/docs/trl/index`
[9] `https://huggingface.co/docs/peft/index`
[10] `https://www.langchain.com`
[11] `https://github.com/UKPLab/sentence-transformers/pull/2629`
[12] `https://code.visualstudio.com`
[13] `https://colab.research.google.com`
[14] `https://wandb.ai/site`

4.2 Used Hardware

For our work, we used several devices. We used either a personal laptop or a free version of Google Colab for quick prototyping. For more complex computation we used the following clusters: CIIRC Cluster[15], Barbora[16] and Karolina[17]. The following table 4.1 summarizes the used computational tools. Please note that we have requested and used much less RAM in the case of the clusters.

Device	GPU	CPU	RAM
Apple MacBook	10-core GPU	11-core Apple M3 Pro	18GB
Google Colab	Tesla T4 (15GB)	2x Intel Xeon, 2.30GHz	12.7GB
CIIRC Cluster	Tesla V100 (32GB)	2x Intel Xeon E5-2690 v4, 2.6GHz	256GB
Barbora Cluster	Tesla V100 (16GB)	2x Intel Skylake Gold 6126, 2.6GHz	192GB
Karolina Cluster	NVIDIA A100 (40GB)	2x AMD Zen 3 EPYC 7763, 2.45GHz	1024GB

Table 4.1: Used hardware

4.3 Used Models

Before diving into the experiment results, we will present the pre–trained models employed in our conducted experiments and their size. Table 4.2 presents the model names, sizes, and tasks they were used for. All models are accessible on HuggingFace[18]. Although, we used the original PyTorch *CLIP* weights from the OpenAI's repozitory[19]. Please refer to Appendix A for the complete model overview, including the links to the models' weights.

4.4 Image Text Relevance Task

In this section, we will describe the results of our experiments on the image–text–relevance task.

4.4.1 Baseline Results

The table 4.3 shows the baseline results for the selected models. The table shows only the best–performing mode on the test set for each model. Without any doubt, the best results achieves the *GPT–4o* model with a gap in *F1–score* of almost 20 percentage points between the

---

[15] `https://cluster.ciirc.cvut.cz`

[16] `https://www.it4i.cz/en/infrastructure/barbora`

[17] `https://www.it4i.cz/en/infrastructure/karolina`

[18] `https://huggingface.co`

[19] `https://github.com/openai/CLIP`

Model Name	**# of Params**	**Usage**
CLIP ViT–B/32 [16]	151M	ITR
CLIP ViT–B/16 [16]	150M	ITR
SigLIP base (patch16/224) [17]	203M	ITR
SigLIP base (patch16/256) [17]	203M	ITR
SigLIP base (patch16/384) [17]	203M	ITR
SigLIP base (patch16/512) [17]	204M	ITR
SigLIP so400m (patch14/384) [17]	878M	ITR
SigLIP large (patch16/256) [17]	652M	ITR
SigLIP large (patch16/384) [17]	652M	ITR
GIT base (TextVQA) [18]	177M	ITR
GIT large (TextVQA) [18]	395M	ITR
GIT base (VQAv2) [18]	177M	ITR
GIT large (VQAv2) [18]	395M	ITR
GIT large (TextCaps) [18]	395M	IC
GIT large (COCO) [18]	395M	IC
GIT base (TextCaps) [18]	177M	IC
GIT base (COCO) [18]	177M	IC
BLIP2 OPT 6.7B (COCO) [19]	7.96B	ITR, IC
BLIP2 OPT 2.7B [19]	3.87B	ITR, IC
BLIP2 FlanT5XL [19]	3.94B	ITR, IC
BLIP2 OPT 6.7B [19]	7.96B	ITR, IC
InstructBLIP FlanT5XL [23]	4.02B	ITR, IC
InstructBLIP Vicuna 7B [23]	7.91B	ITR, IC
LlaVA–NeXT Mistral 7B [31]	7.57B	ITR, IC, PD
LlaVA–NeXT 34B [31]	34.8B	ITR, IC
Idefics2 8B [36]	8.4B	ITR, IC, PD
deplot [37]	282M	PT

Table 4.2: Utilized model variants, their size and the tasks they were utilized for (*ITR*: stands for image–text–relevance task, *IC*: stands for icon captioning, *PD*: stands for plot detection, and *PT*: stands for plot translation, the names in the parentheses represent the tasks onto which the models were fine–tuned, ***patchX/Y*** means that the given model was pre–trained on images with $Y \times Y$ resolution and it splits input images into patches of size $X \times X$)

second best model *Idefics2*. Such a massive difference in performance is sporadic these days. The open–sourced LLMs and VLMs are slowly closing in on their proprietary rivals. Other models showcasing promising results include *CLIP*, *SigLIP*, and *LlaVA–NeXT*. Unsurprisingly, both *CLIP* and *SigLIP* perform relatively okay on this task since, as we already stated, this task could be considered as an instance of an image retrieval problem. By far, the worst performance showcased all variants of the *GIT* model except for the *GIT large (VQAv2)* version, which performs relatively well in comparison with other models. Even though the *GIT* models were trained on VQA tasks (either *VQAv2* or *TextVQA* [71]), they were not able to cope with the task and generated only meaningless texts. The *BLIP2* and *InstructBLIP* models are better than most of the *GIT* variants, but still, the results are not usable.

The results show that the optimal mode can be different for each model. It depends on the training data on which the specific model was trained. However, the best–performing mode overall was the *answer* mode. The best results were achieved with this mode, so we further focused mainly on it.

Model Name	**Mode**	**Accuracy** [%]	**F1**	**Precision**	**Recall**
CLIP ViT–B/32	A (0.27)	71.90	51.43	46.15	58.06
CLIP ViT–B/16	A (0.26)	67.77	55.17	42.86	77.42
SigLIP base (patch16/224)	A (0.16)	79.34	50.98	65.00	41.94
SigLIP base (patch16/256)	A (0.11)	78.51	48.00	63.16	38.71
SigLIP base (patch16/384)	maxQA (0.15)	80.17	55.56	65.22	48.39
SigLIP base (patch16/512)	maxQA (0.17)	78.51	48.00	63.16	38.71
SigLIP so400m (patch14/384)	maxQA (0.50)	82.64	63.16	69.23	58.06
SigLIP large (patch16/256)	maxQA (0.14)	80.17	61.29	61.29	61.29
SigLIP large (patch16/384)	A (0.17)	83.47	65.52	70.37	61.29
GIT base (TextVQA)	Q	74.38	0.00	0.00	0.00
GIT large (TextVQA)	Q	74.38	0.00	0.00	0.00
GIT base (VQAv2)	Q	74.38	0.00	0.00	0.00
GIT large (VQAv2)	A	75.21	48.28	51.85	45.16
BLIP2 OPT 6.7B (COCO)	A	36.36	38.40	25.53	77.42
BLIP2 OPT 2.7B	A	29.75	38.85	25.0	87.10
BLIP2 FlanT5XL	QA	64.46	49.41	38.89	67.74
BLIP2 OPT 6.7B	Q	48.76	41.51	29.33	70.97
InstructBLIP FlanT5XL	QA	55.37	50.91	35.44	90.32
InstructBLIP Vicuna 7B	Q	62.81	47.06	37.04	64.52
LlaVA–NeXT Mistral 7B	A	79.34	64.79	57.50	74.19
LlaVA–NeXT 34B	A	71.07	63.16	46.88	96.77
Idefics2 8B	A	78.51	67.50	55.10	87.10
GPT-4o	A	**92.56**	**86.96**	**78.95**	**96.77**

Table 4.3: A table showing the results of the baseline VLMs on the image–text–relevance data set (for *CLIP* and *SigLIP* models, the number in the parentheses in the column *mode* represents the value of the automatically found threshold)

4.4.2 Using LLM to shorten the text

The tables 4.4 and 4.5 show the results of the pre–trained *CLIP* and *SigLIP* models with long answers replaced by their summaries. Please note that in the case of *maxQA* and *meanQA* modes, only the answers were replaced by their summaries if they were too long.

The results are pretty interesting; for *CLIP ViT–B/32*, the *F1–score* while using summaries is smaller than without them. For *CLIP ViT–B/16* and the *maxQA* mode, the *F1–score* while using summaries is significantly higher than without. Actually, the score for the experiment without the summaries is pretty low, which is interesting. The low *recall* suggests that the model predicts positive class very sparsely. In the case of the selected two *SigLIP* models, the *F1–score* is higher for the *SigLIP so400m (patch14/384)* with summaries and the *A* mode, otherwise the *F1–score* is smaller or equal (in case of *SigLIP large (patch16/384)* and the *maxQA* mode). Overall, the *SigLIP* models seem more stable because the difference in the metrics for experiments with and without summaries is minor. In contrast, for the selected two *CLIP* variants, the difference in performance between the two settings is more prominent. These results indicate that using summaries might be beneficial in some cases; however, further studying would be required. Another and most likely a better solution would be to pre–train both *CLIP* and *SigLIP* with longer context window sizes to overcome the problem with small context windows.

Model Name	**Mode**	**Accuracy** [%]	**F1**	**Precision**	**Recall**
CLIP B/32 `w\o` summaries	A (0.27)	71.90	**51.43**	46.15	**58.06**
CLIP B/32 w summaries	A (0.29)	**74.38**	24.39	**50.00**	16.13
CLIP B/32 `w\o` summaries	maxQA (0.27)	70.25	**51.35**	44.19	**61.29**
CLIP B/32 w summaries	maxQA (0.29)	**73.55**	30.43	**46.67**	22.58
CLIP B/32 `w\o` summaries	meanQA (0.27)	**73.55**	**46.67**	**48.28**	**45.16**
CLIP B/32 w summaries	meanQA (0.28)	71.90	26.09	40.00	19.35
CLIP B/16 `w\o` summaries	A (0.26)	67.77	**55.17**	42.86	**77.42**
CLIP B/16 w summaries	A (0.29)	**80.99**	54.90	**70.00**	45.16
CLIP B/16 `w\o` summaries	maxQA (0.32)	75.21	11.76	**66.67**	6.45
CLIP B/16 w summaries	maxQA (0.29)	**79.34**	**56.14**	61.54	**51.61**
CLIP B/16 `w\o` summaries	meanQA (0.28)	76.03	**49.12**	53.85	**45.16**
CLIP B/16 w summaries	meanQA (0.29)	**76.86**	39.13	**60.00**	29.03

Table 4.4: Comparison of the CLIP results when using summaries vs. not (please note that *CLIP B/32* refers to *CLIP ViT–B/32*, and *CLIP B/16* refers to *CLIP ViT–B/16*)

4.4.3 Fine–tuning of CLIP and SigLIP

Table 4.6 presents the contribution of the training of *CLIP* models on the task data. To see the utilized hyperparameters and their values, please refer to the Appendix to the section B.3. The trained *CLIP* models were uploaded to HuggingFace here and here.

Table 4.7 presents the contribution of training of *SigLIP* models on the task data. To see the utilized hyperparameters and their values, please refer to the Appendix to the section B.4. The trained *SigLIP* models were uploaded to HuggingFace here and here.

As can be seen from the two tables, the additional training has improved the models' performance on our task by approximately 7 percentage points in *F1–score*. The interesting thing

Model Name	Mode	Accuracy [%]	F1	Precision	Recall
siglip-so-p14 `w\o` summaries	A (0.18)	**79.34**	61.54	**58.82**	64.52
siglip-so-p14 w summaries	A (0.19)	78.51	**63.89**	56.10	**74.19**
siglip-so-p14 `w\o` summaries	maxQA (0.50)	**82.64**	**63.16**	**69.23**	58.06
siglip-so-p14 w summaries	maxQA (0.19)	74.38	59.74	50.00	**74.19**
siglip-so-p14 `w\o` summaries	meanQA (0.25)	**82.64**	**63.16**	**69.23**	58.06
siglip-so-p14 w summaries	meanQA (0.13)	76.03	62.34	52.17	**77.42**
SigLIP L/16 `w\o` summaries	A (0.17)	**83.47**	**65.52**	70.37	**61.29**
SigLIP L/16 w summaries	A (0.13)	82.64	60.38	**72.73**	51.61
SigLIP L/16 `w\o` summaries	maxQA (0.17)	**80.17**	**62.50**	**60.61**	**64.52**
SigLIP L/16 w summaries	maxQA (0.10)	80.17	**62.50**	**60.61**	**64.52**
SigLIP L/16 `w\o` summaries	meanQA (0.17)	**82.64**	**63.16**	**69.23**	**58.06**
SigLIP L/16 w summaries	meanQA (0.17)	81.82	60.71	68.00	54.84

Table 4.5: Comparison of the SigLIP results when using summaries and when not (please note that *siglip-so-p14* refers to *SigLIP so400m (patch14/384)* and *SigLIP L/16* to *SigLIP large (patch16/384)*)

Model Name	Mode	Accuracy [%]	F1	Precision	Recall
CLIP B/32	A (0.27)	71.90	51.43	46.15	58.06
CLIP B/32 +T	A (0.26)	**72.73** (+0.83)	**60.24** (+8.81)	**48.08** (+1.93)	**80.65** (+22.59)
CLIP B/16	A (0.26)	67.77	55.17	42.86	**77.42**
CLIP B/16 +T	A (0.26)	**78.51** (+10.74)	**62.86** (+7.69)	**56.41** (+13.55)	70.97 (-6.45)

Table 4.6: Table showing the contribution of training the CLIP models in comparison with the non–trained versions (please note that *+T* stands for training, *CLIP B/32* refers to *CLIP ViT–B/32*, and *CLIP B/16* refers to *CLIP ViT–B/16*)

Model Name	Mode	Accuracy [%]	F1	Precision	Recall
siglip-so-p14	A (0.18)	79.34	61.54	58.82	**64.52**
siglip-so-p14 +T	A (0.23)	**85.12** (+5.78)	**68.97** (+7.43)	**74.07** (+15.25)	**64.52** (+0.00)
SigLIP L/16	A (0.17)	83.47	65.52	70.37	61.29
SigLIP L/16 +T	A (0.10)	**86.78** (+3.31)	**72.41** (+6.89)	**77.78** (+7.41)	**67.74** (+6.45)

Table 4.7: Table showing the contribution of training the CLIP models in comparison with the non–trained versions (please note that *+T* stands for training, and that *siglip-so-p14* refers to *SigLIP so400m (patch14/384)* and *SigLIP L/16* to *SigLIP large (patch16/384)*)

to notice is the decrease in the *recall* for the *CLIP ViT–B/16* model. It is an excellent example of a trade–off between *precision* and *recall* meaning that massive gain in one metric can result in a decrease in the other, but not necessarily as can be seen in the case of *CLIP ViT–B/32*, where the model gained 22.59 percentage points in the *recall*, which is a vast improvement, while still gaining almost 2 percentage points in *precision*. Another example of the *precision–recall* trade–off is visible in the case of *SigLIP so400m (patch14/384)* model, which gained 15.25 percentage points in *precision* while not gaining anything in *recall*.

4.4.4 Fine–tuning of Idefics2

Table 4.8 shows the results of the fine–tuned *Idefics2* model using the *LoRA* algorithm compared with the non–trained variant. As can be seen from the table, the additional training helps, and the model achieves better results. Although, the performance gain is not as significant as we have seen for both *CLIP* and *SigLIP*. This shows that there could still be room for improvement if a better combination of hyperparameters was used. To see the complete list of utilized hyperparameters and their values, please refer to the Appendix to the section B.5. The trained model is available here at HuggingFace.

Model Name	**Mode**	**Accuracy** [%]	**F1**	**Precision**	**Recall**
Idefics2 8B	A	78.51	67.50	55.10	**87.10**
idefics2 8B + LoRA	A	**81.82** (+3.31)	**71.05** (+3.55)	**60.0** (+4.90)	**87.10** (+0.00)

Table 4.8: A table showing the contribution of training the Idefics2 model in comparison with the non–trained version

4.4.5 Final Results

Table 4.9 presents the final results obtained on the image–text–relevance data set. The models solving the image–text–relevance task are crucial in the proposed multi–modal RAG pipeline. Therefore, our primary focus was on this task. The best results overall achieves the proprietary *GPT–4o* model. The results of it are quite astonishing. The second best model based on our targeted *F1–score* is the trained version of *SigLIP large (patch16/384)* followed by the trained *Idefics2 8B* model. The training helped to narrow the performance gap between our models and the *GPT–4o*. However, the difference is still significant. This could mean that our task is complex and not easily solvable because the models need to pay attention to details, and so far, only *GPT–4o* is genuinely able to solve it. However, our data sets were still relatively small, meaning that with additional training, even our models could achieve comparable results as *GPT–4o*. Furthermore, these results show that even a smaller, faster, and more specialized model such as *SigLIP* can perform better than a much more extensive size, a slower but more general model such as *Idefics2*. To conclude, we recommend using the *GPT–4o* model in our pipeline or the trained *SigLIP large (patch16/384)* version in the case *GPT–4o* is too expensive to use.

Model Name	**Mode**	**Accuracy** [%]	**F1**	**Precision**	**Recall**
CLIP ViT–B/32 +T	A (0.26)	72.73	60.24	48.08	80.65
CLIP ViT–B/16 +T	A (0.26)	78.51	62.86	56.41	70.97
siglip-so-p14 +T	A (0.23)	85.12	68.97	74.07	64.52
SigLIP L/16 +T	A (0.10)	86.78	72.41	77.78	67.74
idefics2 +LoRA	A	81.82	71.05	60.0	87.10
GPT-4o	A	**92.56**	**86.96**	**78.95**	**96.77**

Table 4.9: A table showing the final results on the image–text–relevance data set

4.5 Icon Captioning

The table 4.10 shows the results of the selected VLMs on the icon captioning task. The best performing models are *GPT–4o*, *BLIP2 FlanT5XL*, *InstructBLIP FlanT5XL*, *LlaVA–NeXT Mistral 7B* and *Idefics2 8B*. They all perform very similarly. The results of *InstructBLIP FlanT5XL* are in agreement with the results on the *NoCaps* and *Flick30K* shown in table 2.1, where the model performed also very well. On the other hand, all *GIT* versions again did not perform very well. An interesting thing to notice is that it matters which LM is employed inside a VLM because, as the results show, whenever a VLM employs the *FlanT5–XL* model, it performs very well on image captioning tasks, even beating the *InstructBLIP Vicuna 7B* model with almost twice the number of parameters.

Model Name	**BLEU 1**	**BLEU 2**	**METEOR**	**ROUGE L**	**CIDEr**
GIT large (TextCaps)	26.96	14.71	11.19	24.32	31.36
GIT large (COCO)	24.25	12.25	9.09	21.72	24.19
GIT base (TextCaps)	23.83	12.31	11.27	23.80	32.67
GIT base (COCO)	25.00	10.07	8.39	21.41	23.71
BLIP2 OPT 2.7B	23.87	10.29	11.07	20.79	35.10
BLIP2 FlanT5XL	27.42	15.86	13.99	29.09	52.69
BLIP2 OPT 6.7B	22.77	11.89	11.85	22.33	33.72
BLIP2 OPT 6.7B (COCO)	24.44	12.16	13.49	25.91	32.71
InstructBLIP FlanT5XL	**29.81**	16.13	**15.34**	28.72	47.53
InstructBLIP Vicuna 7B	27.50	14.70	12.75	26.41	43.69
LlaVA–NeXT Mistral 7B	27.66	14.48	14.44	**32.69**	49.00
Idefics2 8B	28.50	15.28	11.67	24.52	49.15
GPT–4o	27.65	**16.85**	12.30	24.20	**53.29**

Table 4.10: A table showing the results on the icon captioning data set (*BLEU N* means that the overlap is computed across geven N-grams, e.g. $N = 2$ means bigrams are considerated)

4.6 Icon Localization

As explained in the previous chapter, the localization of the correct icon's position in a text is a complex task, and the proposed algorithm 1 can either miss some icons or mistakenly localize them. Because of this, we designed an experiment in which we skimmed through the Porsche Taycan and Ford Mustang Mach E manuals and manually labeled which icons were localized correctly, which were not, and which were not localized at all. We then computed *accuracy*, *precision*, *recall*, and *F1–score*. Table 4.11 presents the measured values. The average *precision* on the two manuals is slightly above 90 percent, which means that only less than 10 percent of icons are wrongly localized. The average *recall* is approximately 83 percent, meaning around 17 percent of icons are undetected. Please note that a more significant sample than two manuals would be required to measure the algorithm's performance more objectively. However, we did not have enough time for this, so we leave this as a possible improvement for future work.

Manual	**# of Icons**	**Detected Icons**	**Accuracy** [%]	**F1**	**Precision**	**Recall**
Porsche Taycan	819	724 (88%)	79.61	88.65	90.06	87.28
Ford Mach E	470	380 (81%)	73.19	84.52	90.53	79.26
AVG	664.50	84.50%	76.40	86.59	90.30	83.27

Table 4.11: Results of the icon localization algorithm1

4.7 Plots

This section will present the results of experiments conducted with plots. We will start with plot recognition, followed by an experiment designed to measure the quality of the translated charts.

4.7.1 Plots Recognition

Table 4.12 shows the models' performance on these images. The table tells us that the *LlaVA–NeXT Mistral 7B* model outperforms *Idefics2* by quite some margin. Even though the models are similar in architecture, the difference in their performance is enormous. This proves that the data onto which models are pre–trained and fine–tuned play a crucial role in a model's performance.

Model Name	Accuracy [%]		
	Figures	Plots	Overall
LlaVA–NeXT Mistral 7B	**98.00%**	**98.33%**	**98.12%**
Idefics2 8B	88.00%	80.00%	85.00%

Table 4.12: Model results for plot recognition

4.7.2 Plot to Text Translation

We evaluated the 60 plots we accumulated to measure the plot translation's quality. For translation of a particular chart, we used the following instruction: *"Generate underlying data table of the figure below:"*. Out of the 60 plots, 25 were correctly translated into a text. 14 plots had either negligible or slightly bigger mistakes in the translation, but nothing serious. Lastly, in 25 cases, the translation was utterly wrong. This results in an accuracy of 41.67% and weak accuracy (considering all translations that were at least a bit correct) of 65.00%. While evaluating the *DePlot* model, we learned how well it performs on different types of plots. Namely, the *DePlot* model excels on bar plots or histograms (as seen on the chart 4.1), and it can also handle very well simple line charts. Pie charts are also handled very well, except for one mistake when the title is below the pie chart very closely to the chart. The model mistakes the title as another piece of the chart (as seen on the chart 4.2a with the incorrect part highlighted in red). *DePlot* is not able to handle the translation of confusion matrices (as can be seen on the chart 4.2b) and some complex plots (e.g., a plot with 10 lines).

4.8 Multi–modal RAG Pipeline Implementation

In this part, we put it all of our results together and implemented the proposed proposed multi–modal RAG pipeline. The implementation is available at GitHub[20]. We chose models for each component based on their performance and implemented the proposed multi–modal RAG pipeline. Here are the chosen components:

- **Text embedding model**: *BGE base*[21] [72]
- **Icon captioning model**: *InstructBLIP FlanT5XL*[22]
- **LLM for generating answers**: *Mistral 7B*[23]
- **Plot Recognizer**: *LlaVA–NeXT Mistral 7B*[24]

---

[20] `https://github.com/HonzaCuhel/multimodal-rag-pipeline`

[21] `https://huggingface.co/BAAI/bge-base-en-v1.5`

[22] `https://huggingface.co/Salesforce/instructblip-flan-t5-xl`

[23] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`

[24] `https://huggingface.co/llava-hf/llava-v1.6-mistral-7b-hf`

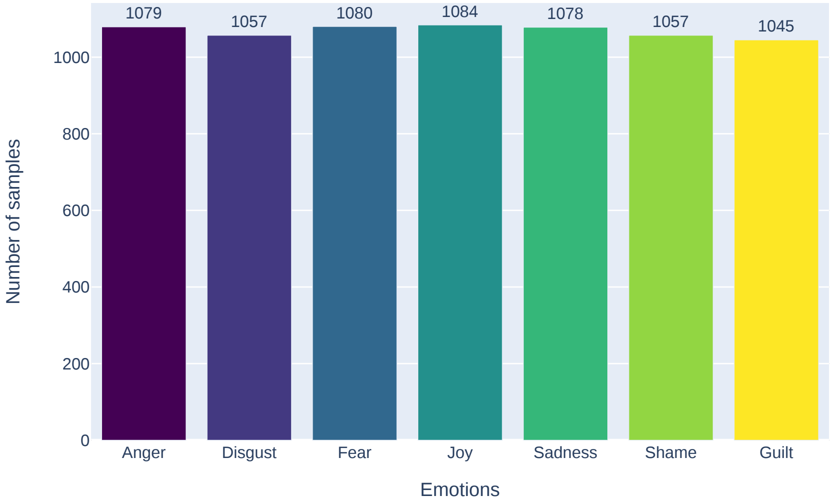
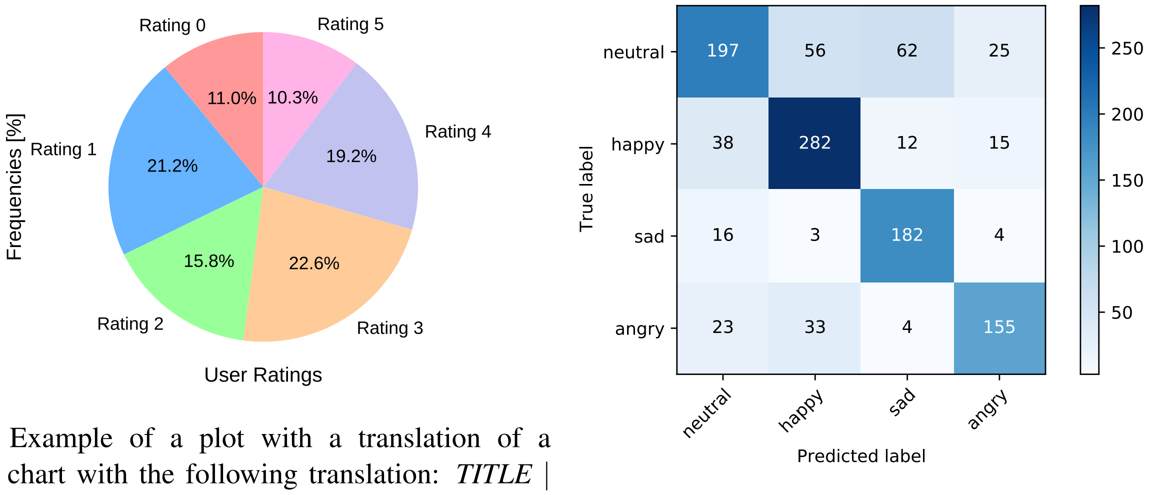


Figure 4.1: Example of a plot with a correct translation that is following: *TITLE | <0x0A> Emotions | Number of samples <0x0A> Anger | 1079 <0x0A> Disgust | 1057 <0x0A> Fear | 1080 <0x0A> Joy | 1084 <0x0A> Sadness | 1078 <0x0A> Shame | 1057 <0x0A> Guilt | 1045*



(a) Example of a plot with a translation of a pie chart with the following translation: *TITLE | <0x0A> Frequencies [%] | 0.1% <0x0A> Rating 0 | 11.0% <0x0A> Rating 1 | 21.2% <0x0A> Rating 2 | 15.8% <0x0A> User Ratings | 22.6% <0x0A> Rating 3 | 22.6% <0x0A> Rating 4 | 19.2% <0x0A> Rating 5 | 10.3%*

(b) Example of a plot with a incorrect translation with the following translation: *TITLE | <0x0A> Predicted label | True label <0x0A> neutral | 197 <0x0A> happy | 56 <0x0A> sad | 182 <0x0A> angry | 155*

- **Plot Translator**: *DePlot*[25]
- **Vector Store**: *FAISS* [73]

---

[25] `https://huggingface.co/google/deplot`

- **VLM**: our trained variant of *SigLIP large (patch16/384)*[26] with threshold 0.10

Please note that we could chose any text embedding and LLM models trained in English. We chose these two models because of their popularity. Moreover, if we opted to use *GPT–4* instead of the *InstructBLIP* and *SigLIP* models, we should expect even better results.

---

[26] `https://tinyurl.com/2ee6avv6`

Chapter 5

Conclusion

In this thesis, we have proposed a general multi–modal RAG pipeline, which incorporates knowledge from icons (small images), plots, and other images. Even though the proposed pipeline is general, we focused on improving the quality of question–answer systems that let users ask questions about information from technical manuals. We described the components of the proposed pipeline that are responsible for image integration. The crucial components include an icon localizer and captioner, a Visual–language model (VLM) capable of finding relevant images to a given answer/question, and a plot recognizer. We have presented candidate models for these components and proposed a simple algorithm for icon localization based on matching text around an icon's coordinates. We created data sets that we used to model training and evaluation. As a reference model, we chose *GPT–4o*, the current state–of–the–art model on many tasks, which performed the best on both the image–text–relevance and icon captioning tasks. After the *GPT–4o*, the best results on the image–text–relevance task showed trained variants of *SigLIP* and *Idefics2 8B* models. On the icon captioning task, the *InstructBLIP FlanT5XL* model performed comparably with the reference *GPT–4o* model, even outperforming it on *BLEU 1* and *METEOR* metrics. The results showed that one of the most crucial aspects of a well–performing VLM is its LM, which is employed inside. For image captioning tasks, that LLM is any model from the *FlanT5* model family. Our proposed icon localization algorithm achieves precision slightly over 90 percent. Furthermore, we have implemented the proposed multi–modal RAG pipeline and contributed to the open–source world by implementing *SigLIP* support to a popular *SentenceTransformers* Python library.

5.1 Future Work

There is certainly room for future work. In this work, we did not manage to evaluate the performance of the implemented multi–modal pipeline. Moreover, it would be interesting to compare our implemented solution against the existing ones. Another area we could improve in the future is to train a model on icon captioning. This should improve the quality of the generated icon captions, possibly even outperforming the *GPT–4* model. Another idea would be to collect more manuals to enlarge the created data sets. Instead of computing a cosine similarity of *CLIP* and *SigLIP* embeddings to determine whether a given text is relevant to a given image, we could train a simple feed–forward neural network to predict the likelihood of the relevancy of the text w.r.t. the image.

Furthermore, we could focus on providing better support for other languages. We suggested a possible multilingual solution. However, the ratio of non-English languages, such as Czech, in the training data of these multilingual models is usually low. Therefore, we could focus on developing a better solution for specific languages. This could include creating a custom Vision–Language models for those specific languages. Another possible experiment is to feed multiple images at once to VLMs tasked with solving the image–text–relevance problem. Lastly, we could also improve the proposed RAG pipeline by adding a model capable of highlighting the image's relevant segment to which the generated answer refers. This would result in an even more user-friendly QA system.

Appendix A

Model Overview

Model Name	**Weights Source**	**# of Params**	**Data Type**	**Latency** [ms]
CLIP ViT–B/32	OpenAI	151M	fp32	16±1
CLIP ViT–B/16	OpenAI	150M	fp32	15±1
SigLIP base (patch16/224)	HuggingFace	203M	fp32	14±2
SigLIP base (patch16/256)	HuggingFace	203M	fp32	14±1
SigLIP base (patch16/384)	HuggingFace	203M	fp32	21±1
SigLIP base (patch16/512)	HuggingFace	204M	fp32	37±2
SigLIP so400m (patch14/384)	HuggingFace	878M	fp32	74±2
SigLIP large (patch16/256)	HuggingFace	652M	fp32	31±2
SigLIP large (patch16/384)	HuggingFace	652M	fp32	51±1
GIT base (TextVQA)	HuggingFace	177M	fp16	23±6
GIT large (TextVQA)	HuggingFace	395M	fp16	35±18
GIT base (VQAv2)	HuggingFace	177M	fp16	23±2
GIT large (VQAv2)	HuggingFace	395M	fp16	36±8
GIT large (TextCaps)	HuggingFace	394M	fp16	20±3
GIT large (COCO)	HuggingFace	394M	fp16	42±74
GIT base (TextCaps)	HuggingFace	177M	fp16	15±4
GIT base (COCO)	HuggingFace	177M	fp16	17±7
BLIP2 OPT 6.7B (COCO)	HuggingFace	7.96B	fp16	91±79
BLIP2 OPT 2.7B	HuggingFace	3.87B	fp16	71±34
BLIP2 FlanT5XL	HuggingFace	3.94B	fp16	90±45
BLIP2 OPT 6.7B	HuggingFace	7.96B	fp16	94±52
InstructBLIP FlanT5XL	HuggingFace	4.02B	fp16	93±3
InstructBLIP Vicuna 7B	HuggingFace	7.91B	fp16	143±76
LlaVA–NeXT Mistral 7B	HuggingFace	7.57B	fp16	271±067
LlaVA–NeXT 34B	HuggingFace	34.8B	int4	1 656±303
Idefics2 8B	HuggingFace	8.4B	fp16	236±134
DePlot	HuggingFace	282M	fp32	1 990±1 937

Table A.1: Used pre–trained models, link to their weights and their latency on a batch with a single sample measured across around a hundred samples (except for the DePlot model) on a device with one NVIDIA Tesla A100 GPU with 40GB of memory

Appendix B

Image Text Relevance Task

B.1 Semi–automatically Curated Data Set: Utilized Questions

Here is the list of utilized questions during the creation of the semi–automatically curated image–text–relevance data set:

1. How do I turn on the cruise control?
2. How to use Siri?
3. How to connect an Android phone?
4. How to unlock the bonnet?
5. Is it possible to unlock the bonnet with the key?
6. How do I adjust the regenerative braking settings?
7. What are the different driving modes available?
8. How to change a driving mode?
9. How can I schedule charging times for my car?
10. What is the maximum range of on a single charge?
11. Where can I see the remaining range?
12. How do I activate the lane-keeping assist feature?
13. Can I customize the interior lighting settings?

14. How do I program the memory seats in my car?
15. What are the maintenance intervals and procedures?
16. How do I pair multiple phones with the car?
17. How do I use the car’s navigation system?
18. Can I disable the automatic start-stop function?
19. How do I adjust the ride height or suspension?
20. How do I lock and secure the charging port?
21. How to enable or disable the car’s sport sound?
22. How to use the car’s efficiency monitoring feature?
23. How do I engage the car’s launch control?
24. How do I activate the car’s parking assist?
25. How to pair garage door opener with HomeLink?
26. Can the car be locked remotely using an app?
27. How do I use the car’s adaptive headlights?
28. How do I activate the ambient lighting?
29. How to set up the car’s Wi-Fi hotspot?
30. Can I configure different driver profiles?
31. What is the sport driving mode called?
32. How to use a wireless charging pad for a phone?
33. Can I set a maximum charge limit?
34. How do I adjust the side mirrors?
35. How do I use the car’s energy recovery system?
36. How to enable the remote climate control?
37. How to customize the steering wheel?
38. Can I set up a charging schedule?

39. What should I do if I experience a loss of power?
40. How do I enable the car's rear-wheel steering?
41. How to set up the collision avoidance system?
42. How do I maintain the cabin air filters?

B.2 Utilized Prompts

Here are the utilized prompts for the answer mode for each model in the image–text–relevance task:

- **All GIT variants**
  Is this image relevant to the following text or not?
  Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'.

- **All BLIP2 variants**
  Question: Is this image relevant to the following text or not?
  Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'. Answer:

- **All InstructBLIP variants**
  Is this image relevant to the following text or not? Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'.

- **LlaVA–NeXT Mistral 7B**
  [INST] <image>
  Is this image relevant to the following text or not?
  Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'. [/INST]

- **LlaVA–NeXT 34B**
  <|im_start|>system
  Answer the questions.<|im_end|><|im_start|>user
  <image>
  Is this image relevant to the following text or not? Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'. <|im_end|>
  <|im_start|>assistant

- **Idefics2 8B**
  User:<image> Is this image relevant to the following text or not?
  Text: ...
  Reply 'Yes' only if this image is relevant to the text, otherwise reply 'No'.<end_of_utterance>
  Assistant:

B.3 Fine–tuning CLIP

The CLIP models were trained on a device with four Tesla V100 GPUs, each with 16GB of memory. The table B.1 presents the utilized hyperparameters.

Hyperparameter	**ViT-B-32**	**ViT-B-16**
Train batch size	128	128
Evaluation batch size	64	64
Number of train epochs	4	13
Optimizer	Adam [74]	Adam [74]
Learning rate	5e-06	5e-06
$\beta 1$	0.9	0.9
$\beta 2$	0.98	0.98
ϵ	1e-6	1e-6
Weight decay	0.2	0.2
Seed	42	42
Data type	fp32	fp32

Table B.1: Hyperparameters used for training the CLIP models

B.4 Fine–tuning SigLIP

The SigLIP models were trained on a device with one NVIDIA Tesla A100 GPU with 40GB of memory. The table B.2 presents the utilized hyperparameters.

B.5 Fine–tuning Idefics2

The Idefics2 model was trained on a device with one Nvida A100 GPU with 40GB of memory. The table B.3 shows the utilized hyperparameters.

Hyperparameter	**SigLIP large (patch16/384)**	**SigLIP so400m (patch14/384)**
Train batch size	4	8
Number of train epochs	3	3
Optimizer	Adam [74]	Adam [74]
Learning rate	4.8e-06	4.8e-06
$\beta 1$	0.9	0.9
$\beta 2$	0.98	0.98
ϵ	1e-6	1e-6
Weight decay	0.2	0.2
Seed	42	42
Data type	fp32	fp32

Table B.2: Hyperparameters used for training the SigLIP models

Hyperparameter	**Value**
Vase model	Idefics2 8B
Learning rate	1.4e-5
Train batch size	1
Evaluation batch size	1
Gradient accumulation steps	8
Number of train epochs	1
Gradient checkpointing	True
Data type	auto
Fp16	True
Use peft	True
Lora r	8
Lora alpha	16
Lora target modules	all-linear

Table B.3: Hyperparameters used for training the Idefics2 model

Appendix C

Icon Captioning Task

C.1 Utilized Prompts

Here are the utilized prompts of the selected VLMs for the icon captioning task:

- **All InstructBLIP variants**
  Generate a caption for this given icon from a manual of an electric vehicle.
- **LlaVA–NeXT Mistral 7B**
  [INST] <image>
  Generate a caption for this given icon. Do not write whole sentences. Write a short, concise and precise description of this icon. [/INST]
- **Idefics2 8B**
  User:<image>Generate a caption for this given icon. This icon was taken from a manual of an electric vehicle. Do not write whole sentences. Write a short caption of this icon.<end_of_utterance>
  Assistant:

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need”, *Advances in neural information processing systems*, vol. 30, 2017.

[2] D. Jurafsky and J. H. Martin, *Speech and language processing. vol. 3*, 2014.

[3] P. Lewis, E. Perez, A. Piktus, *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks”, *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[4] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”, in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[6] S. A. Afshine Amidi. “Convolutional neural networks cheatsheet”. (2018), [Online]. Available: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks` (visited on 05/20/2024).

[7] A. Chowdhery, S. Narang, J. Devlin, *et al.*, “Palm: Scaling language modeling with pathways”, *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[8] T. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners”, *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[9] J. L. Elman, “Finding structure in time”, *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, *arXiv preprint arXiv:1810.04805*, 2018.

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding with unsupervised learning”, 2018.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale”, *arXiv preprint arXiv:2010.11929*, 2020.

[14] D. M. Squire, W. Müller, H. Müller, and T. Pun, “Content-based query of image databases: Inspirations from text retrieval”, *Pattern Recognition Letters*, vol. 21, no. 13-14, pp. 1193–1198, 2000.

[15] S. P. Alara Dirik. “A dive into vision-language models”. (2023), [Online]. Available: `https://huggingface.co/blog/vision_language_pretraining#a-dive-into-vision-language-models` (visited on 05/20/2024).

[16] A. Radford, J. W. Kim, C. Hallacy, *et al.*, “Learning transferable visual models from natural language supervision”, in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[17] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, *Sigmoid loss for language image pre-training*, 2023. arXiv: `2303.15343 [cs.CV]`.

[18] J. Wang, Z. Yang, X. Hu, *et al.*, “Git: A generative image-to-text transformer for vision and language”, *arXiv preprint arXiv:2205.14100*, 2022.

[19] J. Li, D. Li, S. Savarese, and S. Hoi, “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models”, *arXiv preprint arXiv:2301.12597*, 2023.

[20] Y. Fang, W. Wang, B. Xie, *et al.*, “Eva: Exploring the limits of masked visual representation learning at scale”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 358–19 369.

[21] S. Zhang, S. Roller, N. Goyal, *et al.*, “Opt: Open pre-trained transformer language models”, *arXiv preprint arXiv:2205.01068*, 2022.

[22] H. W. Chung, L. Hou, S. Longpre, *et al.*, *Scaling instruction-finetuned language models*, 2022. arXiv: `2210.11416 [cs.LG]`.

[23] W. Dai, J. Li, D. Li, *et al.*, *Instructblip: Towards general-purpose vision-language models with instruction tuning*, 2023. arXiv: `2305.06500 [cs.CV]`.

[24] “Vicuna”. (2023), [Online]. Available: `https://github.com/lm-sys/FastChat`.

[25] H. Touvron, T. Lavril, G. Izacard, *et al.*, “Llama: Open and efficient foundation language models”, *arXiv preprint arXiv:2302.13971*, 2023.

[26] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, *Minigpt-4: Enhancing vision-language understanding with advanced large language models*, 2023. arXiv: `2304.10592 [cs.CV]`.

[27] J. Chen, D. Zhu, X. Shen, *et al.*, *Minigpt-v2: Large language model as a unified interface for vision-language multi-task learning*, 2023. arXiv: `2310.09478 [cs.CV]`.

[28] E. J. Hu, Y. Shen, P. Wallis, *et al.*, “Lora: Low-rank adaptation of large language models”, *arXiv preprint arXiv:2106.09685*, 2021.

[29] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning”, *arXiv preprint arXiv:2304.08485*, 2023.

[30] H. Liu, C. Li, Y. Li, and Y. J. Lee, *Improved baselines with visual instruction tuning*, 2023. arXiv: `2310.03744 [cs.CV]`.

[31] H. Liu, C. Li, Y. Li, *et al.*, *Llava-next: Improved reasoning, ocr, and world knowledge*, 2024. [Online]. Available: `https://llava-vl.github.io/blog/2024-01-30-llava-next/`.

[32] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations”, in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[33] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning", *arXiv preprint arXiv:2003.04297*, 2020.

[34] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, "Mistral 7b", *arXiv preprint arXiv:2310.06825*, 2023.

[35] "Nous-hermes-2-yi-34b". (2023), [Online]. Available: `https://huggingface.co/NousResearch/Nous-Hermes-2-Yi-34B`.

[36] H. Laurençon, L. Tronchon, M. Cord, and V. Sanh, "What matters when building vision-language models?", *arXiv preprint arXiv:2405.02246*, 2024.

[37] F. Liu, J. M. Eisenschlos, F. Piccinno, *et al.*, *Deplot: One-shot visual language reasoning by plot-to-table translation*, 2022. arXiv: `2212.10505 [cs.CL]`.

[38] H. W. Chung, L. Hou, S. Longpre, *et al.*, "Scaling instruction-finetuned language models", *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[39] F. Liu, F. Piccinno, S. Krichene, *et al.*, *Matcha: Enhancing visual language pretraining with math reasoning and chart derendering*, 2022. arXiv: `2212.09662 [cs.CL]`.

[40] OpenAI, *Gpt-4 technical report*, 2023. arXiv: `2303.08774 [cs.CL]`.

[41] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions", *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.

[42] H. Agrawal, K. Desai, Y. Wang, *et al.*, "Nocaps: Novel object captioning at scale", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8948–8957.

[43] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6904–6913.

[44] D. Gurari, Q. Li, C. Lin, *et al.*, "Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. "Vqa". (2017), [Online]. Available: `https://visualqa.org` (visited on 05/21/2024).

[46] D. Gurari, Q. Li, A. J. Stangl, *et al.* "Vizwiz - visual question answering". (2018), [Online]. Available: `https://vizwiz.org/tasks-and-datasets/vqa/` (visited on 05/21/2024).

[47] R. Vedantam, C. L. Zitnick, and D. Parikh, *Cider: Consensus-based image description evaluation*, 2015. arXiv: `1411.5726 [cs.CV]`.

[48] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval", *Journal of documentation*, 1972.

[49] Y. Lu, D. Jiang, W. Chen, W. Wang, Y. Choi, and B. Y. Lin, *Wildvision arena: Benchmarking multimodal llms in the wild*, 2024. [Online]. Available: `https://huggingface.co/spaces/WildVision/vision-arena/`.

[50] O. Contributors, *Opencompass: A universal evaluation platform for foundation models*, `https://github.com/open-compass/opencompass`, 2023.

[51] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, *et al.*, "Okapi at trec-3", *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: `1810.04805 [cs.CL]`.

[53] E. M. Voorhees *et al.*, "The trec-8 question answering track report.", in *Trec*, vol. 99, 1999, pp. 77–82.

[54] M. Lewis, Y. Liu, N. Goyal, *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension", *arXiv preprint arXiv:1910.13461*, 2019.

[55] AI@Meta. "Introducing meta llama 3: The most capable openly available llm to date". (2024), [Online]. Available: `https://ai.meta.com/blog/meta-llama-3/` (visited on 05/22/2024).

[56] M. Abdin, S. A. Jacobs, A. A. Awan, *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone", *arXiv preprint arXiv:2404.14219*, 2024.

[57] D. Bergmann. "What is instruction tuning?" (2024), [Online]. Available: `https://www.ibm.com/topics/instruction-tuning` (visited on 05/22/2024).

[58] W. Chen, H. Hu, X. Chen, P. Verga, and W. W. Cohen, "Murag: Multimodal retrieval-augmented generator for open question answering over images and text", *arXiv preprint arXiv:2210.02928*, 2022.

[59] M. Yasunaga, A. Aghajanyan, W. Shi, *et al.*, "Retrieval-augmented multimodal language modeling", *arXiv preprint arXiv:2211.12561*, 2022.

[60] A. B. Annie Surla and T. Varshney. "An easy introduction to multimodal retrieval-augmented generation". (2024), [Online]. Available: `https://developer.nvidia.com/blog/an-easy-introduction-to-multimodal-retrieval-augmented-generation/` (visited on 05/22/2024).

[61] Nvidia. "Multi-modal vision-language model that understands text/images and generates informative responses". (2024), [Online]. Available: `https://build.nvidia.com/nvidia/neva-22b/modelcard` (visited on 05/22/2024).

[62] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks", in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: `http://arxiv.org/abs/1908.10084`.

[63] M. Abdin, S. A. Jacobs, A. A. Awan, *et al.*, *Phi-3 technical report: A highly capable language model locally on your phone*, 2024. arXiv: `2404.14219 [cs.CL]`.

[64] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation", in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[65] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments", in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[66] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries", in *Text summarization branches out*, 2004, pp. 74–81.

[67] O. Kobza, J. Čuhel, T. Gargiani, D. Herel, and P. Marek, "Alquist 5.0: Dialogue trees meet generative models. a novel approach for enhancing socialbot conversations", *arXiv preprint arXiv:2310.16119*, 2023.

[68] Č. Jan, "Multimodální rozpoznávání emocí z řeči", B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2021.

[69] M. Poludin, *Tableqa-llmagent*, `https://github.com/poludmik/TableQA-LLMAgent`, 2024.

[70] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks", in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: `https://arxiv.org/abs/1908.10084`.

[71] A. Singh, V. Natarajan, M. Shah, *et al.*, *Towards vqa models that can read*, 2019. arXiv: `1904.08920 [cs.CL]`.

[72] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, *C-pack: Packaged resources to advance general chinese embedding*, 2023. arXiv: `2309.07597 [cs.CL]`.

[73] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus", *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[74] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: `1412.6980 [cs.LG]`.