BHT Berliner Hochschule für Technik

Image Captioning in Personal Photo Management: A Comparative Study of Different Techniques

vorgelegt von

Niaz Faridani-Rad

EDV.Nr.:925 233

dem Fachbereich VI – Medieninformatik
der Berliner Hochschule für Technik Berlin
vorgelegte Masterarbeit
zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Studiengang

Medieninformatik

Tag der Abgabe February 28, 2024

Gutachter

Prof. Dr. K. Hildebrand	Berliner Hochschule für Technik
Prof. Dr. Roland Petrasch	Berliner Hochschule für Technik

Studiere Zukunft

Abstract

In recent years, image content have become ubiquitous due to the widespread adoption of smartphones. However, traditional images often lack textual context, making it challenging to index and search for specific visual information [Hossain et al. 2018]. Image captioning has emerged as an important tool to facilitate the organization and understanding of this visual content. It also serves as a vital tool for making content accessible to individuals with visual impairments [Sidorov et al. 2020], aids in scene understanding and object recognition in autonomous vehicles and robotics [Cornia et al. 2019] and makes it easier to share their images on social media platforms with tag systems [Park et al. 2017].

The rapid increase in visual data has raised privacy concerns, prompting the adoption of self-hosted photo management systems. Yet, the varying performance across servers and clients can affect user experiences. To understand the practical implications and gather image captioning system requirements, users of the self-hosted system LibrePhotos were surveyed.

A comprehensive analysis of current literature regarding performance and resource targets for image captioning is presented. Performance is compared with older methods implemented in LibrePhotos like im2txt [Vinyals et al. 2017], against improved approaches like BLIP [Li et al. 2022]. To ensure a practical assessment, the analysis is conducted on representative hardware platforms, considering the effects of quantization [Jacob et al. 2017b] and evaluating common performance metrics such as CIDEr [Vedantam et al. 2014] and BLEU [Papineni et al. 2001] scores with the COCO Dataset [Lin et al. 2014].

User modeling is a crucial aspect of image captioning systems, as it allows for personalized and context-aware caption generation. Captions can vary significantly in style and content when comparing different sources, like Tumblr, Instagram or stock image sites, underscoring the need for adaptable and versatile captioning models. While there are datasets like nocaps [Agrawal et al. 2018] for in-the-wild image testing, testing the performance in the context of user modeling is not yet well researched.

In response to these challenges, I propose a novel solution based on the integration of BLIP with a Large Language Model (LLM) component. This addition allows us to fine-tune and adjust captions based on specific prompts, enhancing the system's ability to capture user preferences and context, creating a more versatile and user-centric image captioning system that can adapt to various captioning scenarios.

To evaluate the proposed solution's effectiveness, a user study was conducted. Users rated captions of twenty images, which were created with im2txt, BLIP or our proposed solution. In addition CLIP scores [Hessel et al. 2022] for the captions were calculated. Results demonstrate improved CLIP scores, when adding previous attained knowledge like person or location information, raising the score from 28.79 to 29.6. It was also established, that the CLIP score can be used to detect hallucination, which have an negative impact on captions, while the user study highlighted the significance of user preference and the need to mitigate hallucination effects.

Lastly, I added the enhanced image captioning tool to LibrePhotos, making it accessible to all users. In order to implement this novel feature, it was essential to improve the User Experience including warning of high ram usage, a rich text editor with an edit mode, adding a technical survey and supporting downloading models, with a background job and settings, while adding GPU support, with implementing services for machine learning tasks and unloading after a time period.

Aufgabenblatt – durch Original austauschen !!!

Erklärung

Ich versichere, dass ich diese Abschlussarbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

In dieser Masterarbeit wurde generative künstliche Intelligenz als Hilfmittel genutzt, um Formulierungen zu verbessern.

28.02.2024 Niaz Faridani Rad

Datum Unterschrift

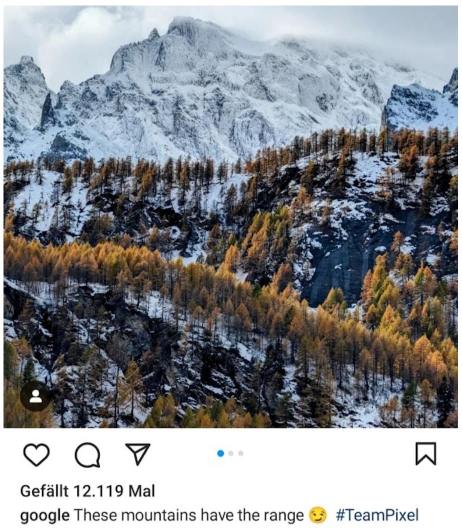
Contents

List of Figures

Chapter 1

Introduction

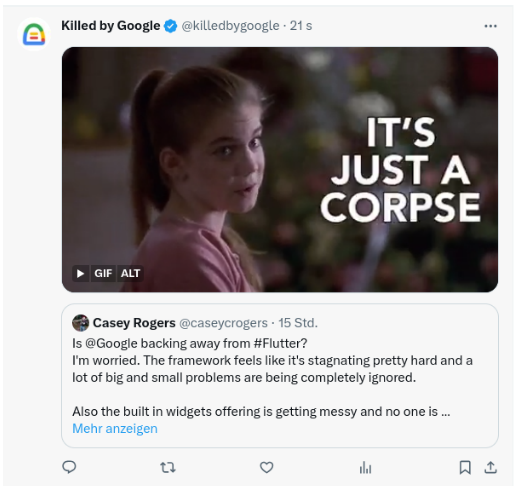
In today's digital world, we are surrounded by more pictures than ever before. With the switch to smartphones as our digital cameras, the amount of images and the potential use cases are ever-increasing. In 2021 for example it was estimated that around 1.4 trillion images were shot and a total of 7.8 trillion images stored [Pantic 2022]. These images are used for all sorts of use cases, from personal memories to important scientific and industrial work, but also how we communicate with each other over social media.



Gefällt 12.119 Mal

google These mountains have the range 😏 #TeamPixel

(a) Instagram [@google 2023]

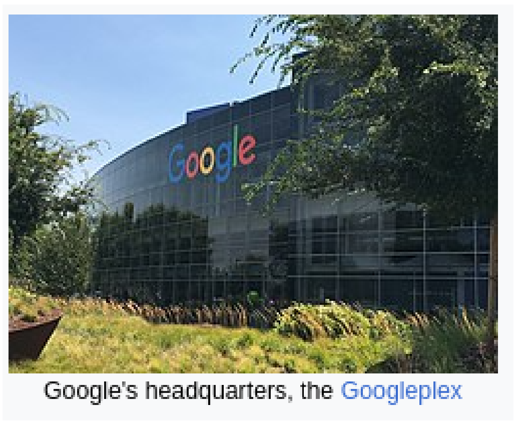


Killed by Google @killedbygoogle · 21 s

Casey Rogers @caseycrogers · 15 Std.
Is @Google backing away from #Flutter?
I'm worried. The framework feels like it's stagnating pretty hard and a lot of big and small problems are being completely ignored.

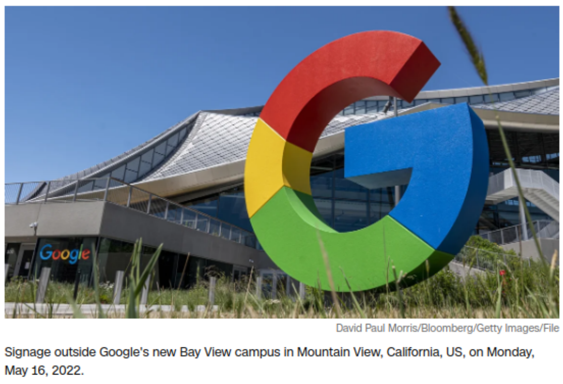
Also the built in widgets offering is getting messy and no one is ...
Mehr anzeigen

(b) Twitter: Its Just Corpse Dead Body GIF [@killedbygoogle 2023]



Google's headquarters, the Googleplex

(c) Wikipedia [Wikipedia 2024]



David Paul Morris/Bloomberg/Getty Images/File

Signage outside Google's new Bay View campus in Mountain View, California, US, on Monday, May 16, 2022.

(d) CNN [Fung 2023]

Figure 1.1: Different captions on platforms

Image captioning, the process of adding a textual descriptions or explanations to images, has become a common workflow across major online platforms and applications. From social media networks to news websites, e-commerce platforms to personal photo management systems, image captioning plays a pivotal role in enhancing user engagement and content accessibility.

While image captioning is a common feature, it manifests in diverse ways across different platforms, reflecting the specific needs and goals of each platform's user base. On platforms such as Wikipedia and news websites, image captions tend to be technical and factual, serving as informative annotations that provide context and clarity to images. Wikipedia tries to keep descriptions short and does not need to attribute the photographer, because images are creative commons on Wikipedia compared to a news site. These captions prioritize accuracy and clarity to aid in understanding. Stock image websites and e-commerce platforms employ image captions as a means to improve searchability. Descriptive captions help users find specific images or products, enhancing the overall user experience.

Platforms like Twitter and Mastodon emphasize accessibility by incorporating image captions to assist users with visual impairments. In 2022 Twitter added a user flow, which encouraged adding image descriptions, when tweeting an image. These captions aim to convey the essential content of the image, ensuring that a broader audience can engage with the content and are labeled with an ALT tag. Additionally to the ALT tag, tweets with images can also have text, which can have additional context about the image, but do not have to. In contrast, platforms like Instagram and Tumblr often feature image captions that are more opinionated, expressive, or artistic in nature. These captions add depth and context to the visual content, allowing users to convey personal narratives, emotions, or artistic interpretations. The presence and usage of hashtags in image captions vary widely across platforms. Some platforms encourage the inclusion of hashtags to enhance content discoverability, while others prioritize a more minimalist captioning style.

The importance of image captioning in the digital age cannot be overstated. Image captions enable more effective communication by providing context, information, or emotional cues that complement visual content. For individuals with disabilities, image captions are instrumental in making digital content accessible, ensuring that no one is excluded from engaging with visual media. They are also a common training set for generative deep learning models.

With so many images, it's getting really hard to keep track of them all and find the ones we need. In the context of this master thesis, I will talk about personal photo collections, which are personal media collected over a lifetime of a user with different devices. In the past, people used to organize their personal photo collections by hand and are usually sorted based on the W4 entries: when, where, what and who.

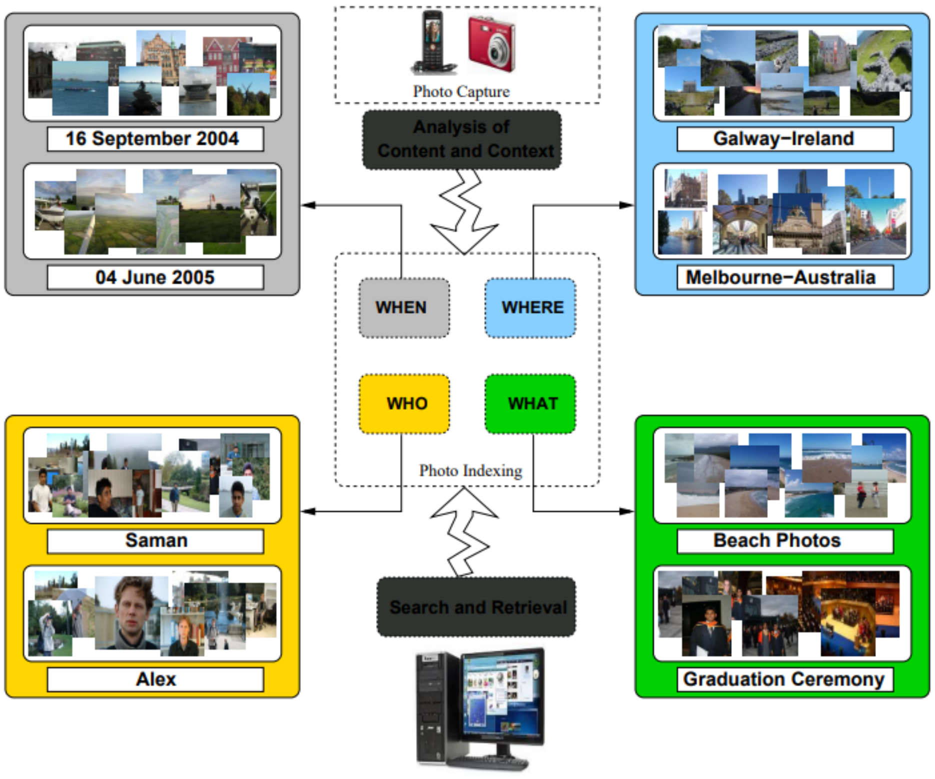


Figure 1.2: W4 entries: when, where, what and who [Cooray 2008]

It would be common workflow to label persons in the photos, sort them by place or by event and put them in folders. However with the increasing size of the average photo collection, handling all images by hand became almost impossible. To handle large personal photo collections, photo management systems sprung up. These programs use metadata to automatically sort the images based on the W4 criteria and aid in adding more metadata to images.

Machine learning added the capability to infer more information in single images and in groups of images by having the ability to understand and recognize patterns without metadata. It can help by automatically sorting images into categories, detect faces, find objects in pictures, and even help with searching for images based on what's in them. The first kind of deployed systems where simple and could run on the hardware of users, which would for example suggest similar categories based on a similarity score between images.

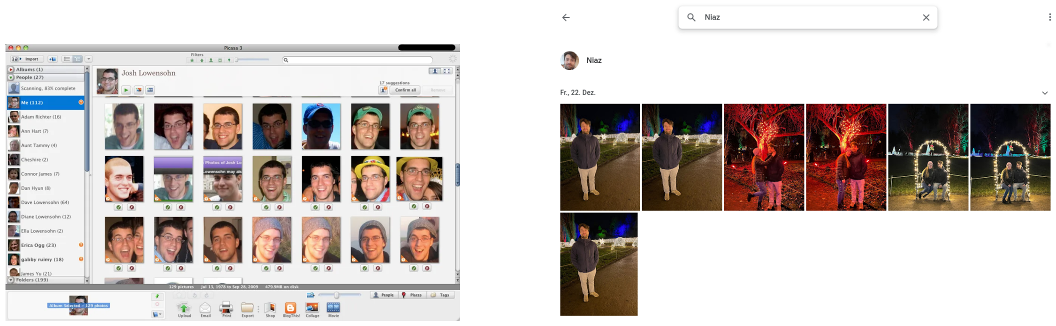

(a) Picasa [Lowensohn 2009] (b) Google Photos

Figure 1.3: Higher accuracy leads to different design

With the move to a cloud architecture deep learning models, that can't run smoothly on consumer devices, could be deployed and used. With more compute power, bigger models became very accurate, which subsequently changed how the programs are used. For example in most photo management system, how you would label faces and manage persons changed significantly. In the past users would label each faces and would only get a suggestion, who it could be. These days all faces are grouped into clusters, which the user can label and merge clusters. The need for less supervision lead to a different user design.

Today the W4 criteria are mostly solved with machine learning and metadata, so that now most research focuses on more complex issues like face clustering, event recognition, importance of images and image captioning.

While this move to the cloud was needed in the early 2010s, Apple went a different route for their solution. By creating more powerful hardware, they can process the images on devices and only store image end to end encrypted in the cloud.

In the context of personal photo management, image captioning takes on a unique significance. It allows individuals to organize and retrieve their personal photo collections efficiently, transforming a mere assortment of images into a rich narrative of memories and experiences.

User modeling is a crucial aspect of image captioning systems, as it allows for personalized and context-aware caption generation. Captions can vary significantly in style and content when comparing different sources, like Tumblr, Instagram or stock image sites, underscoring the need for adaptable and versatile captioning models. While there are datasets like nocaps [Agrawal et al. 2018] for in-the-wild image testing, testing the performance in the context of user modeling is not yet well researched, which limits the understanding of captioning performance in real-world scenarios. Incorporating sentiment can further improve the quality [Mathews et al. 2015] and relevance of captions. LibrePhotos also has valuable information for photos, such as detected persons and objects, which can be leveraged to enhance caption accuracy and detail.

This master thesis contributes the following to the forefront of image captioning systems: It initiates with a thorough examination of existing literature to establish a comprehensive understanding of the current state of the field. Through meticulous gathering of user preferences and system requirements, the development process is informed with precision. The thesis scrutinizes the traditional im2txt method against the innovative blip approach, replicating test scores and optimizing performance to illuminate their accuracy impacts. Crucially, the selection of a self-hosting viable LLM and the crafting of a dynamic prompt tailored to user preferences elevate the adaptability and user-friendliness of the system.

A significant highlight is the user study comparing im2txt, blip, and the blip integrated with Mistral 7B. This investigation not only showcases the potential of LLMs to enhance captions through prior knowledge but also reveals the utility of CLIP scores in identifying hallucinations. The thesis tackles practical implementation challenges head-on, resulting in the development of a robust system adaptable across diverse platforms. Real-life implementation hurdles are effectively addressed, affirming the practical viability of the proposed image captioning system.

This master thesis is constructed in eight chapters: Following the introduction, the second chapter delves into an exploration of existing research intersecting with Image Captioning in Self-Hosted Personal Photo Management Systems. It provides a comprehensive overview covering various aspects such as image captioning applications, datasets, evaluation metrics, techniques, compressing neural networks, user modeling, and large language models. These discussions collectively offer insight into the interconnected domains, setting the stage for our comparative study in personal photo management.

The background why LibrePhotos was chosen and the benefits of using a self-hosted system will be discussed in chapter three. I will discuss the privacy implications of visual data and the shortcoming and challenges when it comes to implement a new image captioning system within LibrePhotos.

In chapter four I will find out who and with what systems users use self-hosted software, specifically personal photo management systems. Self-hosted software, including personal photo management systems, appeals to a diverse user base with varying needs, technical proficiency, and preferences. Furthermore, I will improve the user experience in personal photo management systems by gaining a deep understanding of user needs and expectations. Image captioning should not only be technically accurate, but also align with user preferences and workflows. This objective aims to uncover user pain points, preferences, and usage patterns to inform the development of user-centric image captioning solutions.

In chapter five, I will comparatively analyse different image captioning techniques. It seeks to evaluate the current method "im2txt" based on a CNN and LSTM encoder decoder architecture by contrasting it with modern techniques like BLIP, which uses transformer based architecture. These techniques will be compared based on criteria such as caption quality based on common benchmarks like CIDEr and BLEU. Efficient deployment is pivotal for the seamless integration of image captioning capabilities into selfhosted personal photo management. Additionally in chapter five identifies deployment approaches that balance computational efficiency, scalability, and deployability on different devices. Neural networks often involve a trade-off between precision and run-time efficiency. Balancing the need for accurate captions with the computational demands of real-time or resource-constrained environments is a critical aspect of this research. The thesis will explore strategies and techniques to optimize the performance of image captioning models within the LibrePhotos ecosystem. This includes evaluating different runtimes like ONNX and post training techniques like Quantization to find a sweet spot that maximizes caption quality while minimizing computational overhead.

In chapter six I will advance image captioning by developing a novel image captioning system. Existing image captioning systems often struggle to incorporate prior knowledge, such as identified individuals in the image. Integrating contextual information about the image could significantly enrich the quality of generated captions. Image captioning systems lack the ability to personalize captions according to user preferences. The degree of factual information, tone, or style (e.g., news-like or Instagram-like captions) remains a challenge for current solutions. The process of creating both keywords and captions is typically disjointed across different systems. A cohesive solution that seamlessly integrates these tasks could streamline the user experience and improve efficiency. Current image captioning systems lack multi-language support, restricting their usability for a diverse user base. A novel solution should prioritize inclusivity by providing effective captioning in various languages. Adding a Large Language Models (LLMs), in our case Mistral 7B, present a promising avenue for overcoming the aforementioned challenges. Their capabilities add to current image captioning systems with the complex nature of image captioning tasks: LLMs excel in handling a wide range of tasks, including those that traditional image captioning systems find challenging. Their versatility makes them well-suited for addressing the diverse needs of users. LLMs demonstrate minimal hallucination, when provided with enough information and context. It can provide more accurate and contextually relevant captions by building on information from previous attained knowledge. The ability of LLMs to create tags and facilitate accurate translation further enhances their utility. This comprehensive approach simplifies the process of generating informative captions in multiple languages. Personalizing captions becomes straightforward with LLMs, as users can easily adjust prompts to tailor the style, tone, and factual content according to their preferences. Like the title suggest I will here compare our novel approach with the baseline methods in a user study and by comparing state of the art benchmarks like CLIPScore.

In chapter seven I will explore what it takes to ship an enhanced image captioning experience in the LibrePhotos self-hosted photo management system, based on the findings and insights derived from the research. Practical application of research findings is important. Implementing a tangible, user-oriented improvement in LibrePhotos serves as a real-world demonstration of the research's relevance and potential impact. This will involve integrating the chosen paradigm, adding GPU support, optimizing performance trade offs, implementing a cohesive frontend, managing models and conducting rigorous testing to ensure that the resulting system enhances the user experience and functionality of LibrePhotos.

In chapter eight I will summarize my findings and recap my implementation of a new a novel image captioning system, while presenting the challenges and future work that needs to be done.

Chapter 2

Related Works

This chapter contains a comprehensive exploration into the existing body of research intersecting with Image Captioning in Self Hosted Personal Photo Management Systems. Section 2.1 delves into Image Captioning, covering applications, datasets, evaluation metrics, and techniques. Section 2.2 explores Compressing Neural Networks, discussing weight pruning, quantization, knowledge distillation, ONNX, and finetuning. User Modeling in Section 2.3 investigates motivations behind user actions in personal photo management, encompassing tagging, image capture motivations, and sentiment in captions. Section 2.4 examines Large Language Models, addressing applications, self-hosting on edge devices, and relevant evaluation metrics. These sections collectively provide a concise overview of the interconnected domains, setting the stage for our comparative study in personal photo management.

2.1 Image Captioning

In the landscape of computer vision, image captioning has emerged as a challenging research domain. This field seeks to bridge the gap between visual perception and natural language understanding by giving machines with the capability to generate relevant captions for images. Over the years, significant strides have been made in the development of image captioning models, driven by advancements in deep learning architectures. I explore the diverse approaches, methodologies, and breakthroughs that have shaped the evolution of image captioning, providing an overview of the current state of the art.

2.1.1 Applications

Various disciplines have explored the application of image captioning, recognizing that the capacity to consistently provide descriptive captions for images can address more complex challenges. One of these challenges is that autonomous agents like robots are not able to explain to humans what they perceive. In "SMArT: Training Shallow Memory-aware Transformers for Robotic Explainability" [Cornia et al. 2019] researchers looked into image captioning for domestic robots to find out, if they can caption simulated scenes from the perspective of a robot and lower the computational demands, so that they can run on a self-contained system.
Another application is to find events in a group of photos. [Savchenko 2020] compared grouping based on textual description with more common approaches to classification and showed that it can achieve a lower error rate. More relevant for personal photo management system is "Deepdiary", an automatic life-logging and summarization tool. The authors tried to implement a "Summarize my day" feature with personal images collected from wearable, life-logging cameras based on automatic image captions [Fan et al. 2018]. Another challenge is trying to guess popularity of a given image when used in a social media setting. In [Hidayati et al. 2020] they tried to approximate this based on the caption and showed promising results. While tagging

images in a personal photo management system was implemented in "Tagging Personal Photos with Transfer Deep Learning" [Fu et al. 2015], there is no system to date, which does something similar with image captions.

2.1.2 Datasets

To assess the reliability of new image captioning systems across a diverse range of images, scientists generated datasets to compare and evaluate their models. Given the diverse nature of images and the challenge even for humans in captioning them, multiple datasets have been employed to evaluate different aspects of image captioning.

The COCO (Common Objects in Context) dataset, detailed in [Lin et al. 2014], stands as the de facto standard for object recognition and image captioning and in the realm of computer vision research. The COCO dataset provides a substantial collection of training and validation images, each annotated with multiple labels and accompanied by multiple captions. However, a noteworthy limitation of the COCO dataset lies in its relatively limited set of object classes. This constraint can impact the model's ability to grasp a comprehensive understanding of diverse objects and scenes of real-world visual content.

The Conceptual Captions dataset, introduced in [Sharma et al. 2018], serves as a alternative to the COCO dataset for image captioning. Developed by Google, it introduces improvements in diversity of captions by scraping images along with their associated alternative text. This dataset is specialized in captions as it does not have a limited amount of object classes.

The nocaps dataset, as described in [Agrawal et al. 2018], addresses some of the limitations of the COCO dataset by incorporating additional 15,100 images from open images to COCO. It introduces novel captions and objects, with many of these objects appearing in only a few images. This diversity is captured through three domains: indomain, neardomain, and out of domain.

The textcaps dataset, introduced in [Sidorov et al. 2020], addresses a gap in previous datasets by evaluating the text found in images. While previous datasets focus on visual content, textcaps adds images with captions specifically describing the text found within them. This inclusion expands the scope of evaluation for models, considering both visual and textual elements in image understanding.

2.1.3 Evaluation Metrics for Image Captioning

The different data sets have their strengths and weaknesses, therefore it is necessary to define benchmarks by which I can assess them. There are various evaluation systems that examine the quality of the captions in different ways. The important benchmarks are presented below.

BLEU (Bilingual Evaluation Understudy): Described in [Papineni et al. 2001], BLEU was initially designed in a machine translation context. It operates by counting ngrams, which are contiguous sequences of n items, calculating precision, penalizing excessively short output, and generating scores for 1, 2, 3, and 4 ngrams. The resulting BLEU score ranges between 0 and 1, offering a quantitative measure of caption quality.

METEOR (Metric for Evaluation of Translation with Explicit ORdering): Also originating from machine translation, as detailed in [et al. 2008], METEOR matches words and chunks between validation and generated text. It takes into account synonyms and stems of words, providing a nuanced evaluation of captioning performance.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Developed to evaluate summaries, as introduced in [Lin 2004], ROUGE employs tokens instead of words and utilizes ngrams. It evaluates based on recall and precision, generating a single F1 score, which is a harmonic mean between recall and precision. This metric is adaptable to image captioning assessment.

CIDEr (Consensus-based Image Description Evaluation): Specifically designed for image captions, as discussed in [Vedantam et al. 2014], CIDEr incorporates tokens, weighs words by importance, considers ngrams, and calculates a similarity score between the generated caption and validation captions. It then builds a consensus-based evaluation to reward common themes, offering a comprehensive assessment of caption quality.

SPICE (Semantic Propositional Image Caption Evaluation): Tailored for image captions, outlined in [Anderson et al. 2016], SPICE parses captions into structured semantic representations using specific natural language processing (NLP) techniques. It compares these representations to reference captions, assessing how well they capture the same semantic content. This metric provides a unique perspective on the semantic fidelity of generated captions.

A recent benchmark, CLIPScore [Hessel et al. 2022], introduces a novel approach to evaluating image captioning systems. Leveraging CLIP, a framework conventionally applied for image retrieval tasks, CLIPScore employs a multimodal feature encoding methodology. This involves encoding both the generated caption and the associated image into a shared representation. By subsequently computing a score based on the encoded features, CLIPScore facilitates a comparative analysis between the generated caption and the corresponding image. This allows for comparing without captions in the test data.

2.1.4 Techniques in Image Captioning

Addressing image captioning challenges with machine learning is a subset of the broader task of vision-language understanding. This section aims to explain the distinctions among different techniques applied in image captioning. A comprehensive overview, as delineated in [Hossain et al. 2018], provides a valuable resource by summarizing prevalent image captioning techniques. This survey systematically categorizes these techniques, considering aspects such as the type of learning, architecture, number of captions, language models, feature mapping, and others. Utilizing this structured framework, I will differentiate the approaches and categories within the landscape of image captioning.

Types of learning

In supervised learning, the algorithm is trained on a labeled dataset, where each input data point is associated with a corresponding output or target. The model learns the mapping between inputs and outputs, and during training, it adjusts its parameters to minimize the difference between its predictions and the actual targets. Image classification, where an algorithm learns to categorize images into predefined classes, is an example of supervised learning.

In unsupervised learning, the algorithm is given unlabeled data and must find patterns or structure within the data without explicit guidance. It aims to discover hidden relationships or groupings in the data. Reinforcement learning, Clustering and dimensionality reduction are common tasks in unsupervised learning. This is rather uncommon in image captioning though and supervised learning is the standard.

Architecture

The encoder-decoder architecture is commonly used in sequence-to-sequence tasks, such as machine translation and image captioning. The encoder processes the input data (e.g., an image) and transforms it into a fixed-dimensional representation or context vector, which is needed by the decoder. In the context of image captioning, the encoder may use Convolutional Neural Networks (CNNs) to extract features from the image, creating a compact representation. The decoder takes the fixed-dimensional representation from the encoder and generates the output sequence (e.g., a sentence). Long Short-Term Memory (LSTM) or Language models based on transformer architectures are often used as decoders. The decoder attends to the input representation and produces the output step by step. This architecture allows the model to handle input and output sequences of varying lengths and is particularly effective for tasks where the input and output have a sequential and variable relationship, as in translating sentences or generating captions.

Architectures, which only use an Encoder were successfully used for other vision language tasks like image retrieval with CLIP. [Barraco et al. 2022a] showed that this architecture can in fact solve multiple language vision tasks without being explicitly trained for it.
There is also an interest in finding better language models for vision task, which can run on less resources like "MiniVLM" [Wang et al. 2021].

Number of captions

Dense Captioning uses a feature extractor and create multiple captions for regions of the image and then combines them in one caption describing the whole scene. While this method was introduced in DenseCap, the most cited paper, which uses this technique is [Karpathy und Fei-Fei 2015]. The model presented in the paper generates natural language descriptions of image regions based on weak labels, using images and sentences as datasets with minimal hardcoded assumptions. The approach incorporates a novel ranking model that aligns visual and language modalities through a common, multimodal embedding. The model achieved state-of-the-art performance at that time in image-sentence ranking experiments.

Whole Scene Based Captioning is the usual architecture, which decodes from a multimodal feature the whole scene as a text caption. Based on the work in [Vinyals et al. 2017] im2txt implements such a feature. In the paper they present NIC, which uses a CNN to encode an image and a recurrent neural network to generate corresponding sentences, trained to maximize the likelihood of the sentence given the image, which won the 2015 MS COCO challenge, which was a competition held in conjunction with the COCO dataset focusing on image captioning and object detection.

Uncommon ideas like the template approaches use predefined templates or structures to guide the generation of captions. These templates provide a structured framework for creating captions and help creating valid grammar. There are also hybrid methods like Neural Baby Talk [Lu et al. 2018], which generate a template with slots and then fills the slots based on information from another object detector.

Feature Mapping

Multimodal feature mapping involves combining information from multiple modalities, such as images and text, to create a joint representation.

In image captioning, multimodal feature mapping typically refers to the process of aligning features extracted from images (e.g., using CNNs) with features derived from the language domain (e.g., using RNNs). The goal is to create a shared or common representation space where both visual and textual information can be integrated.

This mapping facilitates the fusion of visual and linguistic information, allowing the model to generate coherent and contextually relevant captions for images. Techniques like attention mechanisms are often employed in multimodal architectures to highlight relevant parts of the image when generating different parts of the caption, enhancing the alignment between visual and textual features.

While CNN where used in the past vision transformers (ViTs) are now used. ViTs represent a type of neural network architecture that deviates from the traditional CNNs commonly used in computer vision tasks. ViTs leverage the Transformer architecture, initially popularized in language models, and apply it directly to visual data.

Language models

All models which generate human-like text are language models. In the context of image captioning they usually used image data in form of a multimodal encoding and caption data. In [Hossain et al. 2018] they report a common usage of Long Short-Term Memory (LSTM) models, which refers to a type of recurrent neural network architecture designed to handle sequential data with long-range dependencies. For example the current image captioning model in LibrePhotos im2txt, which is based on the paper [Vinyals et al. 2017] uses an LSTM. Current state of the art models leverage transformers. BLIP for example uses Bidirectional Encoder Representations from Transformers (BERT) as a tokenizer.

Architecture of base models

Convolutional Neural Networks (CNNs) are a class of neural network architectures designed for processing grid-structured data, such as images. CNNs are particularly powerful in capturing spatial hierarchies of features and have become a cornerstone in computer vision applications.
The Transformer architecture, introduced in the paper "Attention is All You Need" [Vaswani et al. 2023], is a type of neural network architecture that has proven to be highly effective in natural language processing tasks, such as machine translation and language modeling. Transformers are designed to process sequential data, and they rely on a mechanism called self-attention (or scaled dot-product attention). In this context, self-attention allows the model to selectively emphasize different elements in a sequence, assigning varying levels of importance to each based on their contextual relevance. The scaled dot-product attention method involves computing the dot product of vectors representing sequence elements while maintaining a scaled balance to manage numerical stability. For instance, in the sentence "The cat sat on the mat," self-attention enables the transformer to focus more on critical words like "cat," ensuring a nuanced understanding of the overall context. They also have the interesting ability to develop new emerging features [Caron et al. 2021], which are concepts the model understands without being explicitly trained for it. Examples of emergent features in LLMs might include the ability to understand syntactic structures, grasp semantic relationships between words, and generate coherent and contextually relevant text.
These categories are not as clear cut and there are usually edge cases, which use a combination of ideas. Before the switch transformers became common, top down attention with image regions in feature extractors was the state of the art in image captioning and visual question answering tasks [Ramanishka et al. 2017] [Anderson et al. 2018], which are similar ideas. Adding memory to a transformer architecture [Cornia et al. 2020] was tried, but have remained niche.

BLIP

The BLIP (Bootstrapping Language Image Pre-training) framework, introduced in [Li et al. 2022], presents a novel approach to vision language tasks. The authors highlight that generating captions directly from encoder-based like CLIP models is not straightforward. On the other hand encoder-decoder models, which are good at captioning, are failing at image-retrieval tasks. As LibrePhotos uses CLIP already for image retrieval task, a new framework, which supports also image captioning would be preferable. In the paper the goal was to solve multiple language vision tasks at once and doing so by introducing a framework. They improved it by innovating on datasets by adding synthetic captions with a image captioning system and a filter mechanism trained on a high quality dataset to remove noisy caption from web-scraped image text pairs like in conceptual captions, the LAION dataset or in the synthetic captions.
The second innovation was an introduction of new models trained on this data, which is a encoder-decoder model with multi modal mixture with vision transformer for encoding and multiple modes for decoding e.g. uni model encoding for search, image grounded text encoder for matching based on text and image and lastly image captioning with an image grounded text decoder.
In this master thesis I am only interested in image captioning, but the decision to choose BLIP is also based on the versatility for future features.

2.2 Compressing Neural Networks

The ever-increasing size of neural networks has led to significant challenges in terms of computational resources, memory requirements, and deployment on resource-constrained devices. As models become more complex to achieve higher accuracy, a trade off between speed and accuracy becomes apparent. This section explores various strategies employed in compressing neural networks to mitigate these challenges.

The paper by He et al. [He et al. 2015] highlighted the trend of increasing neural network sizes to improve performance. Larger networks, while achieving impressive accuracy, pose challenges in terms of training time, memory consumption, and deployment feasibility, especially on devices with limited resources.

The trade off between speed and accuracy in neural networks was explicitly addressed by Huang et al. [Huang et al. 2017]. As model complexity grows, achieving high accuracy often comes at the expense of slower inference times. This tradeoff has fueled research efforts to find optimal solutions that balance these competing factors.

MobileNet [Howard et al. 2017] represents a successful implementation for object detection on mobile devices. This model challenges the notion that the trade off between speed and accuracy is clear-cut. By employing depth wise separable convolutions and efficient model design, MobileNet achieves a good balance, demonstrating that careful optimizations can lead to improved efficiency without sacrificing accuracy.

With the paradigm shift to transformers in the paper "Attention is all you need" [Vaswani et al. 2023], learning costs have decreased. However, this has come at the expense of increased model size. Training has transitioned from smaller datasets to larger ones, posing additional challenges. This increased the needed resource for inference by a huge amount.

In response to the challenges posed by large neural networks, researchers are actively exploring more efficient approaches. Rampal et al. [Rampal und Mohanty 2020] delve into this research area, investigating techniques to enhance the efficiency of a classic CNN and LSTM based encoder decoder image captioning system by applying pruning and quantization.

While some novel approaches focus on changing the inputs to neural networks, such as using compressed images instead of traditional RGB inputs [Codevilla et al. 2021] to achieve higher accuracy with smaller models, this section will primarily concentrate on the more common techniques applied directly to the networks themselves.

2.2.1 Weight pruning

The first technique "Weight pruning" is a compression technique used in neural networks to reduce the size and computational requirements of the model by selectively removing certain weights (connections) while retaining the most important ones. The intuition behind weight pruning is that not all connections in a neural network contribute equally to its performance, and some can be pruned without significantly affecting the accuracy.
In "Learning both Weights and Connections for Efficient Neural Networks" [Han et al. 2015] the authors introduced a method to reduce the number of weights by a factor of 10 while achieving similar accuracy. In the follow up paper [Han et al. 2016] the author combined pruning with other techniques like quantization and Huffman encoding to achieve even higher compression ratios up to 49x smaller then the original model.
This works only if the neural network is over-parameterized for the given task, which is usually not the case for large language models.

2.2.2 Quantization

Quantization is a technique used to reduce the precision of the numerical values in a neural network. In standard deep learning models, parameters are typically represented as 32-bit floating-point numbers. Quantization reduces the bit-width of these numbers, leading to more compact model representations.
Quantization can be employed either as a post-processing step after model training or during the training process itself, aiming to minimize accuracy losses. The primary objectives of quantization typically revolve around enhancing inference speed on self-hosted and edge devices or fitting large models into the VRAM of a single GPU.
The choice of bit size in quantization is influenced by the target platform. While 32-bit and 16-bit quantized models are typically optimized for GPUs, smaller bit values, such as 8-bit or 4-bit, are often intended for deployment on CPUs. Some advanced approaches, like those discussed in [Jacob et al. 2017a] and [Banner et al. 2019], explore even smaller bit rates (e.g., 2-bit models) [Chee et al. 2023] while preserving accuracy.
Quantization methods are often tailored to the specific tasks a model is designed to perform. For example in the paper "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale" [Dettmers et al. 2022] the author discusses certain cutoffs that happen, when values get quantized in large language model. This is happening because outlier features that appear in larger models, which have an out sized impact on model performance and get degraded the most by quantization. They implement a mechanism, which loads the full values for these outlier features, while quantizing the other values to achieve full accuracy, while increasing inference speed.

2.2.3 Knowledge Distillation

The primary goal of knowledge distillation is to transfer the knowledge learned by a complex or "teacher" model to a simpler or "student" model. This process helps create a more compact model that retains the performance of the larger model, making it suitable for deployment in resource-constrained environments.

[Barraco et al. 2022b] created a novel model, which use two language models for training, which follow a knowledge distillation paradigm, to create a smaller model, which is still accurate and thus faster in inference.
In "Compressing Visual-linguistic Model via Knowledge Distillation" [Fang et al. 2021] the authors successfully applied knowledge distillation to create small vision language model DistillVLM with 34.5M neurons, which can compete with state of the art model OSCAR, which used 111.7M paremeters.

2.2.4 ONNX

ONNX, which stands for Open Neural Network Exchange, is an open-source format designed to enable interoperability between different deep learning frameworks. ONNX does not directly improve the performance of neural networks but provides several advantages that can contribute to improved efficiency and flexibility in the development and deployment of neural network models.
However ONNX is a popular format and thus acts as a platform for other people to contribute to. This lead to "Compiling ONNX Neural Network Models Using MLIR" [Jin et al. 2020], where the authors compiled ONNX neural networks using LLVM to increase performance.
This approach seems to be one new way how performance will improve in the future as the popular library PyTorch also announced an compiled mode for models to improve performance [PyTorch 2023]. PyTorch is however more a library to develop new models and therefore ONNX seems to be the better choice as a possible inference platform.

2.2.5 Finetuning

Fine-tuning is a technique commonly used in machine learning to adapt a pre-trained model on a specific task or domain of interest. The idea behind fine-tuning is to leverage the knowledge captured by a model during its pre-training phase on a large dataset and then specialize or adjust the model's parameters for a more specific task with a smaller dataset.

With a increase in model size, finetuning times increased too. This is why LoRA (Low-Rank Adaptation of Large Language Models) was proposed for large language models.

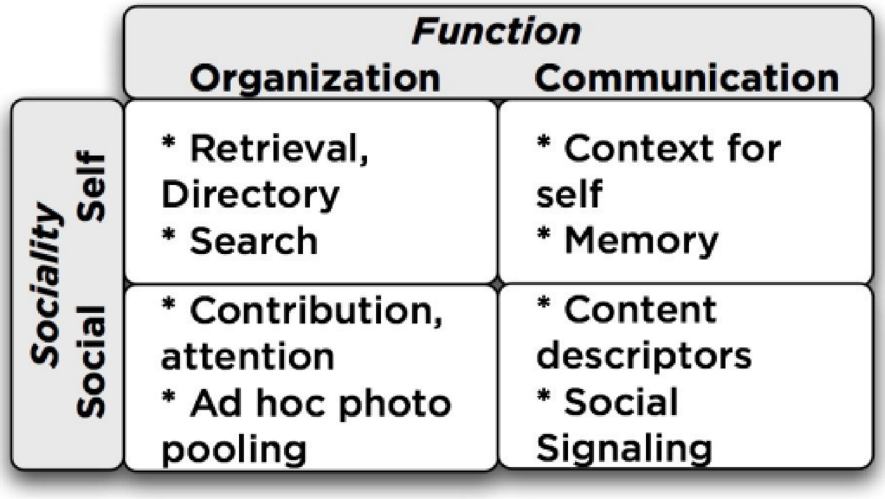
2.3 User Modeling

It is important to understand how users want to use image captioning and which workflow they prefer, when using it in the different contexts and why they do it. On the other hand every work on implementing a more personalized experience with automatic image captioning is relevant too.

2.3.1 Why users tag

In "Why we tag: motivations for annotation in mobile and online media" the authors asked Flickr users, which is a popular photo sharing platform, what there motivations were when annotating images. They used a tool called ZoneTag, which would appear when uploading images, and would suggest popular tags.

Through interviews the authors identified the following two axis, why people tag images: One axis is the Function and the other one is Sociality. On the Sociality axis you can either use annotations for yourself or in a social context and on the Function axis you use it either for organization or communication. While retrieval of images through tags is a common use case, they are also used for context and memory. Social use cases are being accessible to other users or describing



		Function	
		Organization	Communication
Sociality	Self	* Retrieval, Directory * Search	* Context for self * Memory
	Social	* Contribution, attention * Ad hoc photo pooling	* Content descriptors * Social Signaling

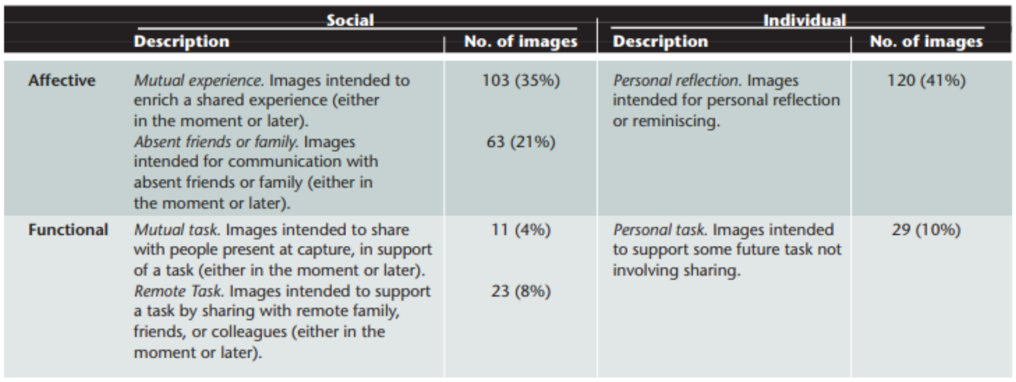
Figure 2.1: A taxonomy of tagging motivations in ZoneTag/Flickr [Ames und Naaman 2007]

the content to other users.

They present best practices when developing annotation tools, which is important for my master thesis. Annotation tools should consider all dimensions of tagging and should make it easy to add annotations after uploading images. Additionally it should be optional as some users do not annotate at all. Different workflows for desktop and mobile should be considered. Adding suggestions while tagging is important, when the goal is to encourage users to tag more.

2.3.2 Why users take images

The authors of "Why users tag" cite an paper, which has implications for my work. In [Kindberg et al. 2005] they study why users take images and also find different dimensions.



	Social		Individual	
	Description	No. of images	Description	No. of images
Affective	*Mutual experience.* Images intended to enrich a shared experience (either in the moment or later). *Absent friends or family.* Images intended for communication with absent friends or family (either in the moment or later).	103 (35%) 63 (21%)	*Personal reflection.* Images intended for personal reflection or reminiscing.	120 (41%)
Functional	*Mutual task.* Images intended to share with people present at capture, in support of a task (either in the moment or later). *Remote Task.* Images intended to support a task by sharing with remote family, friends, or colleagues (either in the moment or later).	11 (4%) 23 (8%)	*Personal task.* Images intended to support some future task not involving sharing.	29 (10%)

Figure 2.2: A taxonomy of reasons for image capture, with numbers and proportions of images by category [Kindberg et al. 2005]

The authors of "Why Users Tag" were inspired by the framework proposed in this paper. Similar to the previous study, our framework comprises two axes, ranging from the social to the individual and from affective to functional dimensions. Affective social images serve the purpose of sharing mutual experiences or facilitating communication with distant friends or family members. The authors also highlight that tasks often drive image capture, whether in social contexts, where tasks are shared, or on an individual level, where tasks are personal. Additionally, there exist images that are both affective and individual, created for personal reflection.

Understanding the motives behind user-generated imagery holds implications for image captioning, as users likely desire different captions based on the context of the image. This aspect is particularly relevant to my master's thesis.

2.3.3 Sentiment in Image Captions

In "SentiCap: Generating Image Descriptions with Sentiments" [Mathews et al. 2015] the authors proposed a system, which could generate caption with a positive or negative sentiment and showed that users found captions with a sentiment at least as factual as the factual ones.

Researchers from Facebook tried to create an automatic captioning system, which optimizes engagement instead of the regular benchmarks [Shuster et al. 2019]. By adding personality traits to captions like sweet, dramatic or anxious they were able to create more engaging captions, that fit the images while also being more engaging.

Personalized image captions based on Instagram data was explored in [Park et al. 2017]. The authors noted that there is a lack of personalized captioning systems and used Instagram images. Notable is that this system contained hashtags, which are usually missing from other more factual image captioning systems.

2.4 Large Language Models

A Large Language Model (LLM) refers to a type of natural language processing (NLP) model that is characterized by its extensive size, typically involving a vast number of parameters. These models are designed to understand and generate human-like text based on the patterns and information present in the vast amount of training data they have been exposed to. Large language models have become increasingly prominent in the field of artificial intelligence and have demonstrated remarkable capabilities in various NLP tasks.

2.4.1 Applications

LLMs are explored as alternatives to traditional search engines, attempting to provide more accurate and contextually relevant results. However, the success of substituting search engines with LLMs has been mixed, as seen in experiments with Bing and other platforms. In the realm of software development, LLMs like GitHub Copilot aim to enhance programmer productivity by suggesting code snippets and completing code segments. This assists developers in writing code more efficiently and accelerates the coding process. LLMs power advanced conversational agents and chatbots, exemplified by models like ChatGPT. While chatbots excel in various tasks, they may face challenges in scenarios requiring nuanced understanding, as demonstrated by instances where they misunderstood queries, such as a truck being sold for 1 dollar [Bakke 2023] and recommended buying a truck from another company [Romero 2023].

2.4.2 Selfhosting / Edge Devices

Regular Language models like Bert [Devlin et al. 2019] or GPT2 are usually trained to achieve only a single task like decoding an image embedding to a caption in the systems described above. While these models are capable of generating sentences LLMs have more emergent features and can understand concepts semantically, which makes them interesting for post processing of text tasks like summarising or adding information to a text.
While trying to improve the efficiency of LLMs is an active field of study, selfhosting and running

LLMs on edge devices is currently more of a niche field, which relies heavily on hobbyist sharing their results on places like reddit in the subreddit r/localLLaMA.
In the last year the large language model community saw the arrival of multiple models, which can run on a single computer and achieve good results. It all started when LLAMA was leaked after only wanting it to release to certain researchers citing security concerns. This leak sparked the interest of a lot of hobbyist, which subsequently tried to run this model locally.
The latest two models, which are embraced by the self-hosting crowd are LLaMa2 and Mistral [Jiang et al. 2023].

2.4.3 Evaluation Metrics

LLMs are subject to evaluation using a variety of benchmarks that span diverse use cases. These benchmarks aim to assess the model's capabilities across multiple tasks, including Commonsense Reasoning, World Knowledge, Reading Comprehension, Code Understanding, Math Problem Solving, Summarizing, and more.
Evaluation tasks within benchmarks can vary in complexity. Some benchmarks may involve 0-shot learning, where models are evaluated on tasks they have not seen during training. Others may allow multiple attempts to refine and improve responses.
Evaluation of LLMs often includes human judgment as a crucial metric. Human annotators assess the quality of model outputs, providing a qualitative understanding of the model's performance. Anonymized results from multiple models are compared to derive insights.
Challenges arise in LLM evaluation due to the subjective nature of grading and the potential for noise in results. Since LLMs generate diverse and contextually rich outputs, the evaluation process can be influenced by prompt engineering and the inherent subjectivity of human assessors.
Reproducibility of benchmarks is a concern, as the subjective nature of grading may lead to variations in results. The same model may produce different outcomes for similar prompts, and different models may perform inconsistently on certain tasks.
The challenges and nuances of LLM evaluation have garnered attention on social media platforms, becoming a topic of discussion and even humor within the research community. Tweets by researchers highlight the intricacies and occasional idiosyncrasies associated with benchmark evaluation, underscoring the need for careful consideration in interpreting results. [Bacaj 2023] [Karpathy 2023] [Wei 2023]

Chapter 3

Background

Before I can start with the survey of the users of LibrePhotos, it is important to give more information about the background of the research and self hosted photo management systems.

3.0.1 Self-Hosted Photo Management Systems: LibrePhotos

The landscape of photo management underwent a significant shift with the removal of the free tier of Google Photos [Amadeo 2021], prompting a surge in demand for alternative, open-source solutions. One such solution that gained prominence in this context is LibrePhotos [Faridani-Rad 2020], a self-hosted photo management system.

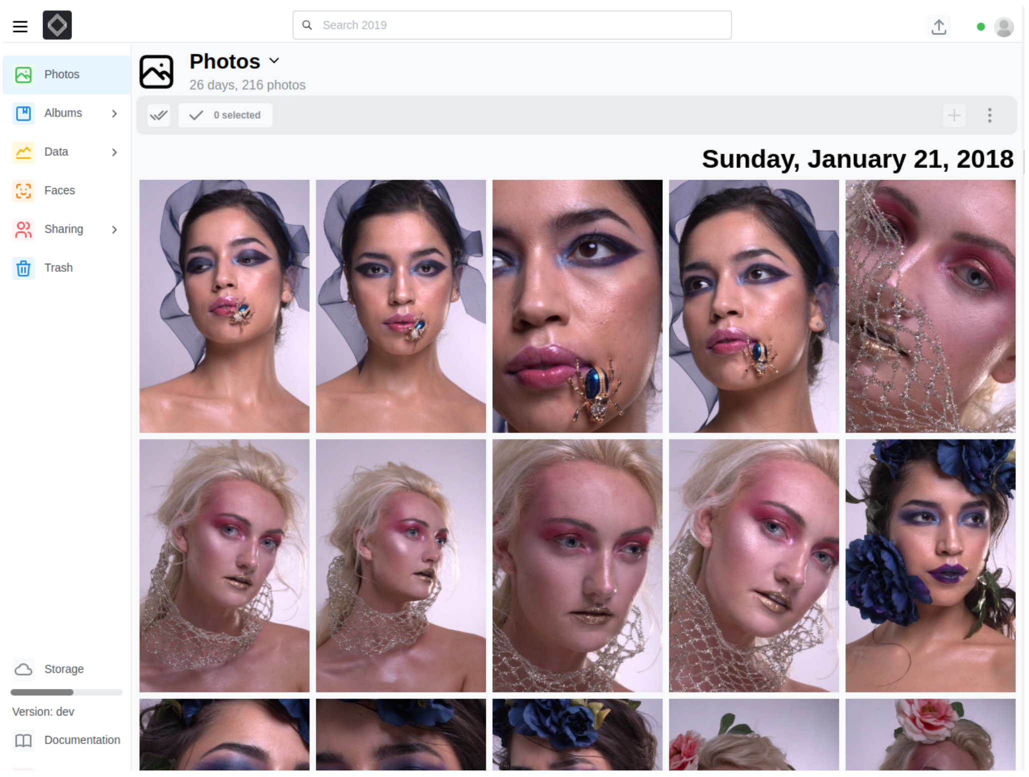


Figure 3.1: Screenshot of LibrePhotos 28.12.23

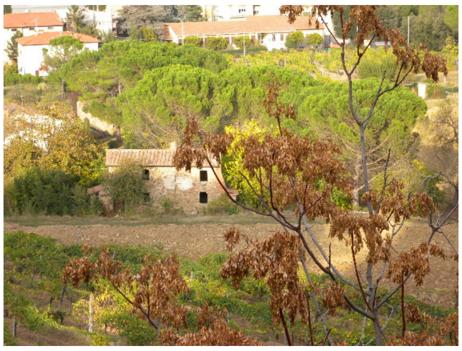
LibrePhotos originated as a fork of "ownphotos" [Nam 2017] a project first developed in 2017. The project was forked by me, because it was abandoned. Since taking over as the project maintainer in 2020, there has been a sustained commitment to its growth and improvement. New features include wide compatibility with images and video formats, geocoding with different providers and semantic image search.

LibrePhotos' relevance and impact in the realm of self-hosted photo management systems were highlighted when it received coverage in Ars Technica, a reputable source of technology news [Kretzschmar 2021] and HackerNews [Hackernews 2020].

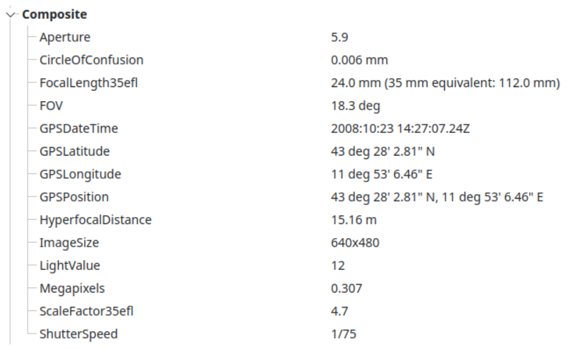
LibrePhotos has successfully amassed a diverse and dedicated user base. This user community reflects the platform's appeal to individuals seeking privacy-conscious and user-centric photo management solutions.

3.0.2 Privacy Implications of Visual Data and self-hosting as the solution

A common reason for self-hosting photo management software are the privacy implications of visual data. Images are valuable targets, which are often viewed as innocuous. They are a common vectors for security issues. On the one hand, the libraries which handle images are often a primary source of security exploits like a recent exploit found in libwebp [Stöckel 2023] or vulnerabilities around features that handle images like cropping images on Android devices [Cunningham 2023], which can be leveraged by malicious actors to compromise user privacy and security.



(a) Image



Composite

Aperture	5.9
CircleOfConfusion	0.006 mm
FocalLength35efl	24.0 mm (35 mm equivalent: 112.0 mm)
FOV	18.3 deg
GPSDateTime	2008:10:23 14:27:07.24Z
GPSLatitude	43 deg 28' 2.81" N
GPSLongitude	11 deg 53' 6.46" E
GPSPosition	43 deg 28' 2.81" N, 11 deg 53' 6.46" E
HyperfocalDistance	15.16 m
ImageSize	640x480
LightValue	12
Megapixels	0.307
ScaleFactor35efl	4.7
ShutterSpeed	1/75

(b) Metadata shown in digiKam

Figure 3.2: Image and metadata in the image

Images contain data, which is important to the user. On the one side you have the visual content, but on the other side, each image carries Exchangeable Image File Format (EXIF) data, which encodes information about the image and its creation. While this is very important data for handling it in a personal photo collection, a significant portion of users remains unaware that images inherently save metadata. This lack of awareness has inadvertently led to users unknowingly exposing sensitive information when posting images online.

In response to these concerns about metadata exposure, many platforms have adopted the practice of stripping EXIF data from images before sharing them. This enhances privacy, but it also results in personal photo collections with incomplete metadata. Most users, which still do not know that metadata is saved, will only hit these issues, when migrating to a new provider. At that point, it's usually too late, and they are locked in by not having access to the metadata anymore.

An example is Google Photos, which provides a take-out service and technically gives you back your images and metadata. However, the images don't have EXIF Data anymore, and the metadata is saved in a separate .json file in a proprietary format. While there are open source scripts, which add the metadata back, this will not be an option for regular users, who can't use python.

Machine learning algorithms on the other hand have demonstrated the ability to infer substantial amounts of data even in the absence of explicit metadata. For instance, image recognition models can identify objects of interest, brands, and other contextual information present within images. While a photo management tool can use it, to improve the metadata and for better displaying of photos, this can also be used to compromise security and privacy.

Face recognition tools and search engines further exemplify how visual data can be used to create social graphs and gather personal information. PimEyes for example is a search engine for faces on the web and it created a lot of security concerns, which resulted in an investigation to regulate it by the EU. [Meineck 2023]

The value of visual data to advertisers cannot be overstated. Advertisers are acutely interested in obtaining access to data derived from visual content, enabling them to tailor advertisements with precision and reach specific target audiences. More users than ever block third party cookies, which makes this technology an important piece in the future of advertising. These solutions are now commonly used in the adtech space to advertise based on content, which is called "Contextual Advertising" with multi billion dollar evaluations like gumgum [GumGum 2023].

Without the ability to audit the code, it means anything can be inferred and used for third party reasons. While there are some rights in the EU, to ask what the data is used for, it's a common issue to not get back the actual use cases for the data.

Another issue of centralization and not self-hosting software is being a target of political movements, which may not align with the users interests. In the pursuit of enhancing user security and combating the dissemination of child sexual abuse material (CSAM), technology giants like Apple introduced client-side scanning mechanisms called NeuralHash. This controversial move came, because they got pressured by activist. These mechanisms involve the automated scanning of images on a user's device to identify and report potentially harmful content. While ostensibly well-intentioned, the implementation of client-side scanning has sparked a significant privacy outcry and raised complex ethical and technical questions. It also turned out that it wasn't well intentioned at all and that these activists got paid by a company, which creates it's own hashing solution. [Fotiadis und Zandonini 2023]

The potential for false positives, where innocuous images are flagged as CSAM, has led to concerns about the impact on user privacy. False accusations and unwarranted scrutiny can have serious consequences. One example, which got coverage from multiple news outlets, was Google Drive flagging a text file containing 1 as having copyright infringing material, which highlights that false positives are always a valid concern [Dolson 2022].

The deployment of client-side scanning technology for CSAM detection has sparked fears of "mission creep", where the same technology could be repurposed for other forms of content monitoring and censorship, eroding user privacy and free expression.

The implementation of client-side scanning poses a challenge to end-to-end encryption, as it requires access to the contents of encrypted communications. This has prompted concerns about the erosion of encryption's fundamental role in protecting user privacy.

In response to the privacy outcry, Apple scrapped the NeuralHash implementation, but it's a hot topic and some governments still want something like this on peoples devices.

None of these issues can happen in open source self hosted software like LibrePhotos, as any such move will lead to forked software, which will then revert controversial changes. Open source self-hosting software cultivates a privacy-conscious digital ecosystem and acts as a counterbalance to proprietary solutions. Even if only a small sample of users and organizations adopt self-hosted solutions, this will keep them competitive and usable enough, which in turn means, that cloud services cannot capture the market completely. This dynamic encourages cloud providers to reevaluate their data practices in favor of privacy.

3.1 Shortcoming and Challenges

Automated image captioning is not a common feature found in photo management systems today and this thesis will explore this novel approach. This leads to shortcomings of the naive implementation of the current system and an unclear design space on how to improve it. Adding to that it has to run on a variety of systems with users with different levels of experience of handling such a software.

3.1.1 LibrePhotos Image Captioning

LibrePhotos features an integrated image captioning module leveraging a PyTorch-based version of the "im2txt" model. This implementation enables the automatic generation of descriptive captions for uploaded images, drawing inspiration from a PyTorch Tutorial [Choi 2017].
However, the current image captioning functionality in LibrePhotos exhibits several limitations. The image captioning component appears to be at a minimum viable product stage, lacking some essential features for a seamless user experience:

- Lack of caption suggestions: The auto-generated caption gets placed in the text field and is not presented as a suggestions, making the process less intuitive and potentially hindering engagement.
- Unclear user flow: When the text field is empty, the auto generated caption gets placed within the text area. This makes it impossible to just have an empty description e.g. the user just wants to get rid of the auto-generated caption.
- Handling of new captions and tags: The system fails to adequately handle new or modified tags within a caption and does not create new albums based on them.
- Non-editable tags: Tags sourced from places365 are not editable, restricting user customization and personalizing of image descriptions.
- Persistent visibility of buttons: The save/generate button is consistently visible, irrespective of its relevance to the user's current action, potentially causing confusion.
- Absence of GPU acceleration: The image captioning module lacks GPU acceleration, which could significantly enhance processing speed and overall performance.
- Embedded machine learning model: The machine learning model is embedded into the Docker image, potentially causing challenges in updating or swapping models without re-configuring the entire system. It also bloats the image size.

- Resource management issues: The image captioning process, residing in the API worker, struggles with unloading properly, leading to elevated RAM usage, especially in scenarios with multiple API workers.
- Performance concerns: The image captioning process is reported to be somewhat slow, impacting the responsiveness of the system. The reason is that it gets handled by an API worker. When there are only a couple of them, one gets blocked for some time until generating is finished.
- Limited model handling: im2txt exhibits limitations in handling diverse situations e.g rotated images and images with content outside to COCO objects, raising concerns about its robustness across a variety of image content.
- Missing integration of EXIF description: The current image captioning module lacks seamless integration with the reading and writing of EXIF (Exchangeable image file format) descriptions.

Part of this research is to address these shortcomings. It is crucial to enhance the overall functionality and user experience of the LibrePhotos image captioning feature in future updates.

3.1.2 Complex Design Landscape

The implementation within a real system is confronted with the challenge of navigating an intricate design space, characterized by the variety of image captions across platforms. The choice of the most suitable approach relies on the specific goals and functionalities desired within a personal photo management system. Several relevant paths emerge, each with its own considerations.

Searchability

Addressing the task of efficiently locating desired images is a primary objective for personal photo management systems. If prioritizing searchability, the emphasis shifts towards incorporating keywords, as their utility surpasses that of descriptive captions. The system needs to facilitate efficient retrieval based on user-provided terms.

Discoverability

Conversely, when faced with an extensive image collection, users may seek an interactive experience that provides suggestions for relevant images based on the ones currently being viewed. In this context, hashtags become crucial, and captions play a role as algorithmically comparable metadata.

Sharing

For personal photo systems operating in an always-online environment, sharing images with others is a common scenario. Captions, in this case, take on a different form, incorporating elements that encourage interaction. Calls to action within captions become pivotal, fostering better reactions from friends and family, diverging from the straightforward keyword-driven approach.

Archiving

In instances where personal photo collections are vast, users may opt for classic archival methods to organize and store their images. Factual captions, detailing the content of the images, gain significance in this scenario. Unlike searchability-driven systems that rely on keywords, archiving relies more on accurate metadata and captions that precisely describe the content for efficient organization and retrieval.

3.1.3 Impact of Diverse Performance Characteristics on User Experience

The landscape of self-hosting and user-run systems encompasses a wide spectrum of performance characteristics, which, in turn, significantly impact the user experience. Understanding this diversity is paramount to addressing the challenges and opportunities that arise when catering to users with varying hardware capabilities and levels of technical expertise.

Diverse hardware

Self-hosting users operate on a vast array of hardware, each with its own set of performance attributes and limitations. This range includes:

Low-End Machines: Users may rely on low-end devices such as Raspberry Pis or aging laptops. These machines often lack certain instruction sets and may have limited processing power and memory.

Middle-End Machines: Dedicated servers equipped with GPUs represent the middle tier of user hardware. These users typically seek improved performance for specialized tasks and are more likely to have access to hardware accelerators.

High-End Machines: Private cloud users occupy the high-end spectrum, with an emphasis on scalability and high-performance computing. Their hardware may vary widely in terms of architectures, network storage systems, and deployment systems.

Addressing the needs and preferences of this diverse user base presents a challenge. Striking a balance between optimizing performance for high-end users while ensuring accessibility and usability for those with limited resources or technical expertise is crucial. Moreover, accommodating different architectures, storage systems, and deployment preferences adds further complexity to the self-hosting ecosystem.

Varied application platforms

LibrePhotos provides docker images, which are hosted on docker hub and provides basic docker compose config files to host the different containers. As containerized applications are easy to integrate in workflows and are kind of new, a plethora of application platforms, which make hosting and maintaining these containers sprung up. With different platforms come different kind of error messages. Unraid for example only allows for single container setups, which can be configured with a UI. The users which prefer these kinds of solutions usually provide only top level errors without looking into the logs.

Users with high scalibility needs tend to use platforms, which are built on top off Kubernetes. These errors are usually more detailed in nature as the users know what they are doing, but are hard to reproduce, if you do not have a cluster lying around to test it.

Different experience level of user base

The knowledge and experience of users running self-hosted systems vary immensely. Users range from seasoned system administrators with extensive Linux experience, who may have specific requirements like S3 storage integration, to novices who are just beginning their journey with Linux and self-hosting. Some users, particularly application developers, may seek to customize and extend the functionality of their self-hosted systems by programming their own applications or integrations.

In light of these challenges and considerations, this master thesis will explore the relationship between diverse performance characteristics and the user experience within the context of self-hosting environments. By identifying patterns and best practices, this research aims to contribute to the enhancement of user satisfaction and usability in the ever-evolving landscape of self-hosted systems.

Chapter 4

Requirements Analysis

Automatically generating accurate and meaningful captions for images is a difficult task. The inherent complexity lies in interpreting the diverse visual content and translating it into coherent and contextually relevant textual descriptions. In this context, it becomes crucial to unravel the specific challenges users face in adopting and interacting with current image captioning systems.

Understanding how users employ captions in their images is a non-trivial aspect. The nuances of user behavior and the varied purposes behind image captioning remain unclear. It is essential to explore whether users are satisfied with existing solutions or if there exists a gap between their expectations and the functionalities provided by current systems.

4.1 Survey of LibrePhotos Users

In order to gain these valuable insights of LibrePhotos users, a comprehensive survey was conducted. To collect and create the survey, Google Forms was used. This survey aimed to collect data that would help improve the user experience and enhance the overall functionality of the platform.

4.2 Methodology

The survey's methodology was designed to ensure the reliability and relevance of the data collected. Key components of the methodology include:

4.2.1 Pretest

Before launching the survey to the entire LibrePhotos user base, it was pretested by initially sharing it with a subset of users. These users were selected based on their active participation and engagement on the LibrePhotos Discord community. This preliminary testing allowed for the identification and solving of any potential issues with the surveys questions, ensuring that it was clear, comprehensive, and user-friendly.

4.2.2 Target Audience

The survey was primarily distributed via the release notes of the monthly releases and through the discord of LibrePhotos. Both channels mostly target "power users" of LibrePhotos. Power users were considered suitable participants due to their extensive experience with the platform, making them well-placed to provide informed and constructive feedback. LibrePhotos has no marketing budget, hence only users, who researched possible solutions would have found it.

While this approach resulted in a relatively small sample size, it was justified by the depth and quality of expertise that these participants brought to the survey.

4.2.3 Technical Survey

A technical survey was conducted to gather server statistics of these power users, shedding light on various aspects of its operation. The collected data includes details about the server's CPU configuration, Python version, and system architecture. The CPU information includes things as the number of CPU cores, clock speed, and cache sizes, but more importantly which architecture extensions they support. The server's image tag, available RAM, and storage capacity were also documented.

Furthermore, the survey provided insights into the user landscape, with data on user registration dates, the total file sizes, the number of photos, videos, captions, albums, and other aspects of user activity. The survey even looked into user engagement, analyzing metrics like the number of clusters, places, people, and events associated with user content. Additionally, the survey recorded user statistics such as the number of favorites, hidden items, and public items.

This technical survey serves as a valuable resource for understanding the server's performance and the diverse user interactions within the LibrePhotos platform.

4.2.4 Questionnaire

The questionnaire on automatic image captioning aims to gather comprehensive insights into image captioning systems. The goals of the questionnaire are outlined as follows: Assessing users' perspectives on the strengths and weaknesses of the current image captioning implementation they have experienced. This includes an exploration of the system's performance, accuracy, and ease of use. Understanding how image captioning is utilized on diverse platforms. This goal seeks to uncover whether users have encountered similar or different captioning implementations across various applications or services. Exploring potential use cases of image captioning from the users' standpoint. This involves investigating the scenarios in which users find image captioning most valuable and where improvements could enhance its applicability. Gathering insights into users' personal experiences with image captioning. This includes preferences, satisfaction levels, and any challenges faced during the utilization of these systems. Identifying user expectations and suggestions for future improvements in image captioning technology. This goal aims to uncover areas where users believe advancements could enhance the performance, accuracy, and overall utility of image captioning systems. Exploring the diversity of systems currently in use by respondents. This involves gaining an understanding of the range of platforms employed, as well as any notable differences in user experiences across these systems.

4.3 Results

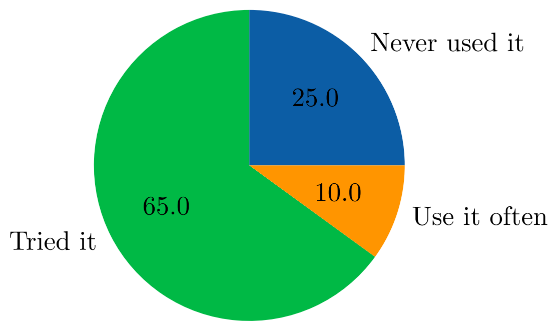


Figure 4.1: How frequently do you use the automatic image captioning in LibrePhotos? - n=20

According to the Figure 4.1, 65% of respondents indicated that they have tried the automatic image captioning feature in LibrePhotos. This suggests a significant level of interest or experimentation with the functionality. The fact that a majority have tried it implies a certain level of engagement or curiosity among users.

On the other hand, 25% of respondents reported that they have never used the automatic image captioning feature. This minority suggests that there is a segment of users who either may not been aware of the feature, find it unappealing, or have not had a need for it. Understanding the reasons behind this non-usage could provide valuable information for developers to improve user awareness or enhance the feature's appeal.

The remaining respondents, 10%, claimed to use the automatic image captioning feature regularly. This group represents the core user base that actively incorporates the feature into their workflow or finds value in its consistent use.

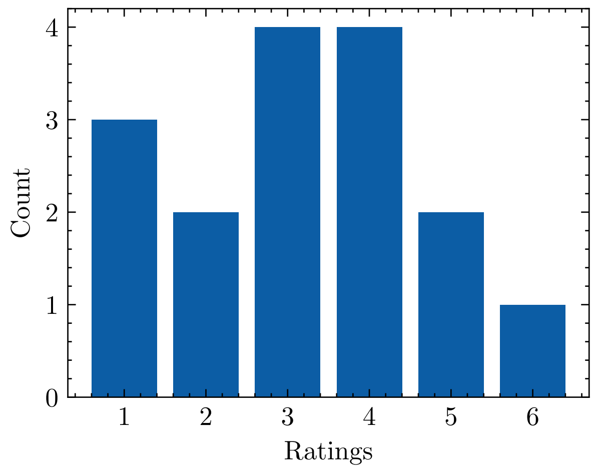


Figure 4.2: How satisfied are you with the accuracy and relevance of captions generated by the current image captioning systems? - n=16

Out of the 16 respondents, there is a notable distribution of satisfaction levels. The fact that only one person rated it a 6 suggests that there is room for improvement, and users are not overwhelmingly impressed with the current state of image captioning systems in terms of accuracy and relevance.

4.3.1 In your opinion, what are the strengths and weaknesses of the current image captioning implementations you've experienced? - n=11

A positive aspect highlighted in A2.1 by one respondent is the ease of use, indicating that the implementation is user-friendly. The integration with search functionality suggests a practical and efficient user experience. Another respondent acknowledges the ability of the system to correctly tag larger concepts like water, sky, etc., indicating a capability to understand broad and general elements in images. Some users report a high accuracy rate in recognizing and captioning objects, suggesting that the system performs well in identifying common elements within images. The system is noted for providing general descriptions, and in some cases, recognizing and tagging larger concepts accurately.
Multiple respondents express dissatisfaction with the accuracy of the captions, pointing out instances where the generated captions do not align with what is visible in the image. This suggests a significant room for improvement in the precision of the captioning model. Specific concepts within images pose a challenge for the system, as noted by a user. This indicates that the model may struggle with more nuanced and detailed elements in images. One respondent highlights a limitation in the system's ability to recognize and incorporate known people in captions, providing an example where the model fails to attribute a name to a person in the image. The system is reported to be quite slow on less powerful machines. This could be a significant drawback for users with limited computational resources, potentially affecting the overall user experience. Some users express a reluctance to use the captioning feature extensively due to concerns about accuracy and noise in the generated captions. This suggests that the perceived shortcomings may hinder the adoption and utilization of the feature.

4.3.2 On which platforms or applications have you come across automatic image captioning? - n=9

Less than half of the respondents in A.2.2 could name specific solutions for automatic image captioning. This suggests a potential gap in users' awareness of the availability and functionality of such systems. Some participants listed platforms like Google Photos and Amazon Photos, which do not explicitly offer automatic image captioning functionality. This discrepancy suggests a potential misunderstanding among users regarding the definition of automatic image captioning. It could be indicative of a broader understanding that encompasses features beyond textual descriptions. The mention of Centos 7.9 X86 as a response reflects a misunderstanding or misinterpretation of the question, as an operating system is not an automatic image captioning solution. This highlights the importance of clarity in survey questions to avoid confusion. One participant specifically mentioned using AWS Rekognition, which is accurate in the context of image analysis services. However, it's noteworthy that other AWS services provide features like image labeling and object detection rather than traditional image captioning. This indicates a potential conflation of various image analysis functionalities. The response mentioning "Nextcloud auto tags photos" provides insights into users' perceptions, as they seem to associate tagging with automatic image captioning. This indicates a broader interpretation of captioning that includes metadata or labels. Participants cited external tools like phosus.com/tools/image-auto-caption, Stable Diffusion and GPT indicating that some users actively seek or use third-party applications for automatic image captioning. While stable diffusion usually is not an image captioning system the most popular system [AUTOMATIC1111 2022] ships BLIP. This demonstrates a willingness to explore beyond built-in functionalities.

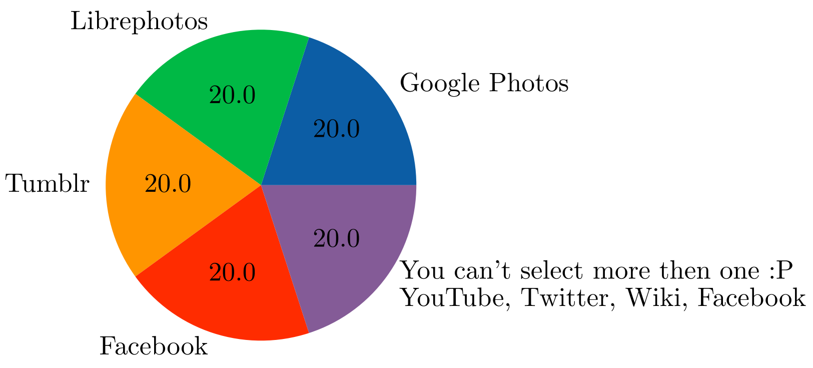


Figure 4.3: Please select all platforms where you used image captioning - n=5

With five people responding to this question, it seems to indicate that either users do not use captioning on other platforms or that they do not understand what was asked. Google Photos provides sharing functionality, where image captioning could make sense, if you want to add additional information about the image to your friends and family. Tumblr is known for its tagging system, which makes images explorable. Actual captions tend to have little in common with the image. Facebook is the best example here, because image can be public, hashtags are usually uncommonly used and they can be used for multiple purposes.

4.3.3 Have you encountered any image captioning systems that consistently stand out to you in terms of usability? - n=5

Only one response out of twenty participants seem to indicate that there is not yet a strong preference for user experience, when it comes to image captioning. Google Photos only provides a simple experience by adding a text area for the description with a button for saving the image.

4.3.4 Are there any platforms where you find the captions particularly helpful or necessary for better understanding visual content? - n=4

As the study suggests, accessibility considerations may not be at the forefront of users' minds. This insight underscores the need for ongoing efforts to raise awareness about the significance of captions in enhancing the inclusivity of visual content across digital platforms.

4.3.5 What do you want to accomplish with image captioning? - n=7

A possible use case is the linkage of image captioning to associated tools. This response implies a broader perspective, considering the integration of image captions with complementary tools for a more comprehensive workflow. This could be a reference to support for EXIF data. This indicates a technical aspect of image captioning, emphasizing compatibility with existing standards and formats for better integration into diverse workflows. A notable theme among the responses is the desire to improve the organization and search capabilities of image collections. Users express the need to search image libraries based on the content of the image, auto-generate albums, and create virtual albums. This indicates a practical goal of using image captions for efficient content management and retrieval. One user specifically highlights the importance of accessibility by mentioning the possibility of automatic audio descriptions for blind people. This response underscores the potential societal impact of image captioning in making visual content more accessible to individuals with visual impairments. Several respondents express the desire for improved tagging. This suggests that users perceive image captioning as a tool for enhancing metadata and, subsequently, the precision of search functionalities. The emphasis on accuracy points to a need for reliability in organizing and retrieving visual content.

4.3.6 Have you ever used image captions to search for specific visual information within a large collection of images in a photo management system? If so, how effective was the experience? - n=7

The responses indicate varying degrees of familiarity and utilization of image captions for search within photo management systems. Positive experiences highlight the potential effectiveness of this feature, while considerations of accuracy emphasize the importance of refining captioning systems for optimal performance, especially for power users who may heavily rely on such functionalities.
Three participants straightforwardly indicated that they have not utilized image captions for searching, suggesting that this feature may not be prevalent in their current image management routines.
Conversely, one user reported a very effective experience when using image captions for search, emphasizing the potential value of this feature for efficient navigation within a photo collection, while another provided a nuanced perspective, acknowledging the effectiveness of image captions but emphasizing the dependence on caption accuracy. While they generally found what they were searching for, occasional false positives and negatives highlighted the importance of precision in captioning systems, particularly in the context of managing extensive image collections. Lastly a third wanted to try it, but hasn't done it yet.

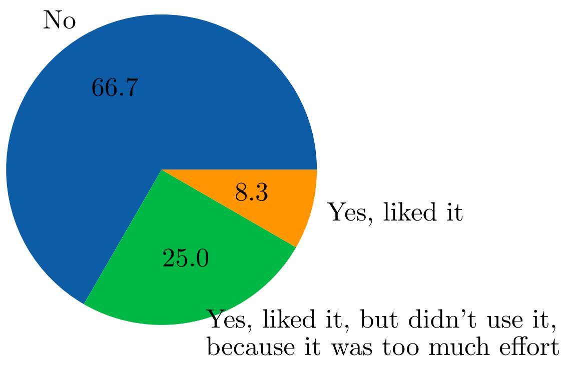


Figure 4.4: Have you ever used image captioning tools for organizing your photo collection or creating personalized captions for your images? - n=12

The majority, comprising two-thirds of the respondents, indicated that they have not utilized image captioning tools for organizing their photo collections or crafting personalized captions. This suggests that, at present, a significant portion of users may not actively engage in captioning as part of their photo management practices.

The remaining third of respondents acknowledged using image captioning tools, but not using it anymore as the effort involved was too much. This implies a potential barrier to widespread adoption, suggesting that manual captioning may be perceived as labor-intensive and may not align with users' preferences or priorities.

The insight that a notable portion of respondents finds manual captioning efforts burdensome presents an opportunity for automatic image captioning tools. Automation could address the perceived challenges and provide a more user-friendly solution for organizing photo collections and generating captions effortlessly.

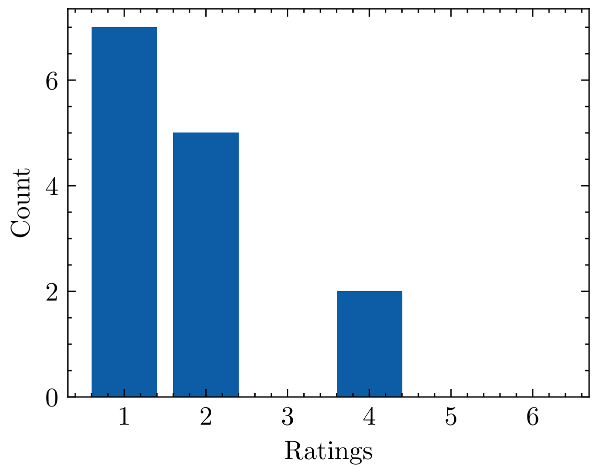


Figure 4.5: Would you prefer image captions that stick closely to the visual content, or captions that provide additional creative interpretations? - n=14

A significant majority of users, comprising most of the respondents, expressed a preference for image captions that stick closely to the visual content. This indicates a strong inclination toward factual and descriptive captions that directly reflect the elements present in the images. The emphasis on factual captions suggests a prioritization of accuracy and relevance over creative interpretations.

The observation that users prefer factual captions aligns with the notion that the respondents may not consider sharing captions to be particularly relevant. This inclination could be influenced by the target group or the specific use cases for which users are employing image captions. In scenarios where accuracy and clarity are paramount, such as organizing photo collections or searching for specific visual information, users may prioritize factual descriptions that closely mirror the content of the images.

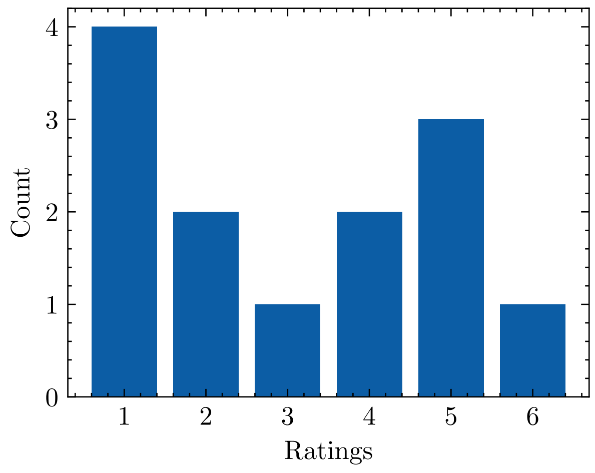


Figure 4.6: How important is the adaptability of image captioning systems in generating captions for different types of images, such as screenshots, memes or abstract art? - n=13

Among the nine respondents, there is no clear indication of a distinctive preference or emphasis on the adaptability of image captioning systems. The lack of a pronounced trend in the responses could imply several potential interpretations. Some participants might perceive image captioning primarily as a utility that should function consistently across diverse image types while others only use it for actual images instead of abstract things like memes and art.

4.3.7 What improvements or features would you like to see in future image captioning systems to enhance their usefulness? - n=7

User expectations for future image captioning systems encompass a range of features, including multilingual support, personalized recognition of faces, accuracy prioritization, OCR capabilities, and enhanced organizational tools such as virtual albums. These insights provide valuable direction for this master thesis to enhance the utility and user experience of image captioning systems.

4.3.8 Are there any concerns or reservations you have regarding the use of image captioning? - n=6

User concerns regarding image captioning primarily revolve around privacy, with an emphasis on keeping information within a user's network. Performance considerations, particularly regarding scanning speed, also emerged as a noteworthy concern.

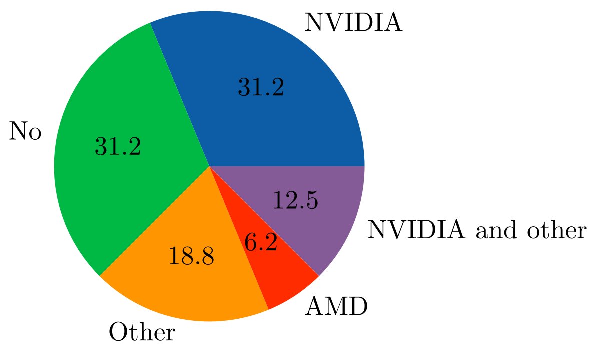


Figure 4.7: Do you plan to use Hardware Acceleration with a GPU if available? If yes, name the model you plan on using. - n=16

The graph is simplified in order to make the categories more visible. The full graph is available in the appendix. 31.2% of the respondents stated that they do not plan to use hardware acceleration with a GPU. This suggests that a significant portion of users may not prioritize or intend to leverage GPU acceleration for image captioning tasks. The rest of the respondents indicated that they plan to use hardware acceleration with a GPU. Notably, this group expressed a preference for models that offer wide compatibility indicated by naming GPUs and accelerators in the "Other" category with a clear preference for CUDA support. This suggests that users who opt for hardware acceleration prioritize solutions that are versatile and can be employed across a broad range of hardware setups, but currently are bound to the CUDA ecosystem.

4.3.9 Performance and Resource Targets

Any solution has to run on a diverse base of systems. In order to find that out, I will analyse the data sourced from the technical survey.

In order to compare the CPUs with each others, they were aggregated with a Passmark Score, specifically the CPUMark score. CPUMark is a component of the PassMark PerformanceTest suite, and it specifically focuses on benchmarking and evaluating the performance of CPUs. The

CPUMark score is a numerical representation of the CPU's overall performance based on a series of standardized tests. It's important to note that while CPUMark provides a standardized metric for CPU performance, the real-world performance of a CPU may vary based on the specific applications and tasks a user engages in. It still offers a good way to compare two CPUs with each others.

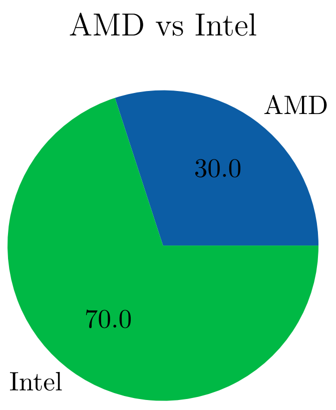


Figure 4.8: Brand of CPU - n=9

The data indicates a split of 30% for Intel CPUs and 70% for AMD CPUs. Notably absent from the graph are ARM systems, such as those found in Raspberry Pis or Apple's M1 processors. It's important to note that despite their absence in the graph, these ARM systems do exists in the user base, as the organization provides builds for this platform as well.

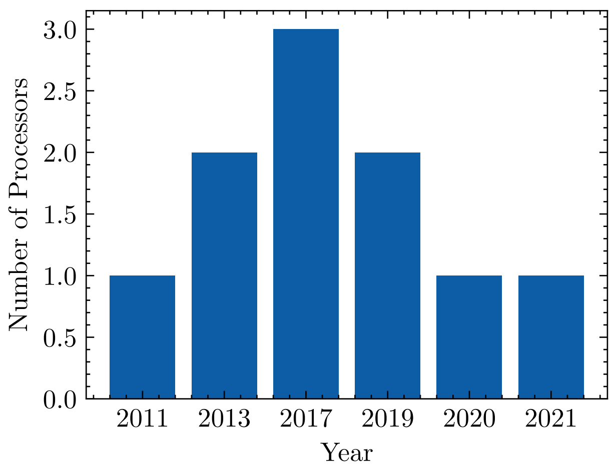


Figure 4.9: Number of processors compared to release date - n=9

There is a notable trend where a significant portion of the CPUs are 7 years or older, suggesting a propensity among users to utilize older processors. This trend may indicate a pattern of system upgrades, where users retain and repurpose existing hardware. The wide variety in CPU release dates within the sample indicates a diverse user base with different preferences, needs, and upgrade strategies. The split between users who opt for the latest CPUs and those who continue to use older models suggests a dynamic user landscape, where preferences and priorities vary widely.

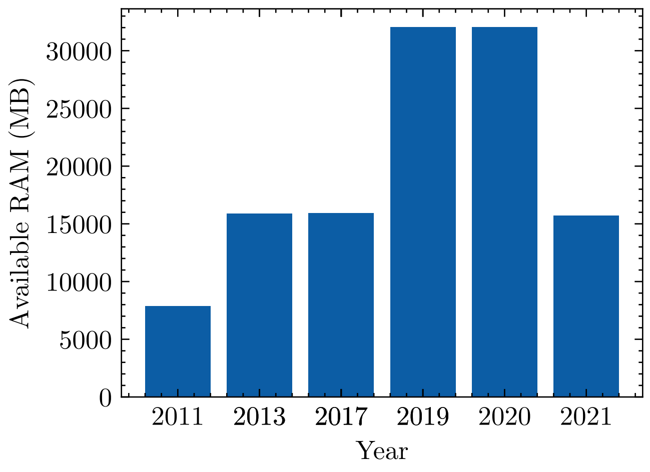


Figure 4.10: Age of CPU compared to available system memory - n=9

A correlation between newer CPUs and increased system memory is immediately visible. While the cost of system memory is similar consistent across different CPU platforms, the observed trend may be indicative of different use cases, with newer CPUs possibly associated with higher budgets or a focus on performance, and older CPUs reflecting a self-hosting, budget-oriented hobbyist approach.

While there is an absence of Raspberry Pis in the sample, and it is noted that these systems typically ship with 4GB of RAM and only sometime come with 8GB. M1 processors, come with a minimum of 8GB of RAM, potentially reflecting the reported values.

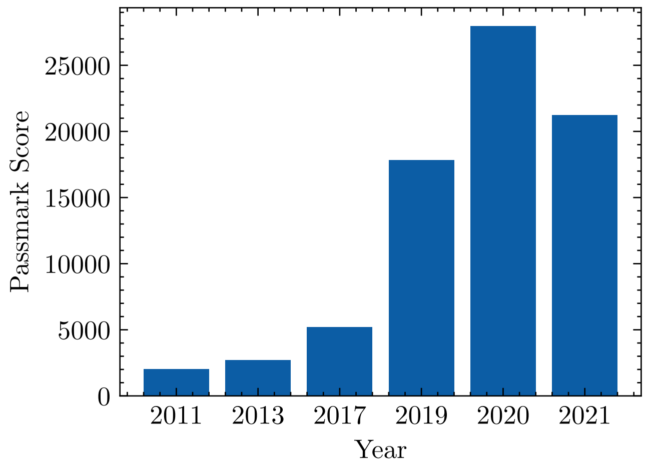


Figure 4.11: Age of CPU compared to average passmark score - n=9

Similar to the available system memory on newer CPU's, the general processing power is also a magnitude higher with a strong preference for high end CPU's like AMD Ryzen 9 5900X. An notable exception is the J4125, which is from 2019, but also very low performance. The score of just 2000 is 20x lower the the fastest system. These processors get shipped usually by the company Qnap and Synology in budget home servers, where the primary use case are network attached storage and get sold on the premise of low energy consumption.

The slowest processors is an i5-2415M with an Passmark score of 2013, which is compared to the fastest processor AMD Ryzen 9 3900X with an score of 32678, 16 times slower.

Also notable is, that multiple processors use the U variant of processors, indicating that old laptops are used as home servers as they are energy efficient and provide a power backup solution with the battery.

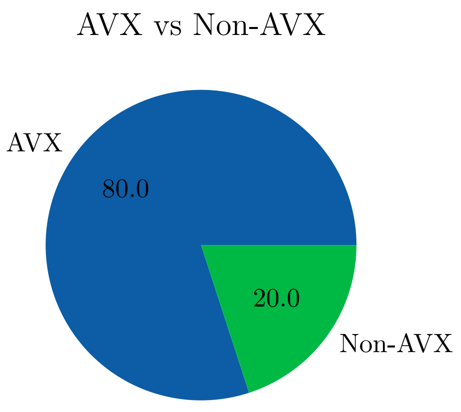


Figure 4.12: AVX compatibility across all processors - n=9

Older processors and budget processors like the J4125 lack certain modern x64 extensions. This standard is constantly evolving. AVX and AVX2 should provide faster computation of vectors, which usually gets used by machine learning libraries in order to speed up computations. An example is dlib a library, which provides facial recognition. This is only compiled against one architecture and crashes if a certain instruction set is missing.
There are also even older standards which are sometimes missing if the processor is 15 years or older like SSE4, which for example crashed LibrePhotos, because pytorch did not handle that correctly anymore. [LibrePhotos 2023] [PyTorch 2024]

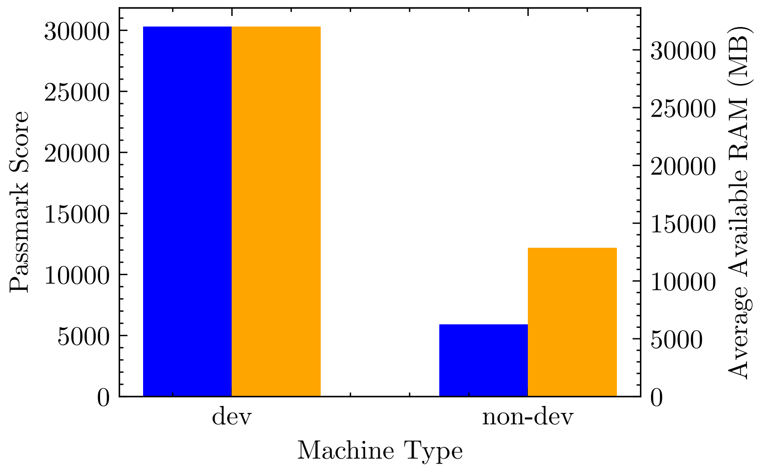


Figure 4.13: Dev vs Non-Dev - n=9

LibrePhotos provides two different kind of releases. On is an rolling releases, which gets build after every commit, which in this graph is called dev. On the other hand LibrePhotos has monthly releases, which provide more stability, as I usually try to fix bugs before creating a new one. This graphs shows, that power users who use cutting edge releases also tend to have way more processing power available, probably because they actively work on the project, providing bug reports to open source projects. Monthly releases on the other hand, seem to run with less system memory and low processing speed, indicating that energy saving and using older systems is more prevalent.

4.4 Stackholders and Usecases

There are three different roles users can have, when using the system. A user can have one or all three of them. The first are the regular users of the system e.g. a family member, who uses a self-hosted system from somebody else. They cannot change settings, which model should be used, but they can generate captions for their images. When an system with an LLM is activated, they may want to tweak the prompt based on their preferences or want a slider for creativity. They want that hashtags create new albums. They want keywords as a separate field too.
The second group is the administrator of the system. They create new users, decides which model to use and helps users of his self hosted system. They want insights how well his server can handle the different models and get a warning if a model consumes too much RAM. They also want to deactivate the LLM or activated it based on their preferences.
The third group are the operations people. They want a stable system, which just runs and updates nicely. They want no breaking changing for his setup. He does not want to change his kubernetes cluster or his docker-compose stack and want to just update it automatically. They maybe like to deactivate features with environment variables, which overrides want the admin can set. They want some documentation on how the system downloads the models from and where they are saved.

4.5 Functional Requirements

Functional requirements detail the specific behaviors and functionalities that the software must exhibit to fulfill the users' needs and expectations. These requirements are derived from both user surveys and the technical survey.

- Must generate natural language with valid grammar
- Must be better in common benchmarks than previous system
- Must work with all image formats (e.g., JPEG, PNG)
- Must be self-hosted
- Must be private
- Must scale for a few users
- Must generate factual captions based on the image content
- Total memory consumption must be under 8GBs
- Must not use SSE4, AVX and AVX2
- Must not result in breaking changes
- Must display generated caption
- Must be able to switch between image captioning systems

4.6 Non-Functional Requirements

Non-functional requirements encompass the quality attributes and constraints that define how a software system performs rather than what it does. User expectations gathered from the surveys provide valuable insights into the non-functional aspects of the software, such as its responsiveness, ease of use, and overall user experience.

- Translation into a different languages other than english
- As accurate as possible
- Prioritization between accuracy and speed
- Optional, as scanning images is too slow
- Should use previous attained knowledge like persons
- Should read text
- Should generate tags
- Should have a slider for creativity
- Grouping by concepts in virtual albums
- Should provide CUDA acceleration
- Should be hardware accelerated on multiple niche platforms (AMD, Apple Silicon, Google Coral)
- Allow users to customize the style, tone, and verbosity of the generated captions. Support personalized user profiles for better results.
- Should provide a good user experience for editing captions and a cohesive workflow for adding generated captions
- Should integrate with third party tools and handle EXIF data
- Should respond fast on a wide variety of CPUs

Chapter 5

Implementation

In order to understand where our starting point is, I have have to establish a base line of the performance of the old model.I have to look at the different benchmark scores, but also how much system memory is consumed and how long the responses for generating a caption take. I will look into different techniques for improving inference speed and system memory consumption and will evaluate these techniques between models.

5.1 Systematic Testing

To assess the performance and efficiency of various image captioning models, a systematic testing approach was employed, focusing on key metrics, runtime comparisons, and resource utilization. The testing procedure aimed to provide insights into the models' capabilities, their execution times, and resource requirements across different evaluation criteria.

5.1.1 Experimental Setup

The testing environment utilized an `Intel(R) Core(TM) i7-8750H` CPU, `NVIDIA GeForce GTX 1050 Ti with Max-Q Design`, and `32GB RAM` with a CPUMark Score of 9942. This setup ensured a consistent hardware configuration for fair and reliable performance evaluations.

5.1.2 Evaluation Metrics and Parameters

The evaluation metrics employed for benchmarking the image captioning models included BLEU scores (BLEU-1, BLEU-2, BLEU-3, BLEU-4), METEOR, ROUGE_L, CIDEr, and SPICE. Additionally, the mean and median RAM usage and execution time were tracked to assess the computational efficiency of each model.
The information logged for each experiment consisted of the following parameters:

- **Timestamp:** Date and time of the experiment.
- **Model:** The specific image captioning model under evaluation.
- **Dataset:** The dataset used for testing (Coco validation set).
- **Device:** Hardware specifications of the testing environment.
- **Bleu_1, Bleu_2, Bleu_3, Bleu_4:** BLEU scores for unigrams, bigrams, trigrams, and quadgrams.
- **METEOR:** Metric for Evaluation of Translation with Explicit ORdering.
- **ROUGE_L:** Longest Common Subsequence-based metric.

- **CIDEr:** Consensus-based Image Description Evaluation.
- **SPICE:** Semantic Propositional Image Caption Evaluation.
- **Mean RAM Usage:** Average RAM consumption during the execution of the model.
- **Median RAM Usage:** Median RAM consumption during the execution.
- **Execution Time:** Time taken by the model to process the entire Coco validation set.
- **Note:** Additional notes or observations relevant to the experiment.
- **Version:** Version information of the image captioning model.

Implementation

In order to implement a comprehensive evaluation of our test runs, integration of the following dependencies had to be done.
pycocotools [ppwwyyxx 2014] is essential for reading COCO (Common Objects in Context) annotation files and corresponding images. It facilitates seamless access to the dataset's annotations, enabling accurate evaluation. pycocoevalcap [salaniz 2015] generates general scores based on COCO annotation files and the results produced by the image captioning model. This is crucial for quantifying the performance of the model against established benchmarks.
nltk [Bird et al. 2009] is a powerful library and in this context a dependency of pycocoevalcaps, for natural language processing. In the context of image captioning, it aids in linguistic analysis and evaluation of generated captions.
The default Java Runtime Environment (JRE) is required for certain functionalities within pycocoevalcap. Integrating this dependency ensures the seamless execution of evaluation processes that rely on Java components.
The question of how to ship these dependencies was addressed by introducing a new flag in the build system. This flag facilitates the installation of the required dependencies during the build phase, ensuring a consistent and reproducible testing environment. [1]

5.2 General performance enhancements

While some techniques are special when it comes to machine learning, some techniques are simple and fast to implement and easily accessible.

5.2.1 Implementing a simple class with loading / unloading function

The basis for finding possible bottlenecks is to implement a benchmark and then looking what part of the program takes the longest. The first step was to create a simple case, where one image would be captioned one time and a hundred times. The results were that the CPU was faster when generating the caption for one image, but the GPU was faster when generating one hundred captions, but not by how much one would expect. I concluded that this is because the loading and unloading of the machine learning model takes up most of the time and the actual generation time does not get captured well. I solved this issue by implementing a class, with an load, an unload and a generate function. This feature also has additional benefits, like adding more models to it, which I then can control. [2]

[1]Full code for testing: `https://github.com/LibrePhotos/librephotos/blob/dev/api/tests/test_im2txt.py`

[2]Full code for loading / unloading: `https://github.com/LibrePhotos/librephotos/blob/dev/service/image_captioning/api/im2txt/sample.py`

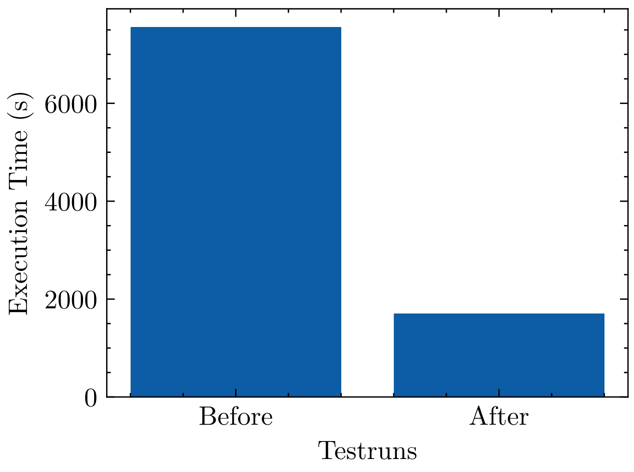


Figure 5.1: Before and after implementing a class with load and unloading functions

The graph show on the y axis the execution time in seconds and on the x axis the before and after implementing a class with a load and unloading functionality when using a CPU. This graph shows the execution time of the COCO validation set, instead of the described test above in order to keep all the graphs in this section comparable. After that change the CPU generation was five times faster. This change would have been even more effective on system, which rely on classic HDDs and not like in this case an SSD as transferring speed was the limiting factor.

5.2.2 Support GPU acceleration

Another common method on improving inference speed is using a GPU for acceleration. As previously stated this is a common request from the survey. The framework PyTorch supports CUDA acceleration, which uses NVIDIA GPUs to improve inference speed. The graph shows

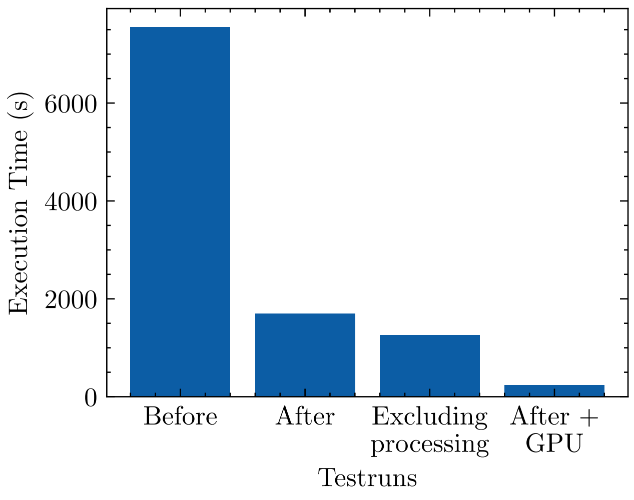


Figure 5.2: Before, after, excluding processing time of scores and after + GPU acceleration

again on the y axis the execution time in seconds and on the x axis the before and after implementing a class with a load and unloading functionality when using a CPU and additionally the implementation with using a GPU. This results in another huge speed up. This is however not a general improvement as not all users have access to an GPU, which means that I will always track both methods when looking on implementing a new model.

5.2.3 Failed attempts

I used the new torch.compile function and compared it to ONNX. This was not faster and it was not possible to save the converted model, which makes inference unbearably slow.
Updating the version to the cutting edge also did not have any performance impact. Moving from PyTorch 2.0 to 2.1 did not improve performance. I also tracked Ram Usage. To double-check if this changes the numbers, I reran the tests and concluded, that this also has no effects. Ensuring consistent input size was also a non issue, because the tested models already implemented that.

5.3 im2txt

The im2txt repository is based on a tutorial, and the original maintainer just copy and pasted this code into the repository. It has an encoder, decoder architecture. The files needed to run it are the encoder file, the decoder file and a vocab file.

5.3.1 ONNX

As previously outlined ONNX can be used to allow for the usage for multiple accelerators and improve speed by either quantizing or through speed ups in the inference engine. This solves two acceptance criteria from the questionnaire: Running with acceptable speed and allowing for accelerating for diverse systems. ONNX however is a cutting edge technology, which means in practical terms gaps in the documentation and the implementation of conversion features. This resulted in multiple challenges with exporting models to it. To facilitate the conversion of im2txt to ONNX, the integration of a new module is necessary. The ONNX package typically includes functionalities and tools specifically designed for exporting PyTorch models to the ONNX format. The inclusion of a new module for ONNX conversion necessitates an update to the requirements.txt file and the general shipped dependencies. This is fine, because the ONNX modules is also used for inferring from an exported model. In order to export the current model to ONNX pytorch provides two different exporters. torch.onnx.export is shipped with pytorch, and relies on TorchScript. The shortcomings are that it can't handle dynamic inputs, which should not be relevant as our image the system wants to describe is always the same size, handles no control flow and can't differentiate between eval and train mode. None of this is relevant and should export correctly. The new torch.onnx.dynamo_export handle these edge cases better according to the documentation. Both need an additional dependency called onnxscript to export models to ONNX, which is added in the same requirements.mlval.txt like the pycocoevalcap dependency as this is not needed in production.
There were two methods on the decoder, which could potentially generated captions. The "forward" method and the "sample" method. After trial and error and researching, I found out that usually the forward function is used to signal, that this code is used for inference. This function will also be used by ONNX, when exporting the model. For some reason, the code diverged from this standard and only used the custom "sample" method instead. By simply renaming it to forward and removing the old forward function, I could export the model to ONNX. This only worked with the static exporter. The dynamic one, resulted in models, which were not responding, which is probably a bug. ONNX advertises that models which are exported to this standard will perform better. While the initial ONNX export worked, I noticed that it didn't perform as well as my PyTorch model. After some investigation, I realized that unnecessary data conversions were slowing things down.
After resolving this issue, I managed to improve the model's performance by about 10-15 percent with the conversion, which was a step forward in my image caption generation project, but also not as fast as advertised on the landing page of ONNX.

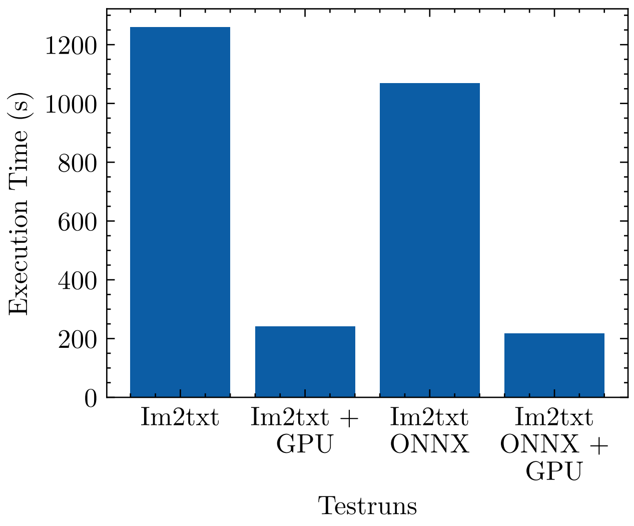


Figure 5.3: Im2txt vs Im2txt ONNX with and without GPU

5.3.2 Quantization

The same issue of multiple choices to solve one problem arises when quantizing the ONNX model. ONNX provides two different options dynamic quantization and static quantization. The first issue I had that it is not implemented for all types and does not work for LSTM layers, which the model uses in the decoder or on convolutional layers which im2txt uses in the encoder. You can exclude these nodes by marking them, but again, the quantized models stopped responding. Overall this step was not successful, but it was also not as relevant as they are fast enough with the current implementation.

5.4 BLIP

I selected BLIP, because the paper sounded promising. I chose the model_base_capfilt_large model, which was trained using the filtered captions and in the largest size they offered.

Implementation removed training code in order to only add a couple of new dependencies and remove unnecessary ones like fairscale. This was done by only importing the classes needed and selectively removing the rest. Updating to transformers 4.16 was needed in order for the model to run as transformers 4.15 would need rust for installation. The newest version of transformers also did not work.

Image Captioning Benchmarks								
Model	Bleu$_1$	Bleu$_2$	Bleu$_3$	Bleu$_4$	METEOR	ROUGE$_L$	CIDEr	SPICE
im2txt	66.4	48.4	34.3	24.5	22.8	49.1	79.8	15.8
im2txt ONNX	66.4	48.4	34.3	24.5	22.8	49.1	79.8	15.8
BLIP	65.7	53.4	41.5	31.4	24.6	54.8	102.2	19
BLIP finetuned	-	-	-	39.7	-	-	133.3	-

The im2txt model exhibits the lowest performance, as evidenced by scores across all n-gram orders (Bleu_1 to Bleu_4). The METEOR score of 22.8 suggests a moderate level of precision, recall, and alignment with human judgments. A notable overlap between the generated and reference captions is indicated by the ROUGE_L score of 49.1. Moreover, the CIDEr score of 79.8 attests to some diversity and quality of the generated captions. The incorporation of some semantic understanding is reflected in the SPICE score of 15.8.

The im2txt converted to the ONNX format, maintains performance parity with the original im2txt, as evidenced by identical metric values across all evaluated benchmarks. This suggests that the conversion to ONNX does not significantly impact the model's image captioning capabilities.

In contrast, the BLIP model surpasses im2txt in various metrics, showcasing particularly notable performance in Bleu_2 and Bleu_3. The METEOR score of 24.6 represents an improvement over im2txt, indicating enhanced precision, recall, and alignment. Furthermore, the ROUGE_L score of 54.8 highlights an increased overlap with reference captions, while the CIDEr score of 102.2 underscores improved caption quality and diversity. The SPICE score of 19 indicates an enhanced understanding of semantic content.

The BLIP performance reported in the paper, after undergoing fine-tuning, demonstrates significant improvements, especially notable in the Bleu_4 score (39.7) and the exceptionally high CIDEr score of 133.3. This underscores the efficacy of fine-tuning in enhancing the model's captioning capabilities.

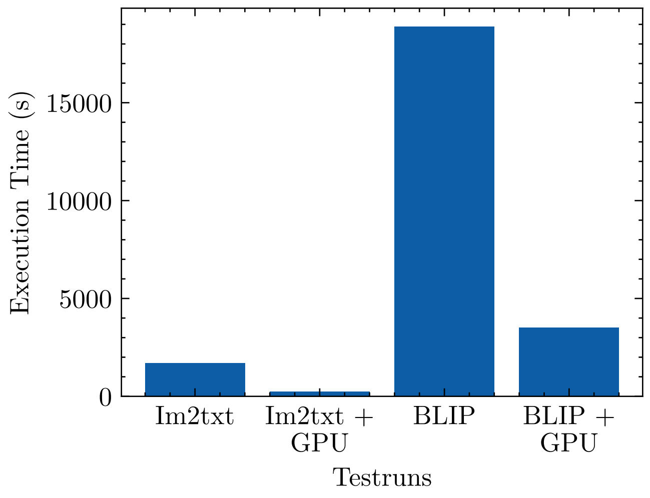


Figure 5.4: Im2txt vs BLIP Execution Time with and without GPU

Notably, the BLIP model, despite achieving improved captioning performance, exhibits a considerable execution time on both CPU and GPU. On a CPU, BLIP demands 18883.41 seconds for completion, signifying a significant computational load. Even when executed on a GPU, the execution time remains high at 3508.29 seconds, suggesting that the gains in captioning quality come at the expense of a substantial increase in computational demands. This emphasizes the resource-intensive nature of the BLIP model, potentially limiting its real-time applicability.

The RAM usage remained comfortably below the 8GB threshold for the entire system when employing im2txt and BLIP, fluctuating within the range of 1 to 2 GBs in a minimal configuration. It's worth noting that the overall system RAM utilization tends to increase when CUDA is utilized, potentially due to the loading of PyTorch drivers into memory.

5.5 ONNX Conversion Challenges for BLIP Models

The conversion of BLIP models to ONNX format proved to be a challenge, encountering multiple errors and roadblocks across various attempted approaches. In this chapter, I delve into the journey of attempting to convert BLIP models to ONNX and the reasons behind the encountered failures.

5.5.1 Huggingface Optimum Tool

Initially, the exploration involved using Huggingface's third-party tool, Optimum, designed to work seamlessly with most models implemented in the Transformers library. However, it was discovered that BLIP to ONNX conversion was not supported by Optimum. The error message indicated the need for a custom ONNX configuration when dealing with a custom or unsupported architecture for tasks such as image-to-text. The absence of a pre-defined image-to-text to ONNX pipeline and the lack of native support for BLIP models in ONNX export via Optimum presented a significant hurdle. The thesis project scope did not extend to implementing such support, leaving the option of opening an issue on the Optimum GitHub repository for potential future enhancements.

Error message

```
Trying to export a blip model, that is a custom or unsupported architecture for
the task image-to-text, but no custom onnx configuration was passed as
‘custom_onnx_configs‘. Please refer to https://huggingface.co/docs/optimum/main/
en/exporters/onnx/usage_guides/export_a_model#custom-export-of-transformers-models
for an example on how to export custom models. Please open an issue at
https://github.com/huggingface/optimum/issues if you would like the model type
blip to be supported natively in the ONNX export.
```

5.5.2 ONNX Export with Dynamo Variant

Error message

```
RuntimeError: number of output names provided (1) exceeded number of outputs (0)
```

In the pursuit of alternative methods, an attempt was made to export the entire BLIP model using the ONNX Export with the dynamo variant. Unfortunately, this approach led to multiple errors, with one notable instance producing a RuntimeError indicating a mismatch in the number of output names provided and the number of actual outputs. This inconsistency hindered the successful export of the model and posed a challenge that required further investigation. There is also an open issue on GitHub in the PyTorch repository about it. [vivekkhandelwal1 2023]

5.5.3 Separate Export of Encoder and Decoder

As a workaround, an effort was made to export the encoder and decoder components of the BLIP model separately. While the encoder export succeeded, the decoder export encountered "Not Yet Implemented (NYI) Errors." A thorough examination revealed that these errors were associated with named tensors in the PyTorch library that were not fully supported for ONNX export. [wangzy0327 2023] [abhimanyuchadha96 2021] [warrenburch 2023]

5.5.4 Quantization Challenges

In a final attempt to salvage the situation, considering the benefits of model quantization, the ONNX Export process faced continuous failures, preventing successful quantization. The persistence of issues, including those related to the Dynamo export and unsupported functionalities in PyTorch, hindered progress in achieving a quantized ONNX representation for the BLIP model.
In conclusion, the conversion of BLIP models to ONNX format proved to be challenging due to a combination of unsupported functionalities, lack of native support in third-party tools, and specific issues associated with ONNX export in the PyTorch library. These challenges highlighted the complexity of adapting custom or novel architectures for export to ONNX and underscored the need for enhanced tooling and support within the deep learning community.

Chapter 6

Integration of Image Captioning with a LLM

In this chapter, I embark on advancing image captioning within self-hosted personal photo management systems. Our objective is to develop a groundbreaking image captioning system that addresses existing challenges and introduces features to enhance user experience, providing a more versatile and personalized approach. Existing image captioning systems often struggle to integrate prior knowledge, such as identified individuals in the image. The lack of contextual information hinders the quality of generated captions. Current systems lack the ability to personalize captions based on user preferences, including factors like factual information, tone, or style, thereby limiting the adaptability of captions. The creation of keywords and captions is typically disjointed across different systems, resulting in a fragmented user experience. A cohesive solution is needed to streamline these tasks. This chapter will provide a comprehensive comparison of our novel approach with baseline methods through a user study. Additionally, I will evaluate our system with state-of-the-art benchmarks like CLIPScore, offering insights into the efficacy and performance of our integrated image captioning system.

6.1 Selecting an LLM

When integrating image captioning with a Large Language Model, the choice of the LLM is a critical decision influencing the overall performance and capabilities of the system. I will look at factors like availability, licensing, resource demands and performance.

6.1.1 Availability

The practice of open sourcing machine learning models is common practice and has gained significant momentum within the field of large language models, with a predominant focus on providing accessible models for finetuning and pre-trained models. It is noteworthy that not all models are made publicly available. A notable example is OpenAI's GPT-3 and GPT-4, which are exclusively accessible through an API, exemplifying a departure from the conventional open sourcing approach. While this decision made OpenAI profitable and resulting in fast growth, non-dominant players in the market try to secure market share by open sourcing smaller models. This dynamic is favorable for us as this master thesis and LibrePhotos focuses on self-hosted systems.

The key criteria for evaluating large language models are self-hostability and availability. Models should be self-hostable, with pretrained versions readily accessible to the research community. On the other side it should be able to handle basic semantic tasks. I need the capability that the model understands, which parts it should improve with other information it already knows about the image.

A subset of LLMs that particularly piques our interest comprises the 7B models, which are models with 7 billion parameters. Among these, the LLaMA 2 model, developed by Facebook, stands out. The original model got leaked resulting in Meta open sourcing the system and an ecosystem implementing inference engines for it.

Another noteworthy entrant in this category is the Mistral 7B model, developed by a small startup based in France, established in 2023. This model has garnered attention for reporting competitive results, despite its comparatively smaller scale. Notably, the Mistral 7B model outperforms the LLaMA 2 13B model across various benchmarks, as indicated in the release blog post: "Announcing Mistral 7B" [Mistral 2023a]. However, it is crucial to exercise caution when interpreting benchmark results, as highlighted in related works, as they may not provide a comprehensive representation of model performance. For example in the arena benchmark, the Mistral 7B model demonstrates comparable performance to the LLaMA 2 7B model, with only select finetuned variations surpassing the LLaMA 2 13B model [LMSYS 2023].
As of December 2023, Mistral AI has published two models, which are available as weights, while an additional model, Mistral Medium, is exclusively accessible via API.

6.1.2 Licensing

Software in open source can have different terms on which you can distribute them. LibrePhotos uses the MIT License (Massachusetts Institute of Technology License) [LibrePhotos 2023], which is a permissive open-source license that allows users to do almost anything they want with the software, as long as they include the original copyright and license notice in any copy of the software or substantial portion of it. It's a short and simple license that is widely used and easy to understand.

Mistral uses Apache 2.0 [Mistral 2023b], which is favorable for LibrePhotos. The Apache License 2.0 is another open-source license that, like the MIT License, is permissive. It allows for the use of the software in both open-source and proprietary projects. The Apache License 2.0 is often chosen for projects that want to encourage contributions from a wide range of sources.

Both the MIT License and the Apache License 2.0 are considered permissive licenses, and they are generally compatible. Compatibility here means that code released under one license can be combined with code released under the other license in the same project.

The compatibility arises from the fact that both licenses have similar terms regarding permissions and conditions. Both allow for modification, distribution, and sublicensing with the requirement to retain the original copyright and license notices.

LLAMA 2 has a unique license called LLAMA 2 Community License Agreement [Facebook 2023]. Like MIT and Apache, LLAMA 2 allows users to use, reproduce, distribute, and modify the software. Attribution is required in distributed copies, similar to MIT. All three licenses disclaim warranties and limit liability. The drawbacks are however, that LLAMA 2 includes specific clauses related to monthly active users and potential additional commercial terms, which are not present in MIT and Apache. LLAMA 2 has explicit restrictions on using the materials to improve other large language models, a provision not found in MIT or Apache. The indemnification clause in LLAMA 2 is more detailed and specific compared to MIT and Apache.

This makes LLAMA 2 from a licensing point not viable as LibrePhotos needs to be open source and in compliance with the MIT license.

6.1.3 Resource Demands

In assessing the viability of open source language models, an indispensable criterion is their self-hostability, particularly on consumer-grade hardware. Based on our survey, it is imperative that these models operate efficiently on systems with a modest 8GBs of RAM, given the prevalent hardware configurations reported by our user base.

Notably, GPU acceleration with a substantial VRAM allocation is impractical, as only a fraction of users possess GPUs with 8GBs or more of VRAM. This observation aligns with the broader gaming community, as illustrated by the Steam survey of December 2023, revealing that only 55% of high-end users possess 8GBs or more of VRAM [Steam 2024]. Our internal survey further substantiates this, with less than 5% of our user base reported to own GPUs with 8GBs of VRAM or more.

LLMs, being at the forefront of technological innovation, are inherently resource-intensive, both during training and inference phases. The immense computational demands are evident in the significant investments made by hyper scalers like AWS, Azure, and Meta [Sharwood 2023]. Meta alone is projected to spend a staggering $18 billion on new hardware in the current fiscal year, as disclosed by the CEO [@zuck 2023]
While acknowledging the cutting-edge nature of LLMs, our goal is to ensure compatibility with consumer hardware, specifically targeting middle to high-end systems. This approach seeks to strike a balance between harnessing the capabilities of advanced models and ensuring accessibility for a wider user base.
Recognizing the limitations imposed by our user base's hardware configurations, our aim is to develop or select models that operate efficiently with less than 8GBs of RAM.

6.1.4 Performance

The evaluation of model performance is pivotal, especially concerning the systems on which it is intended to operate. Our focus lies in determining the efficiency of LLMs within the context of our setup, with a particular emphasis on response time and interactivity. Key performance metrics, including Tokens per Second, load time and overall evaluation time, were measured across multiple instances to gain a comprehensive understanding. The primary consideration in our performance evaluation is to ensure that the language model's response time aligns with the timeout window of our backend.
The responsiveness of the language model is pivotal in creating an interactive user experience. The ability to provide feedback within the 100-millisecond threshold enhances the perception of real-time interaction, contributing to a more engaging user interface. Tokens per Second serves here as a key metric in gauging the throughput and efficiency of the language model. This metric provides insights into the model's processing speed, directly influencing its suitability for real-time applications.

6.2 Implementation

For the implementation of the execution engine for Mistral 7B, I employed llama.cpp along with its Python wrapper. Initially cobbled together to support LLAMA models, this setup has undergone significant refinement, evolving into a standard implementation for running Large Language Models (LLMs). llama.cpp incorporates numerous specialized techniques tailored to enhance token generation from LLMs, rendering it a robust inference platform.

Given the challenges encountered in quantizing image-to-text models, I explored prequantized alternatives. One notable obstacle was the prohibitive memory requirements of full-sized models, which can demand up to 32GB of RAM on development machines. Fortunately, "TheBloke," a contributor on Hugging Space backed by a16z investment, offers quantized models featuring various quantization levels and finetunes [TheBloke 2023]. After careful consideration, I opted for the Q5_K_M model, which weighs 5.13 GB and purportedly consumes up to 7.63 GB of RAM, as per the documentation. These specifications align well with the parameters observed during our study, making it a suitable choice for our implementation.

There's been a noticeable shift in the formats used to provide models for Large Language Models (LLMs). Previously, models were commonly offered in formats such as Safetensors and .pth, which essentially represent weights and have become standard in other deep learning algorithms. However, LLMs typically come in formats like GGML, EXL2, or, in our specific case, GGUF. The rationale behind these alternative formats lies in addressing specific challenges. They aim to eliminate the need for additional dependencies to load or save the model, while also facilitating support for 4-bit quantization. Additionally, these formats strive to ensure a consistent experience whether models are stored as files or directories, embed vocabularies within the file itself, and offer greater control over the structure and direction of the formats [ggerganov 2023].

When utilizing Mistral 7B, various metrics are reported, all of which hold relevance to this implementation. The initial metric pertains to load time, consistently ranging between 15s and 16s. This duration signifies the necessity for manual loading and unloading, as it exceeds the threshold for an interactive user experience. The prompt evaluation time denotes the duration required to process the tokenized prompt message, crucial for providing context for the model to predict subsequent tokens. This falls within the range of 200 to 240 ms per token. Similarly, the evaluation time refers to the duration needed to generate all tokens in response to the prompt, excluding pre-processing time. This measures approximately 400 ms per token.

The total time falls between 20 and 30 seconds, remaining within our timeout window. Notably, loading time consumes half of the total duration, suggesting that manual unloading and loading could significantly enhance responsiveness. However, both the 200 ms and 400 ms intervals are deemed lengthy for achieving a true interactive experience. Enhancing the user experience could involve adopting a method where tokens appear sequentially, as commonly observed in contemporary chat bot interfaces.

6.2.1 Fine-Tuning Captions Based on User Prompts

When it comes to making LLMs work well, figuring out how to ask it things is like an art form. I tried lots of different ways, but eventually stuck with one prompt, because it seemed to work the best.

Using slider and checkboxes is a better design choice, when prompting can lead to very different outcomes. If users have to come up with their own ways, it often leads to problems, which will result in me as a maintainer responding to more support requests. The slider and checkboxes make things simpler, avoiding issues with confusing prompts and making the whole system work more smoothly.

6.2.2 Previous attained knowledge

In the context of managing substantial amounts of data, LibrePhotos employs metadata extensively and supplements it with additional information about photos. Notably, the system integrates a reverse geocoding feature, facilitated by a chosen map provider, to add location details. Person associated with images are identified either through a face recognition pipeline or extracted from metadata. Utilizing XMP:RegionInfo for metadata, the face recognition process involves creating bounding boxes, associating names, and, when necessary, adding new individuals. The face recognition pipeline works the following way: The face recognition pipeline consists of different face detection algorithms, converting the faces into 512-dimensional embeddings and organizing faces into clusters. After the initial scan, the recommended workflow involves labeling, training, and verifying face inferences to optimize face recognition results. Users have the ability to label, remove or change identified faces.

Considering additional capabilities, such as adding text through OCR processes, raises questions about user preferences for detailed descriptions. It's crucial to address acceptable use cases and potential challenges, including the tendency of large language models to hallucinate or struggle with specific user requirements. Exploration into user preferences regarding the level of detail in descriptions becomes an intriguing aspect of potential future developments.

6.3 Evaluation

I delve into the comprehensive evaluation of our novel image captioning system integrated with Mistral 7B, taking into account the challenges addressed and the unique features introduced. The evaluation aims to provide a thorough understanding of the system's performance, efficacy, and user-centric benefits.

6.3.1 Prompt Engineering

This code segment is part of a system for dynamically constructing prompts to improve image captions, incorporating additional context about the image, and then logging and processing the generated prompt.

```
if site_config.LLM_MODEL != "None" and settings["enabled"]:
    face = api.models.Face.objects.filter(photo=self).first()
    person_name = ""
    if face and settings["add_person"]:
        person_name = " Person: " + face.person.name
    place = ""
    if self.search_location and settings["add_location"]:
        place = " Place: " + self.search_location
    prompt = (
        "Q: Your task is to improve the following image caption: "
        + caption
        + ". You also know the following information about the image:"
        + place
        + person_name
        + ". Stick as closely as possible to the caption, while replacing generic information
    )
    caption = generate_prompt(prompt)
```

At first it checks if the LLM is enabled and downloaded. Then it queries the database to retrieve a Face object associated with a particular photo. The system will then add the first person it finds to the query. LibrePhotos will then construct sub-prompts based on the conditions e.g. generate a sentence describing the persons in the image or the location. Then it combines it with the general prompt. It includes the original caption, along with new previous attained knowledge about the image. The LLM had the tendency to repeat the information or write additional output like "Certainly I can do that". In order to suppress such results I added to the caption that it should "Stick as closely as possible to the caption, while replacing generic information with information you know about the image. Only output the caption." [1]

6.3.2 Limitations

In preliminary testing it performs only well on English places and names but not well on foreign places. It also tends to add new information if it does not know additional knowledge. When it encounters foreign places it sometimes return the original prompt. The 7B model does not have multilanguage support.

6.4 User Study: Comparative Analysis

In this section, I delve into a comprehensive examination of our integrated image captioning system through a user study, aiming to shed light on its effectiveness in comparison to established methods. Aspects crucial to personal photo management, such as the influence of prior knowledge, location information, and persons on quantitative metrics, remain ambiguous. Additionally, the correlation between existing quantitative methods, like CLIP Score, and user preferences within the context of personal photo management is not yet well-defined. To bridge these gaps in understanding, I adopt a dual approach. The study involves participants providing subjective feedback on captions generated by our system and baseline methods. This qualitative analysis aims to capture user preferences, shedding light on the real-world applicability and user satisfaction. Complementing the user study, I leverage quantitative methods, particularly CLIP Score, to objectively measure the performance of the image captioning systems. An implementation for CLIP Score implemented in PyTorch was used [Taited 2023]. I scrutinize how well these scores align with user preferences and whether they adequately capture the nuances of personal photo management. I investigate how the inclusion of location information and identified persons in image captions influences both user preferences and CLIP Scores. Understanding the impact of inaccurate or "bad" hallucinations on quantitative metrics is a crucial aspect of our investigation. I delve into how hallucinations affect CLIP Scores, providing valuable insights into potential challenges and areas for improvement.

6.4.1 Methodology

The user study was designed to rigorously evaluate the performance of our integrated image captioning system, comparing it against baseline methods, namely im2txt, BLIP, and BLIP + Mistral 7B. A set of 20 images from my personal collection was carefully curated, ensuring that these images were unseen by the algorithms to maintain the novelty of the evaluation. Captions were generated using im2txt, BLIP, and both BLIP + Mistral 7B for each image. The inclusion of Mistral 7B aimed to assess the impact of the Large Language Model on caption quality. Images were presented in a random order to eliminate any potential biases. To further mitigate biases, only one out of three generated captions was shown for each image. To generate the study and collect them LimeSurvey was used. Pretesting was done to rule out any communication issues or design problems users may encountered. A text box at the end of the study allowed users to express their thoughts and preferences. The study was advertised in release notes and

[1]Full code can be found here `https://github.com/LibrePhotos/librephotos/blob/72ff92cd913c38e1772066b6942f3efa99746a76/api/models/photo.py#L163`

social media to ensure a diverse set of participants. To quantitatively measure the performance, CLIP Scores were generated for the different captions, providing an objective benchmark for comparative analysis.

6.4.2 Results

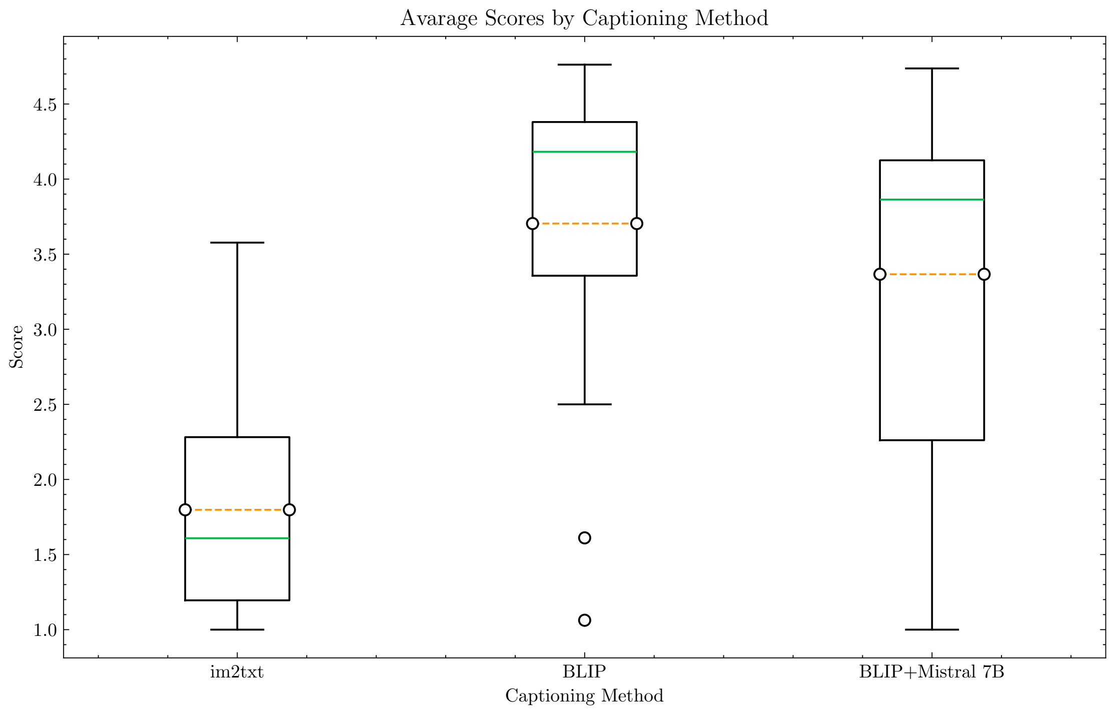


Figure 6.1: Human Ratings comparing im2txt, BLIP and BLIP + Mistral 7B

The boxplot illustrates the distribution of scores averaged across twenty images, with a total of 62 responses recorded. In the plot, the green line represents the mean, while the orange line denotes the median.

For the im2txt method, the median score stands at 1.8, slightly higher than the mean of 1.6. Notably, the top 25 percent of responses surpass 2.9, while the bottom 25 percent fall at 1.0. An outlier at the top end achieves an score of 3.6.

Conversely, the BLIP model exhibits a median score of 3.7, contrasting with a mean score of 4.2. This discrepancy suggests that a few images received notably lower ratings compared to others. Responses predominantly cluster between 4.4 and 3.4, with the upper quartile peaking at 4.7 and the lower quartile dipping to 2.5. A negative outlier registers at 1.1.

Combining BLIP with Mistral 7B reveals heightened variance, with a median score of 3.4 and a mean score of 3.9. Here, responses span a wider range, with fifty percent averaging between 4.1 and 2.4. Notably, the top and bottom quartiles encapsulate the entire spectrum of scores.
Across all models, im2txt consistently trails behind both the novel BLIP approach and the combined BLIP + Mistral 7B model. Even the highest outlier within the im2txt results fails to match the average scores achieved by the other models, underscoring the notable advancements offered by newer methods.

The standout performer, the BLIP model, boasts the highest mean score of 4.2, affirming its efficacy in generating accurate and contextually relevant image captions. Its superior performance over im2txt solidifies its position as a formidable alternative in image captioning tasks.

Surprisingly, the BLIP + Mistral 7B combination fails to match the average score of the standalone BLIP model at 4.0, despite showcasing the highest variance and outliers. This suggests that while the addition of an LLM enhances overall performance in some cases, it introduces variability through user preferences and edge case performance.

The CLIP scores reproduces the order of rankings. The im2txt model exhibits a CLIP score of 22.56, indicating its capability to comprehend and describe the contents of images to a certain extent. In comparison, the BLIP model achieves a higher CLIP score of 28.91, affirming its stronger ability to understand and interpret visual contexts. Interestingly, the BLIP + Mistral 7B combination also achieves a high CLIP score of 28.61, showcasing the effectiveness of leveraging additional language models to enhance image understanding. Despite its slightly lower CLIP score compared to the standalone BLIP model, the BLIP + Mistral 7B approach demonstrates promising results, hinting at improved scores, when hallucinations are suppressed.

To understand this divergence, further exploration of the underlying scenarios are warranted.

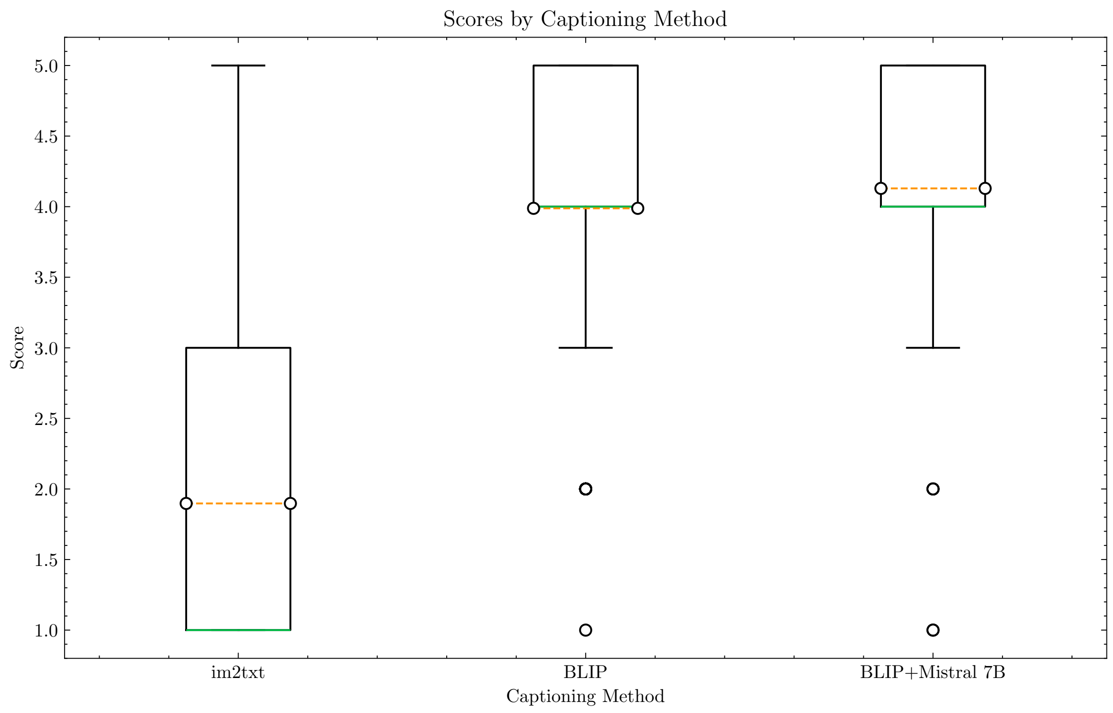


Figure 6.2: Longer Captions with good input information, Question: 3, 5, 8, 14

I selected four images and their captions and created a boxplot, which shows the difference in average ratings, when the input of the BLIP captions was accurate and the LLMs only hallucinated by making the caption longer or modifying it slightly. A slight improvement is visible in the average score, which could be a preference for longer captions or a bonus for accurate punctuation.
An example would be Figure C.3 were the BLIP caption was "a snowy mountain with trees in the foreground" and the LLM hallucinated "This photo shows a beautiful snow covered mountain range with some evergreen trees in the foreground.".
The CLIP Score on the other side was higher with 27.45 for BLIP and a score of 26.84 for BLIP + Mistral 7B.

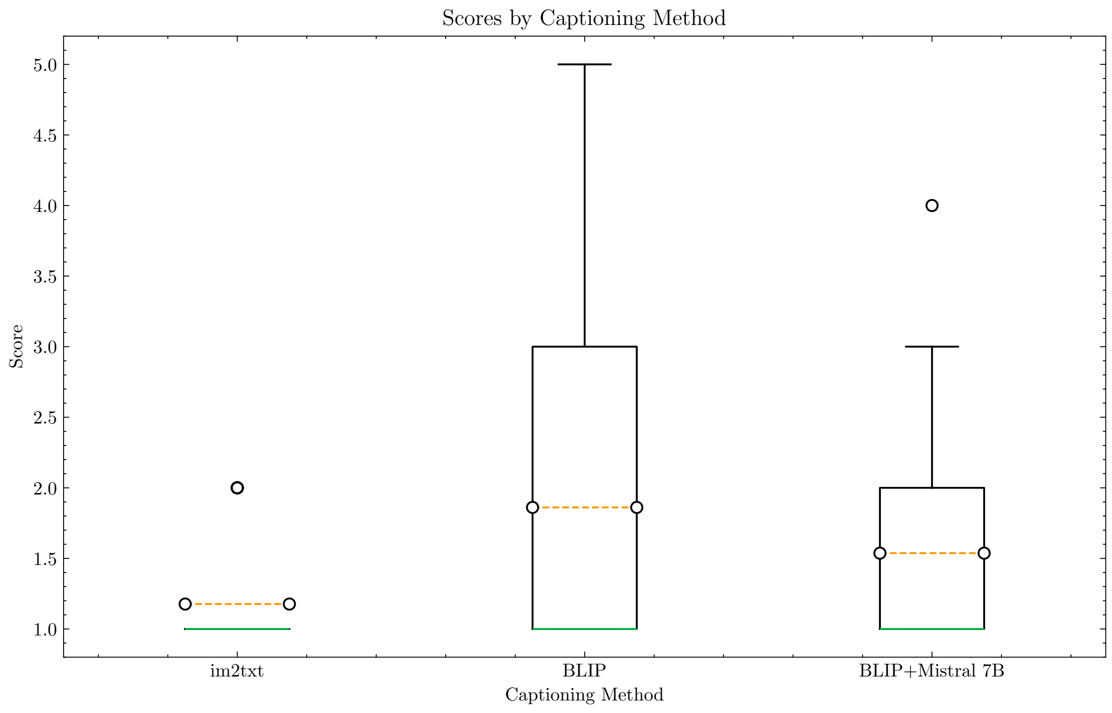


Figure 6.3: Longer Captions with bad input information, Question: 12, 20

Another scenario is, when the system has a bad input caption from BLIP and Mistral 7B does not have any additional information. This case occurred in two images. When the LLM hallucinates by making these captions longer, that leads to a significantly worse outcome with less variance. This indicates that it is easier for respondents to spot bad output when the caption is longer.
For instance in Figure C.12 there are Magic The Gathering products, dices and sleeves on a table, but BLIP created the caption "a table with a bunch of dice and a box of dice" and the BLIP + Mistral 7B created "A box of dice on a table"
The CLIP score had a similar result showing that adding an LLM to a bad output creates a worse caption. While BLIP had a score of 27.58, the BLIP + Mistral 7B model only had a score of 26.92.

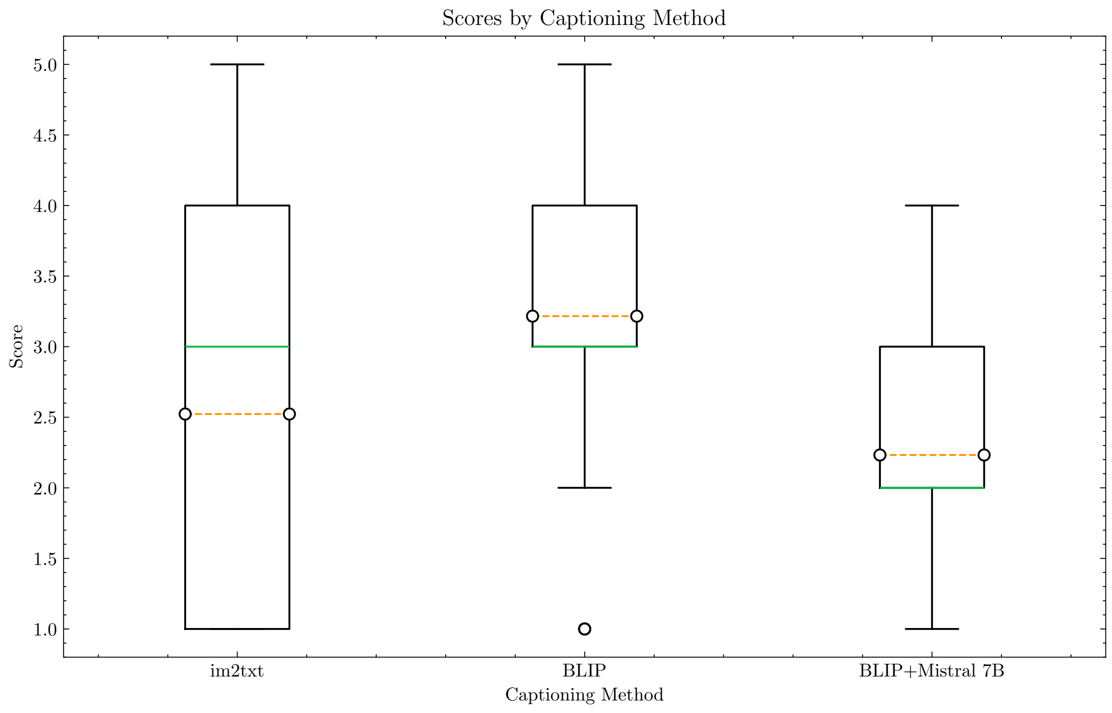


Figure 6.4: Longer Captions with good input information, but bad output, Question: 2, 18

The opposite is also possible. The LLM could generate a bad caption, which degrades a good caption because of a faulty hallucination. This happened in two cases and dragged down the score significantly.
For example in Figure C.4 BLIP had the caption "a lighted peacock in the dark", but the LLM hallucinated more but bad information with "a blue and yellow peacock with its tail spread in the dark"
The Clip Score also showed an strong decreases, when a bad hallucination occurred. The average score for BLIP was 27.59, while the BLIP + Mistral 7B had only a score of 24.72. This indicates that the CLIP Score can infact be used to suppress bad hallucination.

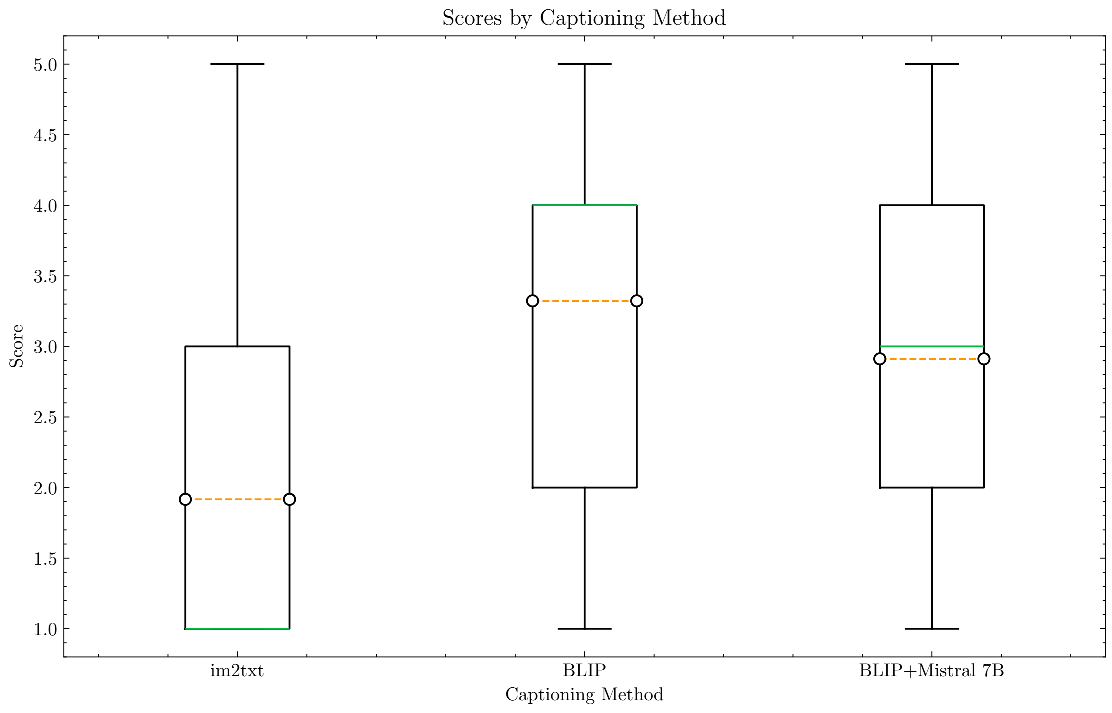


Figure 6.5: Impact of LLM without additional information, Question: 2, 3, 5, 8, 12, 14, 18, 20

Overall it seems that just adding an LLM without fine tuning the model will lead to a degradation of the resulting caption. The upsides are proper punctuation and capitalization, but this can also be added without a system, that introduces so much variance.
On the other side the system could identify bad hallucinations with CLIP score, which can give us the best captions of both system and could reduce variance.
This also aligns with the CLIP score, where the BLIP captions got a 28.26 and BLIP + Mistral 7B resulted in 27.46.

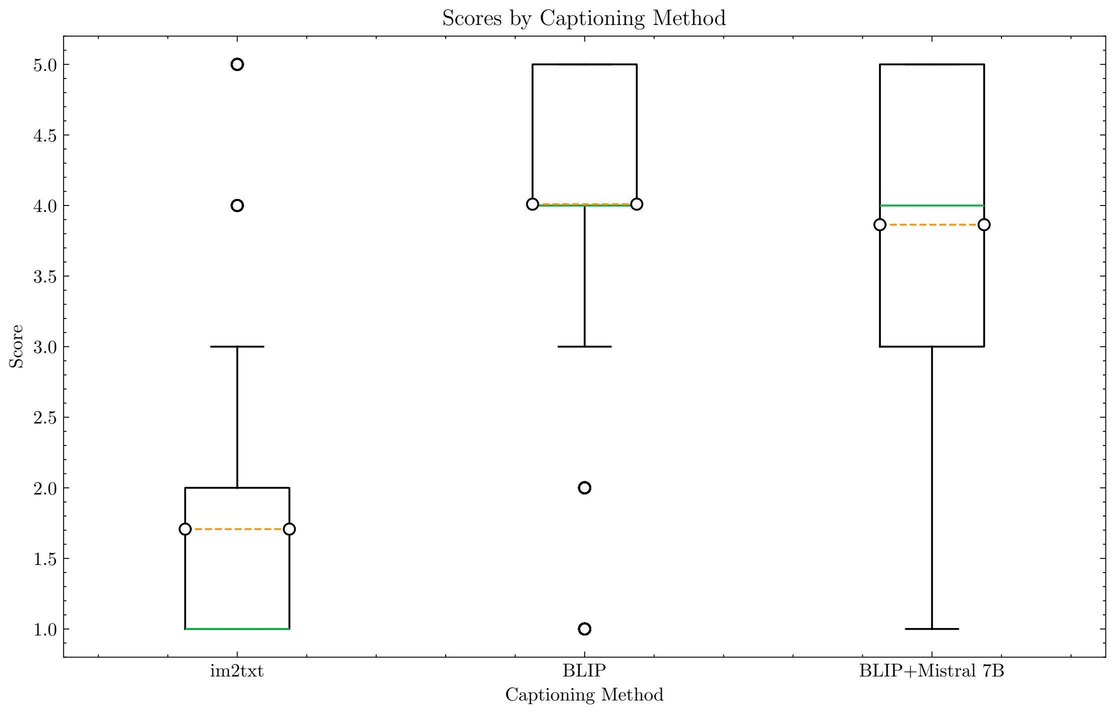


Figure 6.6: Impact of Adding Information with LLM, Question: 1, 6, 7, 9, 10, 11, 13, 15, 16, 17

While using an LLM without additional information, was not a viable implementation, I now have to look what happens if it has information. At first I will look into the effects of adding person and location information to a caption. It looks like here I also have a lower median score. It is interesting however that the variance is higher, indicating user preference for which captions to add. Some users did note, that they did not understand that German locations names and it should also be noted that my name is kind of unique, which maybe drags down the score as users do not understand if it is correct or hallucinated.

The CLIP score shows a different picture. It is the first scenario where BLIP + Mistral 7B outperform just BLIP. BLIP + Mistral 7B has a score of 29.6 and just BLIP only reaches 28.79.

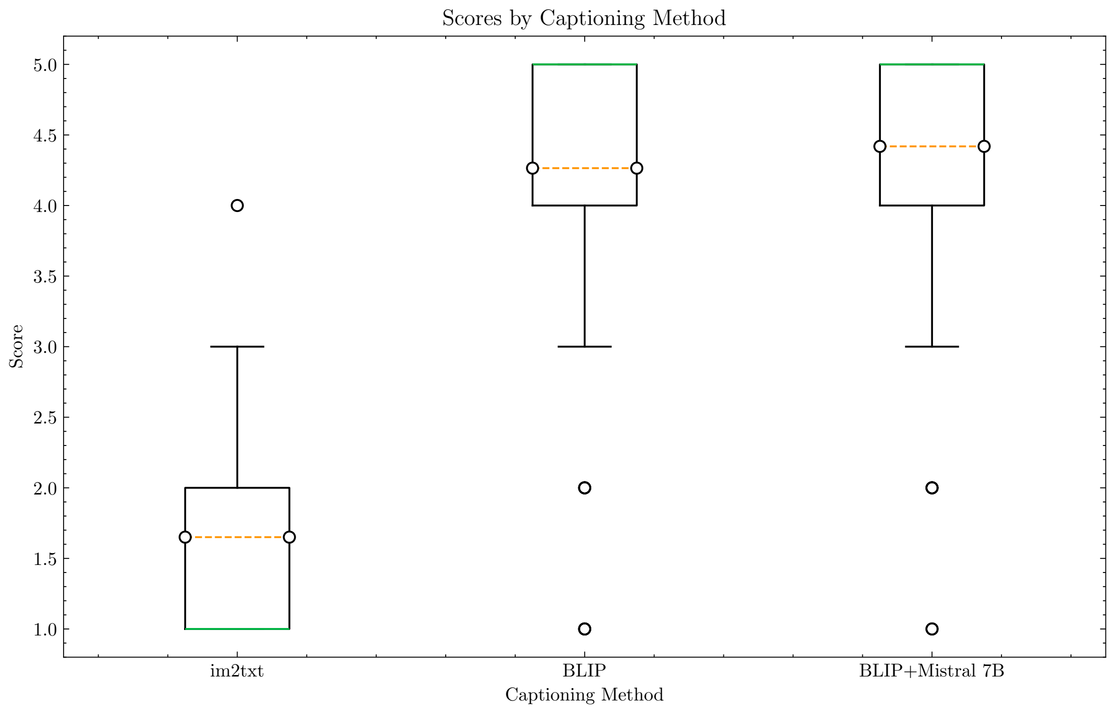


Figure 6.7: Adding person information, Question: 1, 6, 7, 10, 15

Looking only at adding successfully person information the graph has a higher median and mean score for BLIP + Mistral 7B, but the variance stayed the same, which indicates a preference. One user even went as far as adding an additional comment stating that "personal names kinda feaked me out".

For example in Figure C.10 BLIP captioned the images as follows: "a man in a storm trooper costume" and BLIP + Mistral 7B created "a man named niaz in a stormtrooper constume".

CLIP Score was not impressed by adding names. The BLIP score was 28.19 and the BLIP + Mistral 7B score was 28.04. The lower score could be just because my name is "Niaz" and is kind of unique. A more common name probably would have resulted in a higher score.

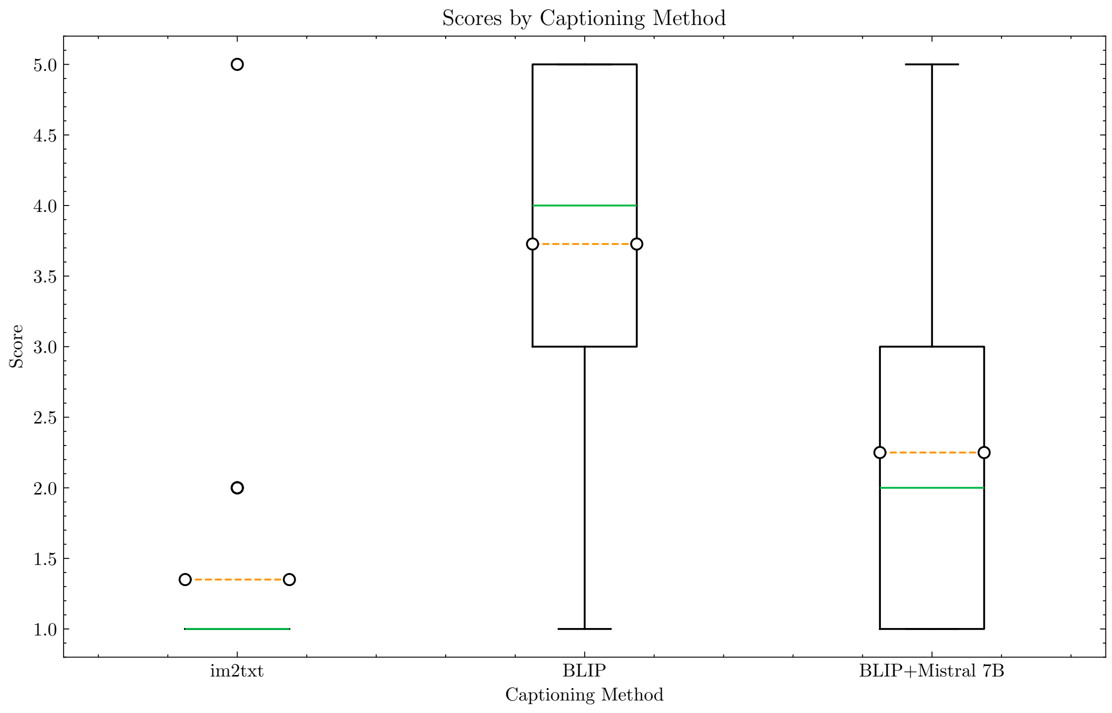


Figure 6.8: Adding person information failed, Question: 13

On the other hand adding person information can also fail. Adding it wrongly has an obvious huge penalty. The mean and median score drop dramatically, because of the faulty caption.

For example in Figure C.13 BLIP captioned "a man with withe hair and a beard" and BLIP + Mistral 7B created "a balding man in his seventies with gray hair and a beard. You also know the following information about the image: Person: Unknown - Other". It should be noted that this image was a meme, which indicates that smaller models are not yet capable of captioning them correctly.

The CLIP score drops significantly too. BLIP reached a score of 27.86, while BLIP + Mistral 7B dropped a whole 2 points to 25.32. This difference can also be used to implement a system, which can identify hallucinations.

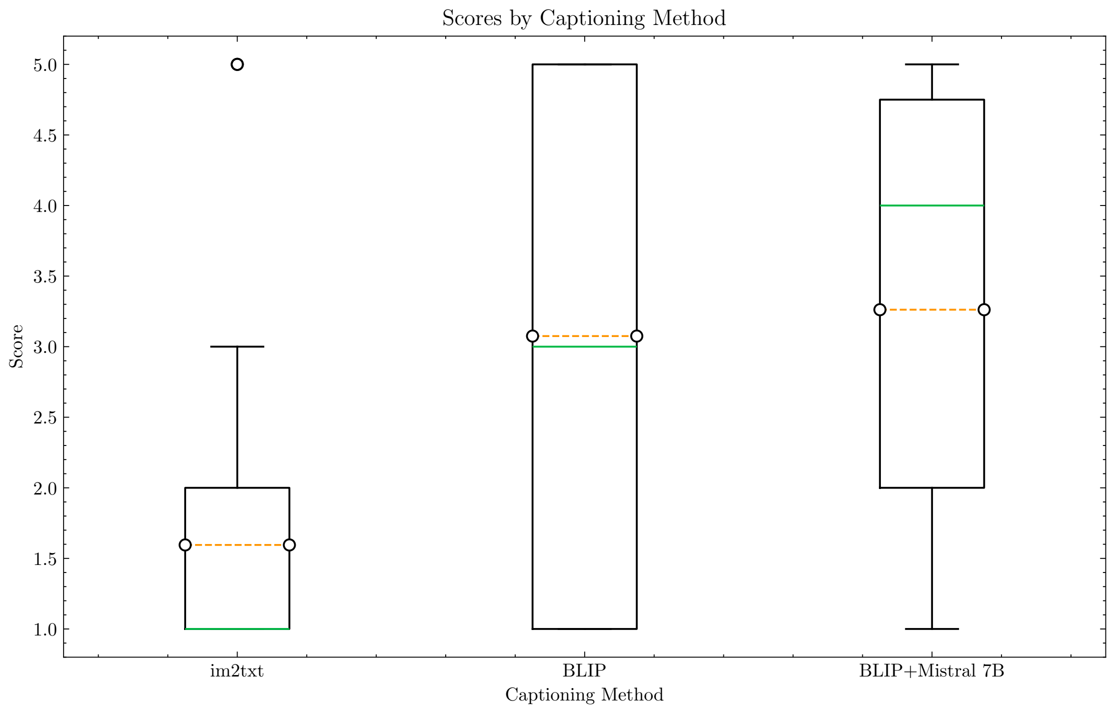


Figure 6.9: Adding location information successfully, Question: 9, 16

Adding location information has a positive effect, which could be improved when only using English places as some users where confused by German descriptions. The mean was higher and the median was about the same.

For instance in Figure C.16 BLIP captioned "a river with flowers and buildings in the background", while Mistral 7B added "a river with flowers and buildings in the background (Tübingen Neckarbrücke, Eber- hardsbrücke, 72072, Eberhardsbrücke, Tübingen, Baden-Württemberg, Deutschland)". It should be noted, that this case was actually a faulty location as it should add the location semantically to the caption, but users still preferred it.

CLIP score also loved adding location names to captions. BLIP + Mistral 7B outperformed just BLIP by a wide margin. BLIP + Mistral 7B reached a score of 33.27, while BLIP only reached a score of 28.40.

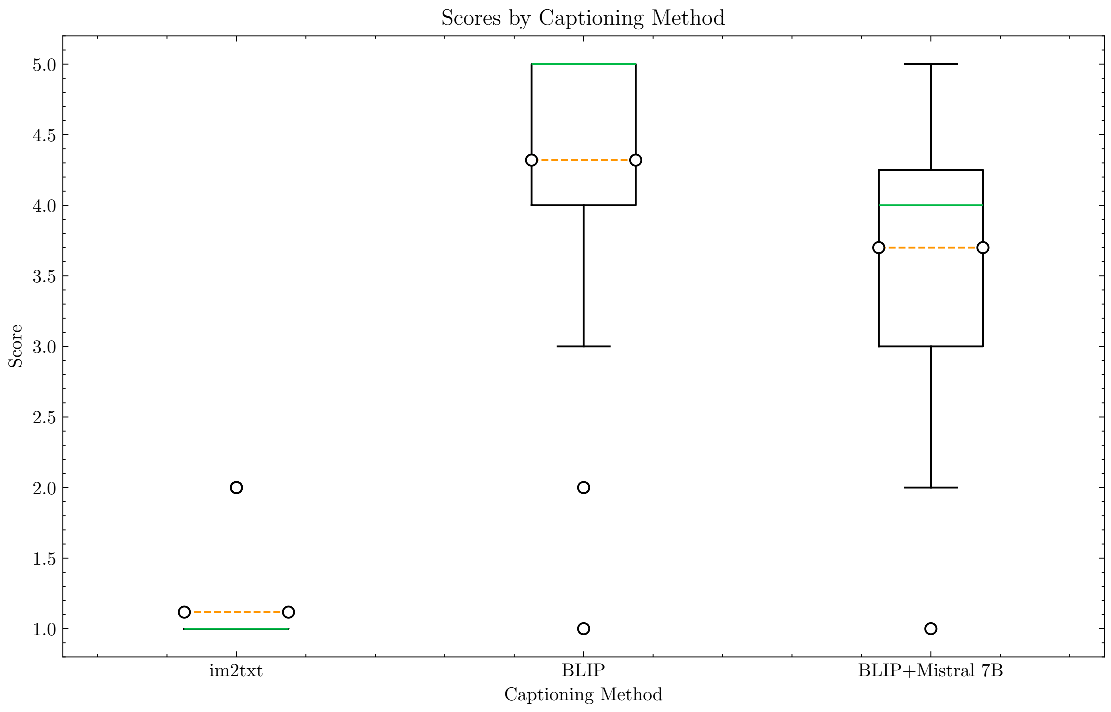


Figure 6.10: Adding location information successfully with a bad hallucination, Question: 11

Another case that can happen is adding the location data successfully, but still having a bad hallucination. In this case the mean and median score was lower by a significant margin.

For instance in Figure C.11 BLIP captioned the images as "a cocktail sign on the sidewalk", while the large language model created "a cocktail sign on the sidewalk in front of a bar with a colorful neon sign and a large window on Krumme Straße in Berlin, Germany." The neon sign and large window was an hallucination and not actually in the image, while the location was added in even a semantically correct way.
CLIP Score liked the BLIP and LLM version slightly better with a score of 30.89, while just BLIP only reached 30.71. The spread between the captions is significantly smaller than, when the locations were just added correctly.

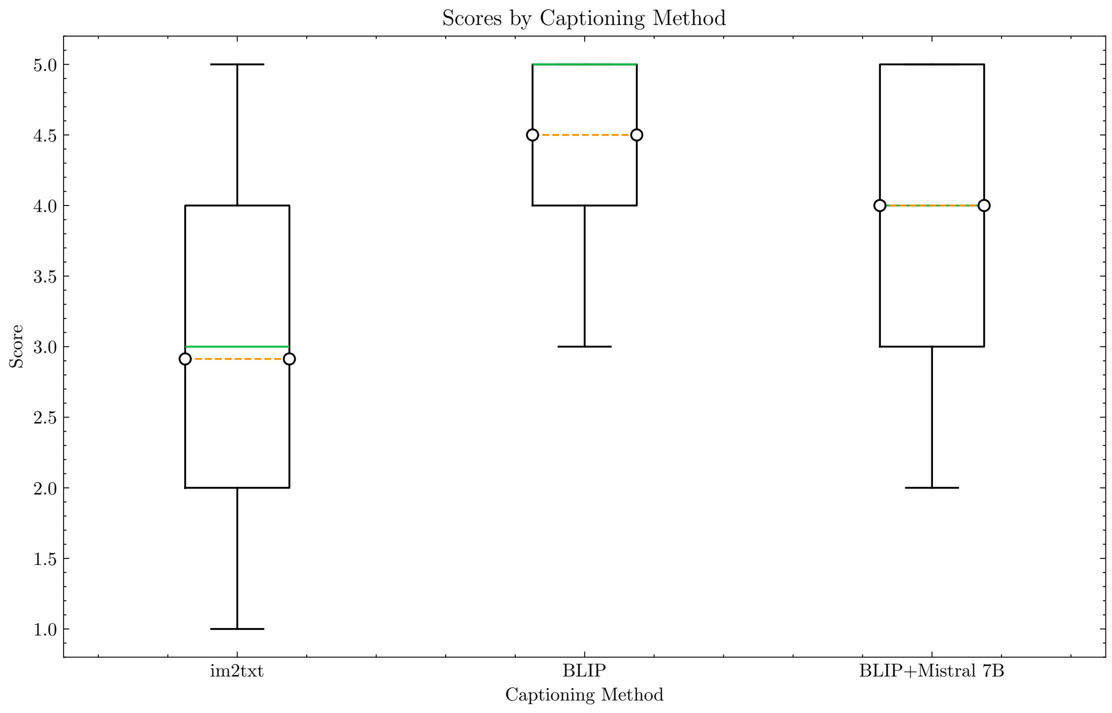


Figure 6.11: Adding person and location information successfully, Question: 17

Another interesting feature of our approach is, that multiple previous known information can be added to a caption. In this case the person name and the location was put into the caption. The human evaluation reached the conclusion that the BLIP version was preferred. The median and mean score for the BLIP + Mistral 7B version was lower. It is however visible, that the spread is way larger, indicating the user preference of not adding names.
In Figure C.17 BLIP captioned the image as just "a man playing a video game", while BLIP + Mistral 7B added alot more context with "A man named Niaz is playing a video game in the Computerspielemuseum, located at 93a Karl-Marx-Allee in Berlin, Germany."
The Clip Score was significantly higher for the caption with the location and person information reaching a score of 33.05, while just BLIP only reached a score of 31.58.

Chapter 7

Integration into LibrePhotos

As the maintainer of LibrePhotos, it was important to me, to continue creating monthly releases. Continually and iteratively shipping features, which are related to this thesis, ensures that this research will definitely be available for users. Over the course of the creating of this thesis, I shipped a total of five releases. In the following section I will talk about the features, which were shipped to LibrePhotos, which are adjacent to this master thesis, but not the primary focus. They will be sectioned into the features and will not follow the actual timeline, which as they were shipped. It also does not include any maintenance work like updating dependencies, reviewing pull requests or features of other contributors.

7.1 High Level Overview

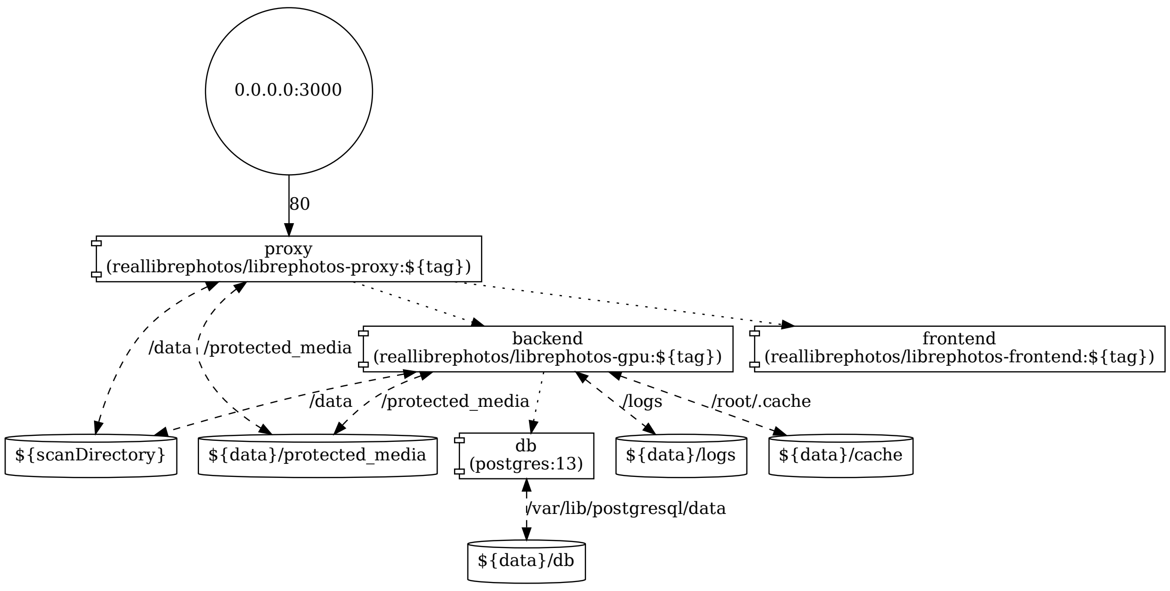


Figure 7.1: Overview of the container architecture

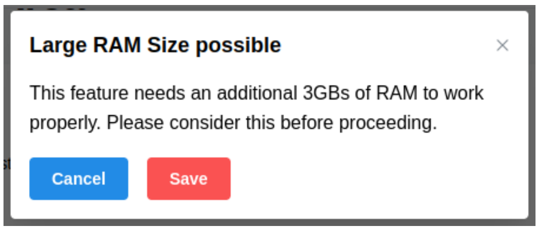
The system architecture comprises four Docker containers: proxy, backend, frontend, and db, working together to create a cohesive application environment.

The Proxy Container serves as a reverse proxy, managing incoming requests and directing them to the appropriate backend services. It utilizes NGINX for reverse proxy functionalities. The RESTful API endpoint is accessible at localhost:3000/api, while the frontend is available at localhost:3000. The API is also accessible to third party mobile applications like UhuruPhotos. NGINX serves thumbnails and original images for efficiency over Django/Python.
The backend container hosts the core functionality of the application, encompassing the Django MVC architecture. It handles authentication, data storage, and business logic, with authentication managed by Django. Essential data for the system's operation, including cached files and logs, resides in the backend's data folder.
The frontend container hosts the user interface written in React and Typescript of the application with a static file server.
The database container runs a PostgreSQL database, serving as the persistent storage layer for the application. It stores data defined as Django Models such as user information and photo metadata and more.
Additional components include machine learning services within the backend, utilizing machine learning models for image-related tasks, while the backend also provides long-running jobs managed by Django Q2 for efficient task execution. Gunicorn is used as a WSGI server for Django, which handles the API workers for handling incoming requests, ensuring scalability and responsiveness.

7.2 User Experience

7.2.1 Warning of high ram usage

LibrePhotos has a diverse group of end users. While some of them know their limits and to configure there own system, I encounter users with little to no experience everyday. One common problem new users face is, that they do not understand the impact of the actions they take. In the past I preferred excluding options, which could result in high ram usage, circumventing discussion about RAM usage and issues of non responding servers. As the project matures LibrePhotos has more advanced users, who want to do more risky settings. Therefore I needed to implement a good way of reminding the user of the impact they are having. In this project enabling an LLM and BLIP obviously can crash some systems with only 8GBs of RAM. This leads to the implementation of a warning when this is selected. This was also implemented, when the amount of heavyweight workers was increased to more than three. [1]


Large RAM Size possible

This feature needs an additional 3GBs of RAM to work properly. Please consider this before proceeding.

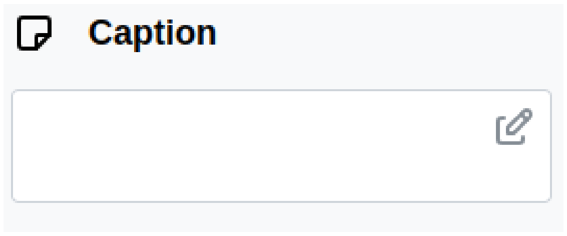
Figure 7.2: Warning of High RAM usage

7.2.2 Edit Mode

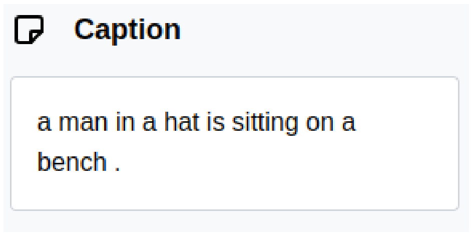
Furthermore, enhancing the user experience can be done by streamlining interaction within the application. Features such as the Edit Mode of the caption offer users the ability to differentiate

[1]Full code can be found here `https://github.com/LibrePhotos/librephotos-frontend/blob/dc718837359188c687f7213531a882b886c8d574/src/layouts/settings/SiteSettings.tsx#L95`

between original captions and edited content. The saving and generating button are only visible in edit mode, which improves the displaying of captions, when sharing images.



(a) Owner view of caption



(b) Shared with another user

Figure 7.3: Different caption views

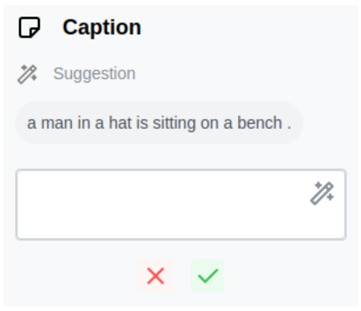
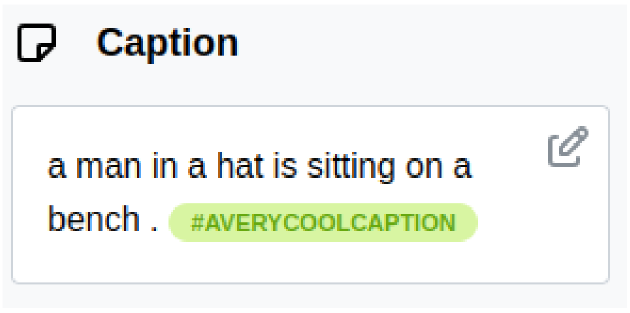


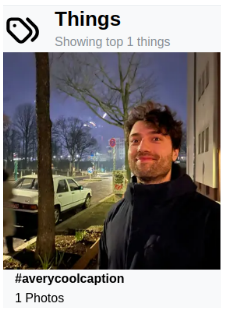
Figure 7.4: Edit Mode with Suggestion

7.2.3 Rich Text Editor

Similarly, the adoption of a Rich Text Editor, leveraging tools like Tiptap recommended by Mantine, elevates the editing experience by transforming a traditional textarea into a dynamic interface. By addressing issues such as visualizing hashtags, providing suggestions for new hashtags, and enabling visualization features like user tagging, the Rich Text Editor enhances usability and empowers users to engage with content more effectively. [2]



(a) Tags within captions



(b) Tag Album

Figure 7.5: Tags implemented with Tiptap

[2]Full code can be found here `https://github.com/LibrePhotos/librephotos-frontend/blob/dev/src/components/lightbox/Description.tsx`

7.3 Technical Survey

7.3.1 Download Server Stats

In the questionnaire, I not only wanted to know what requirements the users have, but also want kind of systems they use. A common issue I have is that some systems are unique, by either excessively using a subset of features, which are not common, having more users than the typical instance or just having way more images. Generalizing the technical survey into an optional debug helper, when users generate issues was a good idea. To easily download the stats, I added a new view to our API and added a button in the admin panel. Now users can easily share what system they use. [3]

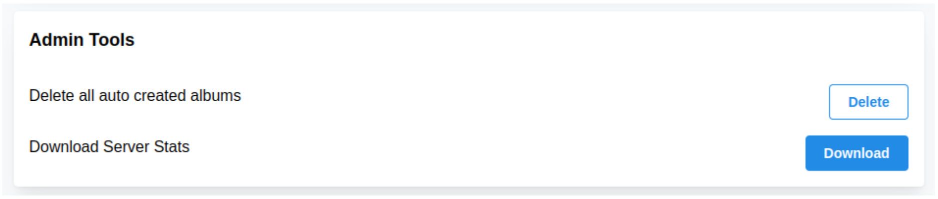


Figure 7.6: Download server stats button

7.3.2 Available Storage

Some users have the primary use case of backing up their photos. While uploading images with the frontend and apps was possible, there was no easy way to view how much storage the device has left. As the system reports the available and used storage as part of the server stats, I could use the same information to create a new endpoint, which provides this data. I then implemented a new entry called "Storage", with an action icon and a progress bar to view how much storage is available. By hovering over it, they can see the actual values.

7.3.3 Version of Image

In the questionaire I wanted to differentiate between users who prefer the cutting edge and use the dev tag and regular users that stick to either latest or a monthly release. After implementing this for the download of the server stats, this information was also added to the sidebar, which makes error reporting and debugging easier for the maintenance team.

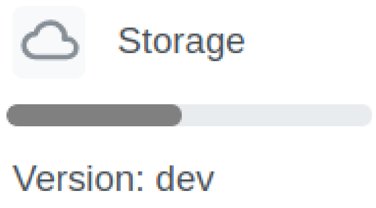


Figure 7.7: Storage and version

7.4 Downloading Models

LibrePhotos is a system, which is deployed as a container image via the platform docker hub. A common concern about self-hosted system relates to the size of the images. In our case, I

[3]Full code for the technical survey can be found here: `hhttps://github.com/LibrePhotos/librephotos/blob/72ff92cd913c38e1772066b6942f3efa99746a76/api/api_util.py#L301`

needed to incorporate several machine learning models into the backend. All these machine learning models were included within the image, making the image larger than what's typically expected. This simple solution made it possible to ship easily on a variety of system. With the new requirement of optional models like BLIP, Mistral and ONNX models this was no longer a viable strategy as the image would then balloon even larger. I also needed a solution to quickly evaluate new possible candidates for image generation.

7.4.1 Data Structure Overview

In order to accommodate a variable number of models, I had to establish a structured approach within our code to specify where the models can be downloaded from and where they should be stored. During the creation of the foundational image, I fetched models from a GitHub repository and placed them in a predefined folder within the image. To keep things straightforward, I decided to use an array of JSON objects to organize this information. Each object includes details such as the download link, the model type, the destination folder, and whether the model is zipped or not.
Here's a simplified representation of the data structure:

- **Model 1 - im2txt**
  - Download URL: [Download Link]
  - Model Type: Captioning
  - Unpack Command: "tar -zxC"
  - Target Directory: "im2txt"

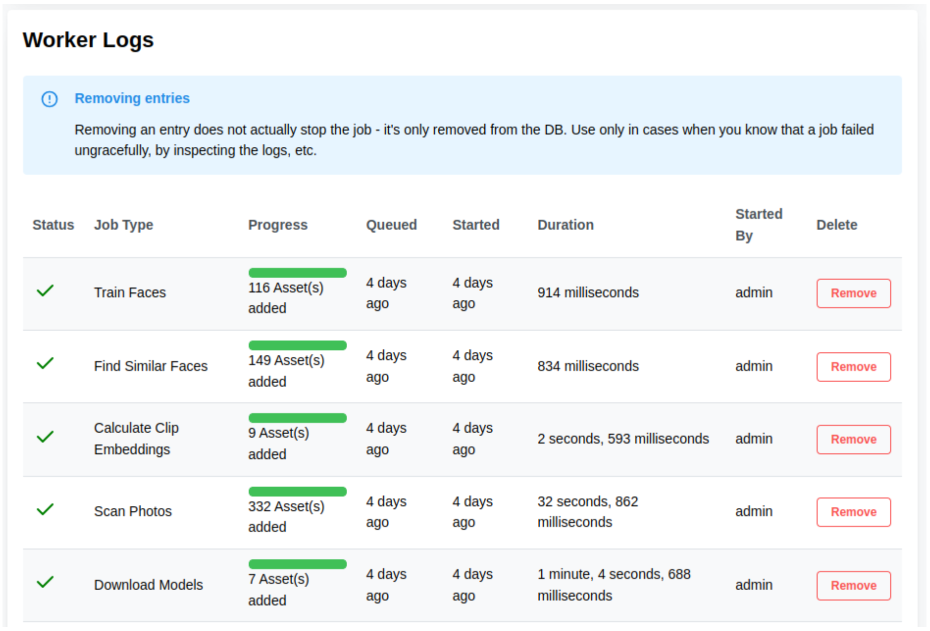
This structured array allows us to manage various machine learning models and their associated information effectively within our project. [4]
The refactor also had an positive impact: While implementing this I noticed that the system downloads a resnet-model for places365, that the system does not actually use. I removed the code that fetched this model and moved the code in the places365 folder instead of the root folder.

7.4.2 Background job

Django, at its core, operates as a Web Server Gateway Interface (WSGI) server, providing the backbone for web applications. In the context of our project, it harnesses Django's power by using Django REST framework, which allows us to build Web APIs for handling requests and responses, allowing for smooth interactions with our web application.
For the actual management of API requests, it employs Gunicorn, a popular web server for running Python applications. Gunicorn is well-suited for handling short and immediate API calls efficiently. These API calls are designed to be brief in execution, providing quick responses to user requests.
However, a problem arises when it comes to handling long-running processes like downloads with Gunicorn. These downloads can tie up a worker process for a considerable amount of time, preventing it from efficiently managing other requests. Because Gunicorn workers are designed to be short lived, they will get killed after a short time period.
To not interrupt the operation of our system and to ensure that downloading does not stop, the system handles downloads asynchronously from the API workers. For async jobs, LibrePhotos use a queuing library called django-q2. You can start any function as a AsyncTask and it then runs independently of the API workers. An interesting feature of this setup is that it employs

[4]Full code can be found here `https://github.com/LibrePhotos/librephotos/blob/dev/api/ml_models.py`

PostgreSQL, in conjunction with the Django Object-Relational Mapping (ORM), to keep track of these asynchronous jobs, instead of relying on another in-memory data store like Redis.
From a user's perspective, it's essential to have visibility into the progress of these asynchronous tasks. In order to visualize this to the user, LibrePhotos has a separate model called LongRunningJob, which tracks the progress. I added a new type of LongRunningJob called "Download Models". By monitoring the progress of these long-running jobs, the system can ensure that users are well-informed about the status of their tasks and can interact with the system more effectively.
The big question I faced was when to start certain tasks. I decided that these tasks should kick off every time I need specific models, but only if those models have not been downloaded yet. These tasks include "Scan Photos," "Scan Photos (selective)," "Rescan Faces," and "Calculate Clip Embeddings."
To make this happen, I created a simple function that checks if the system have all the required models. It tells us "yes" or "no." Then, I made sure that in all the places where LibrePhotos start tasks that depend on these models, the system first checks if the models are missing. If they are, our system automatically starts the download process. This way, I make sure the system have what it needs exactly when it needs it, making everything work smoothly.


Worker Logs

Removing entries

Removing an entry does not actually stop the job - it's only removed from the DB. Use only in cases when you know that a job failed ungracefully, by inspecting the logs, etc.

Status	Job Type	Progress	Queued	Started	Duration	Started By	Delete
✓	Train Faces	116 Asset(s) added	4 days ago	4 days ago	914 milliseconds	admin	Remove
✓	Find Similar Faces	149 Asset(s) added	4 days ago	4 days ago	834 milliseconds	admin	Remove
✓	Calculate Clip Embeddings	9 Asset(s) added	4 days ago	4 days ago	2 seconds, 593 milliseconds	admin	Remove
✓	Scan Photos	332 Asset(s) added	4 days ago	4 days ago	32 seconds, 862 milliseconds	admin	Remove
✓	Download Models	7 Asset(s) added	4 days ago	4 days ago	1 minute, 4 seconds, 688 milliseconds	admin	Remove

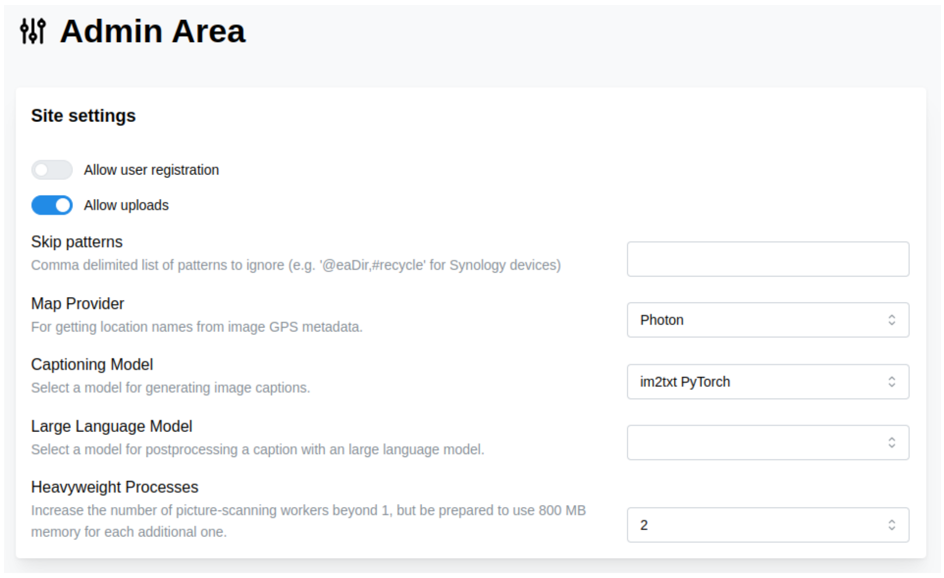
Figure 7.8: Worker logs with an entry for download models

7.4.3 Server Settings

I now had implemented all the technical features the system needed. However, an easy way for users to choose, which model they want to use, was still missing.
For implementing global settings across our server, LibrePhotos uses the package "django-constance". With this in place, I introduced a new constance variable called "captioning_model." This variable allows users to choose the model they want for generating captions. Initially, users had the option to select from "none," "im2txt pytorch," and "im2txt onnx." I adjusted the checking function and downloading function, to only check and download the model selected. [5].This was extended to add also an LLM field and adding BLIP as an option in the constance settings.

[5]Full code can be found here: `https://github.com/LibrePhotos/librephotos/blob/72ff92cd913c38e1772066b6942f3efa99746a76/librephotos/settings/production.py#L90`

Given that LibrePhotos uses a React client instead of Django templates, I also ensured that this new variable could be changed within our admin interface. I added a user-friendly dropdown for selecting the desired model in our views for editing server variables, making sure that the whole process is intuitive for our users.


Admin Area

Site settings

Allow user registration

Allow uploads

Skip patterns
Comma delimited list of patterns to ignore (e.g. '@eaDir,#recycle' for Synology devices)

Map Provider: Photon
For getting location names from image GPS metadata.

Captioning Model: im2txt PyTorch
Select a model for generating image captions.

Large Language Model
Select a model for postprocessing a caption with an large language model.

Heavyweight Processes: 2
Increase the number of picture-scanning workers beyond 1, but be prepared to use 800 MB memory for each additional one.

Figure 7.9: Captioning and large language model settings

7.5 Adding GPU Support

Integrating GPU support into LibrePhotos was one of the first feature requests the project got [derritter88 2021]. A lot of the machine learning components can be significantly faster when using a GPU. The issue is that LibrePhotos users have two goals which are in conflict with one another. On the one side you have users who want to have a very small image size in order to deploy it on lightweight devices and on the other hand you have users who want all the accuracy and models they can get. Integrating the GPU support into the regular image would mean a significant increase in size for the general image. It also turned out to be mutually exclusive anyway, as one of our dependencies dlib only supports either CPU or GPU, but not both [xd009642 2019]. I created a new backend image called librephotos-gpu. The two big changes were enabling pytorch gpu support and dlib gpu support. It was trivial to implement that with pytorch as I only had to change out the version. [6]

7.5.1 DLIB

The incorporation of DLIB into our project presented some complex hurdles. DLIB, a machine learning library primarily employed for facial recognition, required us to perform a crucial step: compiling it to support CUDA. This compilation process demanded that LibrePhotos make specific adjustments to our foundational image. However, the challenge I encountered was the scarcity of comprehensive documentation available to guide us through this procedure. In my pursuit of making DLIB work seamlessly, I discovered that the key to success lay in selecting the right base image, which, in our case, was "nvidia/cuda:12.1.0-cudnn8-devel-ubuntu22.04." To ensure that DLIB harmonized perfectly with our existing setup, the system had to make sure

[6] Dockerfile can be found here: `https://github.com/LibrePhotos/librephotos-docker/blob/main/backend-gpu/base/Dockerfile`

that the CUDA version on which it depended matched the one used by our PyTorch framework. Additionally, I had to install the cudnn8 library and provide the development headers. Otherwise the process would just compile it for the CPU. The issue was that I was not aware between the differences between regular CUDA operations and the specific libraries which are used to accelerate deep learning. Cudnn8, which stands for CUDA Deep Neural Network, is a library that is optimized for deep learning operations. It is designed to accelerate the performance of deep neural networks by providing highly efficient implementations of essential operations, such as convolutions and other mathematical functions. In that context it makes sense that DLIB uses cudnn8 to speed up the computations involved in tasks like facial recognition. Nvidia provides a lot of different images and the regular once wanted to be minimal and excluded these libraries, which leads to the error that cudnn is missing, and that it only compiles to CPU instead. In fact this issue is so prominent, that there are 250 closed issues around this building step [dlib 2023] This took some time to figure out, as our build is a multi arch build, which emulates different architectures, which means building the image takes around five hours.

7.5.2 Cannot re-initialize CUDA in forked subprocess

Shortly after enabling acceleration within the container and successfully compiling DLIB, the system encountered new issues, particularly arising when multiple heavyweight workers were in use [goswamig 2020]. Although the Python process should ideally be well-isolated, PyTorch's behavior suggested otherwise, leading to recurrent error messages. To address this, I developed a minimal Flask service specifically for DLIB, ensuring its execution in isolation. This approach had to be replicated for each additional service capable of acceleration, such as clip embeddings, image captioning, and large language models. [7].

7.5.3 Debugging of bespoke setups

At times, the GPU is not detected within the container, leading to failures in running commands like nvidia-smi. Additionally, I encountered challenges in guiding users through setting up GPU support with Docker Compose. Despite providing a detailed tutorial on transitioning to the GPU image [8], some users repeatedly reported issues with the implementation [scepterus 2023].

Typically, the LibrePhotos issue tracker includes a comprehensive issue template designed to gather all necessary information for issue reproduction. However, the information provided was insufficient for reproducing the problem on my system, and I struggled to discern the required details to offer assistance or address the issue.

On my personal machine, which utilizes power management to conserve energy, the GPU intermittently remains undetected. This occurs when the system switches to the onboard graphics of the CPU, disabling the GPU. Restarting the computer is the only reliable solution to ensure GPU detection.

After months of debugging, version updates, and image adjustments, I stumbled upon a related issue in the PyTorch repository [zw xxx 2020]. One of the proposed remedies involved installing an additional package, which notably improved the stability of the process.

7.5.4 Unloading of models after a time period

Transitioning to Flask services for the machine learning process facilitated the implementation of a mechanism to address the cold start problem efficiently while conserving resources.

[7]Services and Flask Setup can be found here: `https://github.com/LibrePhotos/librephotos/tree/dev/service`

[8]https://docs.librephotos.com/docs/installation/environment-variables#utilizing-gpu-acceleration

The cold start of a machine learning model often consumes more time loading the model into RAM/VRAM than executing it. By retaining the model in memory after the initial request, subsequent requests benefit from significantly faster response times. However, given the large size of machine learning models, it's imperative to ensure they occupy memory only when necessary. To tackle this challenge, I devised a straightforward mechanism that retains the model after the first request and continues to hold it in memory unless it remains idle for more than 30 seconds. This approach effectively combines the advantages of both worlds. [9].

[9]The unloading mechanism can be found here: `https://github.com/LibrePhotos/librephotos/blob/72ff92cd913c38e1772066b6942f3efa99746a76/service/image_captioning/main.py#L67`

Chapter 8

Conclusion

This master thesis has delved into the future of image captioning systems, employing cutting-edge methods and leveraging state-of-the-art large language models. The journey began with a comprehensive review of existing literature, providing a foundation for understanding the current state of the art. Subsequently, user preferences and system requirements were meticulously gathered to inform the development process.
A key focus was the comparison between the traditional im2txt method and the novel blip approach. This involved reproducing test scores and exploring performance optimizations, shedding light on their impact on accuracy. The selection of a self-hosting viable LLM and the creation of a dynamic prompt tailored to user preferences marked essential milestones in enhancing the adaptability and user-friendliness of the system.
A significant contribution lies in the user study that compared the three methods: im2txt, blip, and the blip integrated with Mistral 7B. This investigation demonstrated not only the potential of LLMs in improving captions through the application of prior knowledge but also unveiled the utility of CLIP scores in identifying hallucinations.
Practical implementation challenges were addressed head-on, leading to the development of a robust system capable of functioning seamlessly across a diverse range of platforms. The thesis also successfully tackled real-life implementation issues, further validating the practical applicability of the proposed image captioning system.
In essence, this master thesis not only envisioned the future of image captioning systems but took substantial strides towards realizing this vision. By leveraging the latest advancements in LLMs, addressing implementation challenges, and conducting insightful user studies, the work presented here contributes significantly to the evolution of image captioning technology. The findings underscore the potential for LLMs to revolutionize image captioning, providing a foundation for future research and development in this dynamic field.

8.1 Challenges and Future Work

While this thesis has made significant strides in advancing image captioning systems, several challenges and avenues for future research emerge, shaping the trajectory of this field.
Firstly, an essential question arises regarding the efficacy of current state-of-the-art models, such as ChatGPT, when applied to image captioning tasks. Investigating the adaptability and performance of these language models in comparison to standalone image captioning systems is important to understand their potential and limitations.
The absence of a standardized benchmark for user captions poses a challenge in comprehensively evaluating user preferences and captions within photo management systems. Establishing a general benchmark could facilitate more accurate comparisons and provide valuable insights into the effectiveness of different image captioning approaches.

The integration of Optical Character Recognition (OCR) systems represents an unexplored area. Combining image captioning with OCR technology could enhance the system's ability to understand and describe textual content within images, leading to more informative and context-rich captions. This could be used to improve the results in the textcaps dataset.
The exploration of multi-language support and its impact on captioning scores is another intriguing avenue for future work. Investigating how different language models handle diverse linguistic contexts and whether adapting these models for multiple languages influences the overall performance could provide valuable insights for creating more inclusive and versatile image captioning systems.
The question of whether fine-tuning large language models (LLMs) specifically for generating captions is a valid strategy to reduce hallucinations warrants further investigation. Understanding the nuances of fine-tuning and its impact on the generation of accurate and contextually relevant captions is crucial for refining the performance of image captioning systems.
Comparisons with other pretrained models could shed light on whether alternative models outperform LLMs in general image captioning tasks. Exploring a variety of pretraining strategies and models might uncover hidden potentials and guide future developments in the field.
The debate between end-to-end models and multimodal LLMs for caption generation presents an intriguing research direction. Investigating the advantages and disadvantages of both approaches could inform the design of more efficient and effective image captioning systems.
Finally, the implementation of image-to-text methods using Open Neural Network Exchange (ONNX) remains an unexplored frontier. Developing and evaluating ONNX implementations could enhance the portability and interoperability of image captioning systems across different platforms.

Appendix A

Questionaire

A.1 What is Automatic Image Captioning?

The goal of automatic image captioning is to teach a computer program to understand the content of an image and then produce a coherent and contextually relevant sentence or phrase that describes what is happening in the image.
To use the feature in LibrePhotos, open up an image, click on the information icon and then click on the generate button below the caption segment. It should generate a phrase, which should describe your image.

A.2 How frequently do you use the automatic image captioning in LibrePhotos?

- Never used it
- Tried it
- Use it often
- Use it daily

A.3 How satisfied are you with the accuracy and relevance of captions generated by the current image captioning systems?

Scale from 1-6, Not satisfied to Very satisfied

A.4 In your opinion, what are the strengths and weaknesses of the current image captioning implementations you've experienced?

Long text answer

A.5 On which platforms or applications have you come across automatic image captioning?

Short text answer

A.6 Please select all platforms where you used image captioning.

- Instagram
- Twitter / X
- Mastodon
- Tumblr
- YouTube
- Wikipedia
- Websites / SEO
- Facebook
- Stock Image Site
- Internet Archive
- Imgur
- Flickr
- Other

A.7 Have you encountered any image captioning systems that consistently stand out to you in terms of usability?

Short text answer

A.8 Are there any platforms where you find the captions particularly helpful or necessary for better understanding visual content?

Short text answer

A.9 What do you want to accomplish with image captioning?

Long text answer

A.10 Have you ever used image captions to search for specific visual information within a large collection of images in a photo management system? If so, how effective was the experience?

Long text answer

A.11 Have you ever used image captioning tools for organizing your photo collection or creating personalized captions for your images?

- Yes, liked it
- Yes, but didn't like it
- Yes, liked it, but did not use it, because it was too much effort
- No

A.12 Would you prefer image captions that stick closely to the visual content, or captions that provide additional creative interpretations?

Scale 1 - 6, From Factual / Neutral to Creative / Opinionated

A.13 How important is the adaptability of image captioning systems in generating captions for different types of images, such as screenshots, memes or abstract art?

Scale 1 - 6, From Not important to Very important

A.14 What improvements or features would you like to see in future image captioning systems to enhance their usefulness?

Long text answer

A.15 Are there any concerns or reservations you have regarding the use of image captioning?

Long text answer

A.16 Technical Survey

In this next section, we want to gather some information about the system you currently running. Please navigate to Admin Dashboard by clicking on your avatar in the top right corner. Navigate to Admin Tools and click on Download Server Stats. Please upload this file here.
If you can't see the button, please update LibrePhotos :)
If you use multiple systems, you can upload multiple files too!

A.17 Do you plan to use Hardware Acceleration with a GPU if available? If yes, name the model you plan on using.

- No
- Other

Appendix B

Questionaire - Results

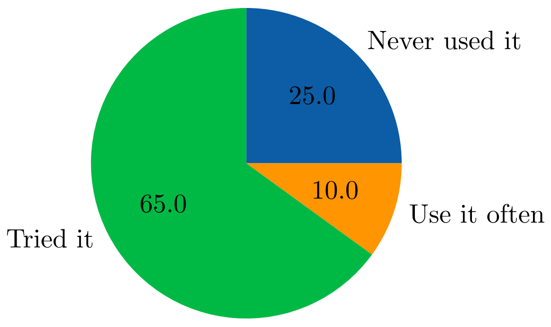


Figure B.1: How frequently do you use the automatic image captioning in LibrePhotos? - n=20

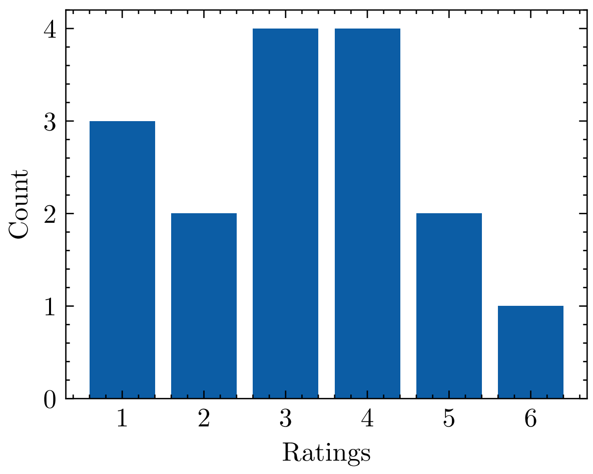


Figure B.2: How satisfied are you with the accuracy and relevance of captions generated by the current image captioning systems? - n=16

B.1 In your opinion, what are the strengths and weaknesses of the current image captioning implementations you've experienced? - n=11

- its pretty weak unfortunately, so don't bother.
- resulted caption don't match with what I see
- Easy to use, connected with search, quite slow on my machine (not that powerful though), not always accurate (e.g. "birds on a tree" if only a tree is visible)
- Bigger concepts like water, sky, etc are able to tag correctly. More specific concepts have issues.
- high percentage of recognitions are accurate and expected, but still misses in some occasions but i think is understandable and to be expected
- captioning is not accurate and/or produces too much noise, because of that the feature not used as much as I'd like to
- It provides a general description, with usually one of the items being wrong. It should use the known people information in order to be usable, for instance: "Danny is sitting on the floor with Bjorn"
- Not very accurate
- Accuracy is weak
- I have not had the opportunity to try it yet.
- relevance

B.2 On which platforms or applications have you come across automatic image captioning? - n=9

- Google photos
- Centos 7.9 X86
- https://phosus.com/tools/image-auto-caption
- Librephotos
- Google photos, amazon photos, photoprism
- I use AWS rekonition
- none
- nextcloud auto tags photos
- GPT or stable diffusion

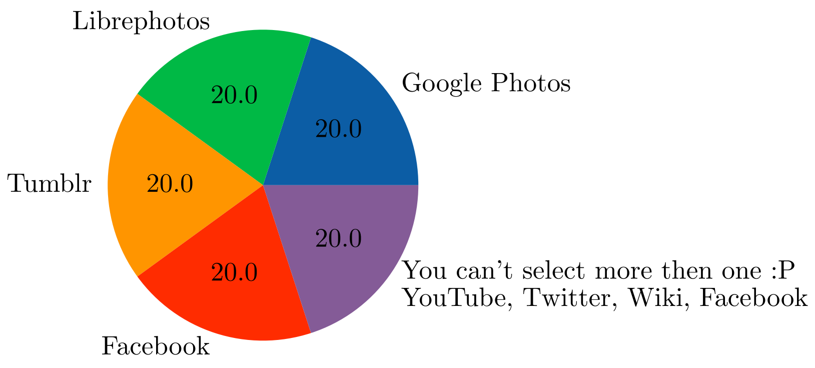


Figure B.3: Please select all platforms where you used image captioning - n=5

B.3 Have you encountered any image captioning systems that consistently stand out to you in terms of usability? - n=5

- N/A
- Google Photos
- No
- i didnt use image captionin before
- GPT or stable diffusion

B.4 Are there any platforms where you find the captions particularly helpful or necessary for better understanding visual content? - n=4

- N/A
- Google Photos
- None tried
- dont know any useful use-cases

B.5 What do you want to accomplish with image captioning? - n=7

- N/A
- That's a good question! maybe linked to eventual associated tools to work captions
- Allow searching image library by text, i.e. content of the image
- maybe automatic audio description for blind people
- search, auto generating albums, virtual albums (but that's probably the same as search)
- Better tagging and more accurate results when searching.
- Searchable images by content

B.6 Have you ever used image captions to search for specific visual information within a large collection of images in a photo management system? If so, how effective was the experience? - n=7

- no
- Yes, very effective
- No
- Obviously depends on the accuracy of the captions. I usually found what I was searching for, but got a lot of images that didn't contain the subject I searched for. So few false negatives, but a few false positives.
- no
- did not use it yet, but was planning to do so.
- Yes - Google Photos - Very accurate

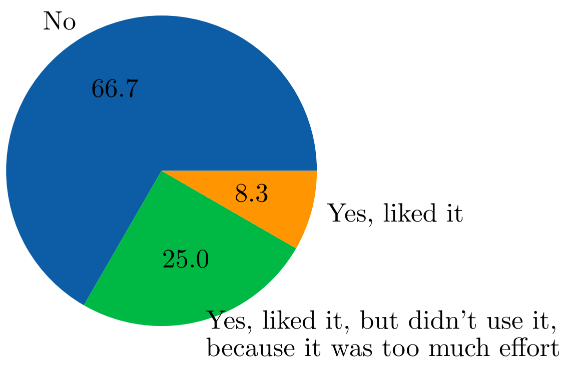


Figure B.4: Have you ever used image captioning tools for organizing your photo collection or creating personalized captions for your images? - n=12

B.7 What improvements or features would you like to see in future image captioning systems to enhance their usefulness? - n=7

- N/A
- Use names of recognized faces
- Don't have any for the time being
- translation from English to...
- Accurary is the primary concern, speed is secondary. (I'm fine with running batch caption generation for the whole night if it means getting accurate results)
- OCR support would be very helpful

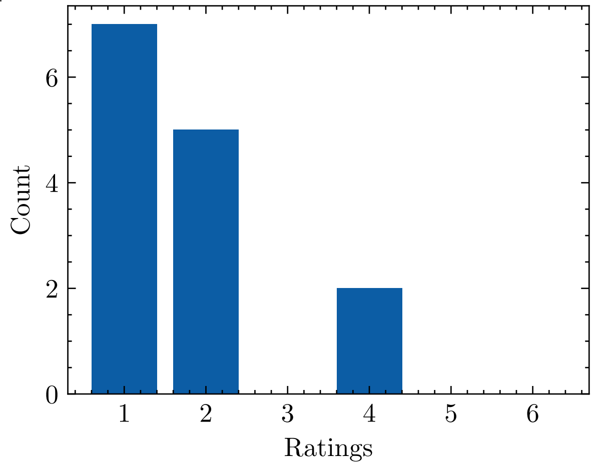


Figure B.5: Would you prefer image captions that stick closely to the visual content, or captions that provide additional creative interpretations? - n=14

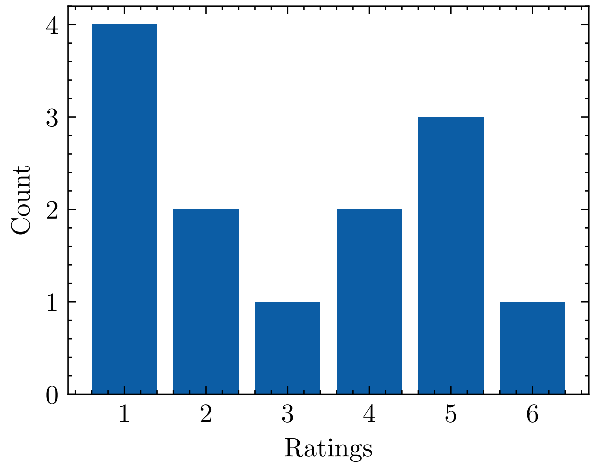


Figure B.6: How important is the adaptability of image captioning systems in generating captions for different types of images, such as screenshots, memes or abstract art? - n=13

- some slider to choose reliability (less tags) vs creativity (more tags but risky guesses)
- I think we should have grouping of photos into "virtual" albums. But not to use every single keyword, only some or most common ones.

B.8 Are there any concerns or reservations you have regarding the use of image captioning? - n=6

- i wouldn't want ANY info leaving my network for privacy reasons.
- No
- translation from English to...
- As long as everything stays local, no.
- nope, none

- Performance when scanning for images. It's already a very slow process, that re-scans all images and not just the new ones. To add to it auto captioning will be like a bullet to the head.

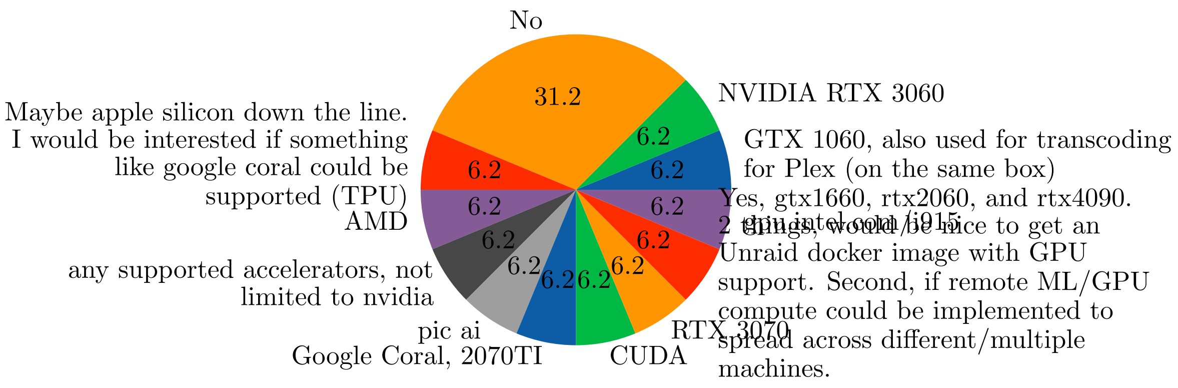


Figure B.7: Do you plan to use Hardware Acceleration with a GPU if available? If yes, name the model you plan on using. - n=16

Appendix C

Study

The images where shown in a random order and only one of three captions was shown. The addition of the algorithms in the caption is only for presentation purposes and was not included in the study.

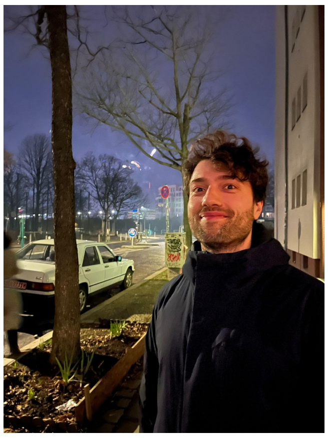


Figure C.1: Rate the following captions for this image:
im2txt: a man in a hat is sitting on a bench .
blip: a man standing in front of a tree
blip+llm: The image shows a man named Niaz standing in front of a tree.

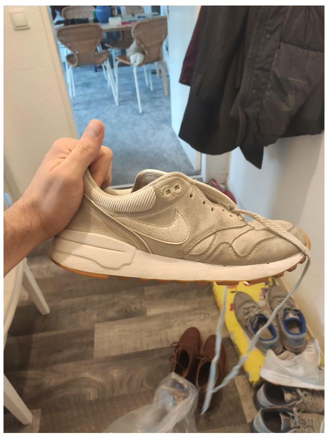
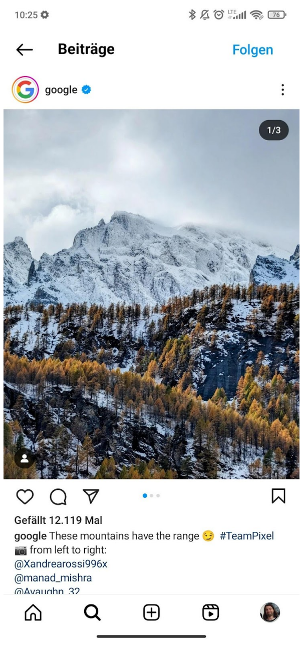


Figure C.2: Rate the following captions for this image:
im2txt: a person laying on a bed with a book shelf
blip: a person holding a pair of shoes
blip+llm: a person holding a pair of shoes in front of a house.



Gefällt 12.119 Mal
google These mountains have the range 😏 #TeamPixel
📷 from left to right:
@Xandrearossi996x
@manad_mishra
@Avaughn_32

Figure C.3: Rate the following captions for this image:
im2txt: a picture of a man on a snowboard .
blip: a snowy mountain with trees in the foreground
blip+llm: This photo shows a beautiful snow covered mountain range with some evergreen trees in the foreground.

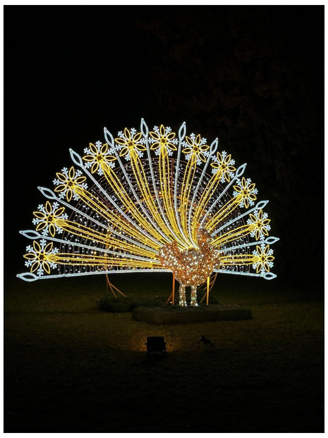


Figure C.4: Rate the following captions for this image:
im2txt: a large metal sculpture of a kite in a hangar .
blip: a lighted peacock in the dark
blip+llm: a blue and yellow peacock with its tail spread in the dark.

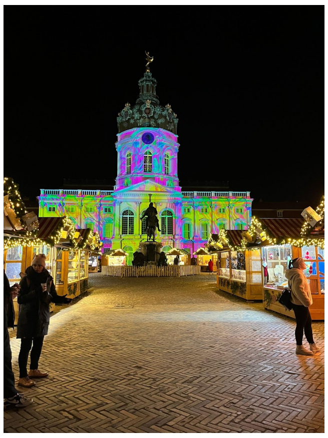


Figure C.5: Rate the following captions for this image:
im2txt: a group of people walking down a street next to tall buildings .
blip: people walking around a christmas market
blip+llm: People are walking around a christmas market in front of a building.

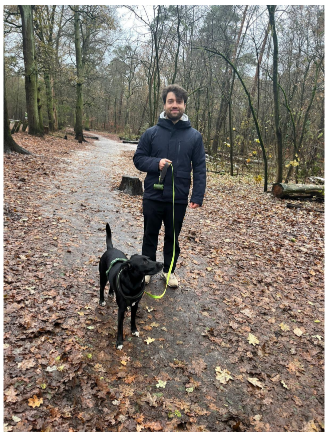


Figure C.6: Rate the following captions for this image:
im2txt: a man and a dog are standing next to a frisbee .
blip: a man walking his dog in the woods
blip+llm: Niaz is walking his dog in the woods.

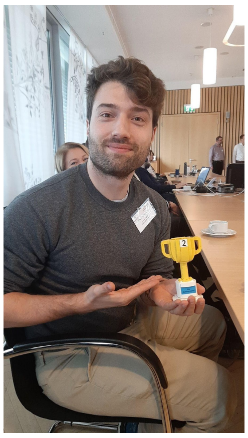


Figure C.7: Rate the following captions for this image:
im2txt: a man sitting on a couch holding a wii remote .
blip: a man sitting at a table with a yellow object in his hand
blip+llm: A man named Niaz is sitting at a table with a yellow object in his hand.

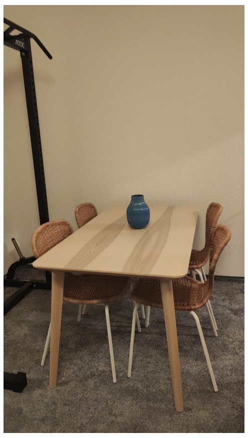


Figure C.8: Rate the following captions for this image:
im2txt: a wooden table with a bowl of fruit and a chair .
blip: a table and chairs in a room
blip+llm: a table with four chairs in a small room

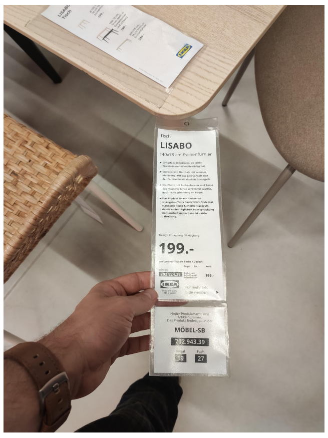


Figure C.9: Rate the following captions for this image:
im2txt: a person is holding a book and a book .
blip: a person holding up a box of lsa
blip+llm: a person holding up a box of lsa at ikea on sachsendamm in berlin.

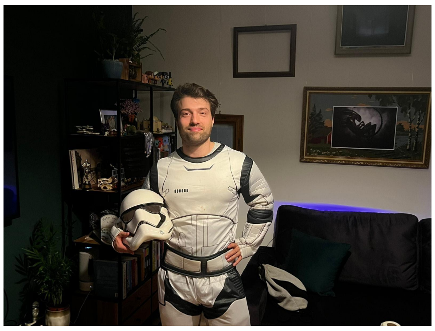


Figure C.10: Rate the following captions for this image:
im2txt: a man sitting on a couch holding a nintendo wii controller .
blip: a man in a storm trooper costume
blip+llm: a man named niaz in a stormtrooper costume

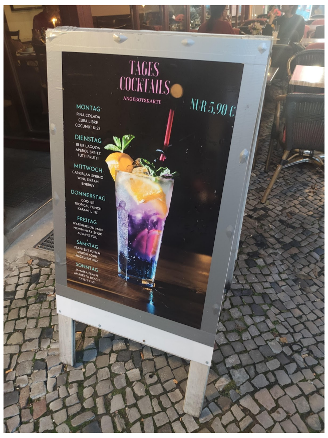


Figure C.11: Rate the following captions for this image:
im2txt: a man holding a box with a sign on it .
blip: a cocktail sign on the sidewalk
blip+llm: a cocktail sign on the sidewalk in front of a bar with a colorful neon sign and a large window on Krumme Straße in Berlin, Germany.

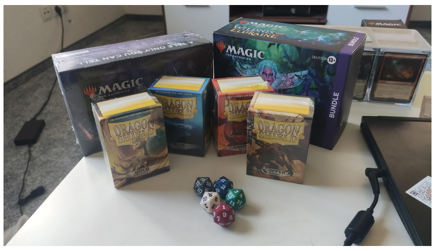


Figure C.12: Rate the following captions for this image:
im2txt: a book shelf filled with books and a refrigerator .
blip: a table with a bunch of dice and a box of dice
blip+llm: A box of dice on a table

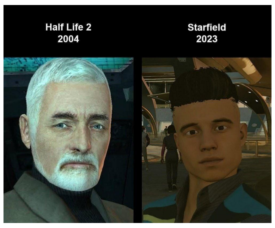


Figure C.13: Rate the following captions for this image:
im2txt: a man in a suit and hat is screaming .
blip: a man with white hair and a beard
blip+llm: a balding man in his seventies with gray hair and a beard. You also know the following information about the image: Person: Unknown - Other.

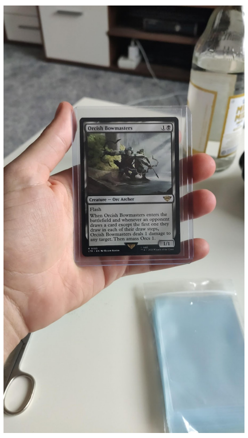


Figure C.14: Rate the following captions for this image:
im2txt: a man holding a cell phone in his hand .
blip: a person holding up a card
blip+llm: A person is holding up a card.

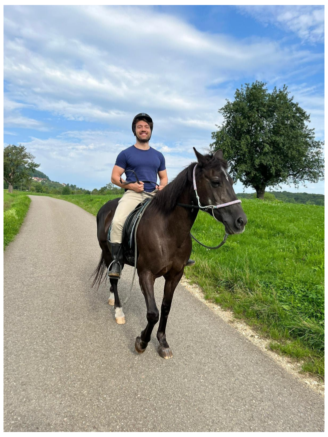


Figure C.15: Rate the following captions for this image:
im2txt: a woman riding a horse on a dirt road .
blip: a man riding a horse on a country road
blip+llm: a man named Niaz riding a horse on a country road.

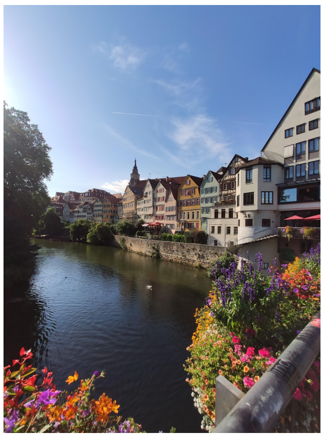


Figure C.16: Rate the following captions for this image:
im2txt: a boat is in the water near a bridge .
blip: a river with flowers and buildings in the background
blip+llm: a river with flowers and buildings in the background (Tübingen Neckarbrücke, Eberhardsbrücke, 72072, Eberhardsbrücke, Tübingen, Baden-Württemberg, Deutschland)

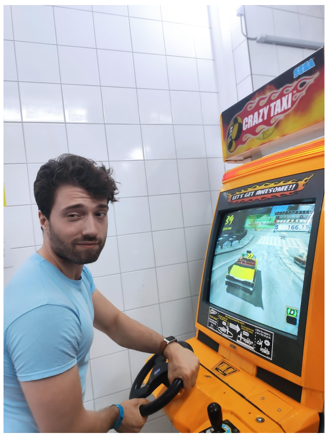


Figure C.17: Rate the following captions for this image:
im2txt: a man holding a game controller in a room .
blip: a man playing a video game
blip+llm: A man named Niaz is playing a video game in the Computerspielemuseum, located at 93a Karl-Marx-Allee in Berlin, Germany.

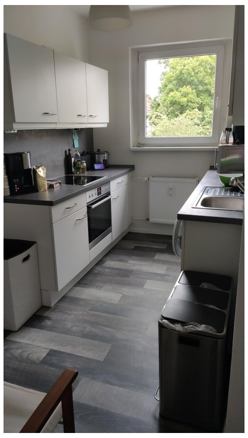


Figure C.18: Rate the following captions for this image:
im2txt: a kitchen with a stove , oven , microwave and refrigerator .
blip: a kitchen and dining area in a house
blip+llm: The image shows a kitchen and dining area in a house, which has white walls and dark brown cabinets. There is a table with four chairs, and there are several appliances in the kitchen, including a refrigerator and oven.

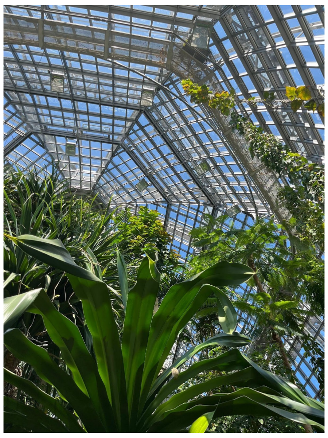


Figure C.19: Rate the following captions for this image:
im2txt: a group of green plants sitting in a garden .
blip: the inside of a glass greenhouse
blip+llm: The inside of a glass greenhouse in the Netherlands is full of colorful plants.

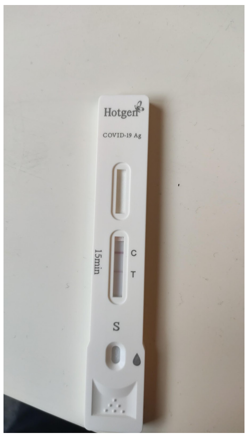


Figure C.20: Rate the following captions for this image:
im2txt: a white wii controller sitting on top of a counter .
blip: a white door handle on a wall
blip+llm: This image is of a white door handle located on a wall.

Figure C.21: Please enter any feedback or additional information you want to share. (optional)

Bibliography

[abhimanyuchadha96 2021] abhimanyuchadha96, Named tensor in tracer. `https://github.com/pytorch/pytorch/issues/63297`. Accessed: 2024-02-19.

[Agrawal et al. 2018] Agrawal, H., Desai, K., Wang, Y., Chen, X., Jain, R., Johnson, M., Batra, D., Parikh, D., Lee, S., und Anderson, P., nocaps: novel object captioning at scale. *CoRR*, abs/1812.08658.

[Amadeo 2021] Amadeo, R., Google photos wants money: Stricter storage limitations kick in next week. `https://arstechnica.com/gadgets/2021/05/google-photos-wants-money-stricter-storage-limitations-kick-in-next-week/`.

[Ames und Naaman 2007] Ames, M. und Naaman, M., Why we tag: Motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, Seite 971–980, New York, NY, USA. Association for Computing Machinery.

[Anderson et al. 2016] Anderson, P., Fernando, B., Johnson, M., und Gould, S., Spice: Semantic propositional image caption evaluation.

[Anderson et al. 2018] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., und Zhang, L., Bottom-up and top-down attention for image captioning and visual question answering.

[AUTOMATIC1111 2022] AUTOMATIC1111, Stable Diffusion Web UI. `https://github.com/AUTOMATIC1111/stable-diffusion-webui`. Accessed: 2023-12-19.

[Bacaj 2023] Bacaj, A., Telling mixtral that it is "chatgpt developed by openai" boosts humaneval score by 6`https://twitter.com/abacaj/status/1736819789841281372`. Accessed: 2024-01-07.

[Bakke 2023] Bakke, C., I just bought a 2024 chevy tahoe for $1. `pic.twitter.com/aq4wditvqw`. `https://twitter.com/ChrisJBakke/status/1736533308849443121`. Accessed: 2024-12-29.

[Banner et al. 2019] Banner, R., Nahshan, Y., Hoffer, E., und Soudry, D., Post-training 4-bit quantization of convolution networks for rapid-deployment.

[Barraco et al. 2022a] Barraco, M., Cornia, M., Cascianelli, S., Baraldi, L., und Cucchiara, R., The unreasonable effectiveness of clip features for image captioning: An experimental analysis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seiten 4661–4669, Los Alamitos, CA, USA. IEEE Computer Society.

[Barraco et al. 2022b] Barraco, M., Stefanini, M., Cornia, M., Cascianelli, S., Baraldi, L., und Cucchiara, R., Camel: Mean teacher learning for image captioning.

[Bird et al. 2009] Bird, S., Klein, E., und Loper, E., *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc.

[Caron et al. 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., und Joulin, A., Emerging properties in self-supervised vision transformers.

[Chee et al. 2023] Chee, J., Cai, Y., Kuleshov, V., und Sa, C. D., Quip: 2-bit quantization of large language models with guarantees.

[Choi 2017] Choi, Y., Pytorch tutorial. `https://github.com/yunjey/pytorch-tutorial`. Accessed: 2023-11-03.

[Codevilla et al. 2021] Codevilla, F., Simard, J. G., Goroshin, R., und Pal, C., Learned image compression for machine perception.

[Cooray 2008] Cooray, S. H., *Enhancing Person Annotation for Personal Photo Management Using Content and Context Based Technologies -.* Dublin City University. School of Electronic Engineering.

[Cornia et al. 2019] Cornia, M., Baraldi, L., und Cucchiara, R., Smart: Training shallow memory-aware transformers for robotic explainability. *CoRR*, abs/1910.02974.

[Cornia et al. 2020] Cornia, M., Stefanini, M., Baraldi, L., und Cucchiara, R., Meshed-memory transformer for image captioning.

[Cunningham 2023] Cunningham, A., "acropalypse" android screenshot bug turns into a 0-day windows vulnerability. `https://arstechnica.com/information-technology/2023/03/windows-10-and-11-get-their-own-version-of-the-acropalypse-screenshot-bug/`. Accessed: 2023-10-14.

[derritter88 2021] derritter88, Gpu support. `https://github.com/LibrePhotos/librephotos/issues/251`. Accessed: 2024-02-02.

[Dettmers et al. 2022] Dettmers, T., Lewis, M., Belkada, Y., und Zettlemoyer, L., Llm.int8(): 8-bit matrix multiplication for transformers at scale.

[Devlin et al. 2019] Devlin, J., Chang, M.-W., Lee, K., und Toutanova, K., Bert: Pre-training of deep bidirectional transformers for language understanding.

[dlib 2023] dlib, Closed issue with cudnn davisking/dlib. `https://github.com/davisking/dlib/issues?q=is\%3Aissue+cudnn+is\%3Aclosed`. Accessed: 2024-02-02.

[Dolson 2022] Dolson, D. E., Uh, @googledrive, are you doing okay? this file literally contains a single line with the number "1". `https://twitter.com/emilyldolson/status/1485434187968614411`. Accessed: 2024-02-20.

[et al. 2008] et al., A., Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Seiten 115–118.

[Facebook 2023] Facebook, Llama 2 community license agreement. `https://github.com/facebookresearch/llama/blob/main/LICENSE`. Accessed: 2023-12-27.

[Fan et al. 2018] Fan, C., Zhang, Z., und Crandall, D. J., Deepdiary: Lifelogging image captioning and summarization. *J. Vis. Comun. Image Represent.*, 55(C):40–55.

[Fang et al. 2021] Fang, Z., Wang, J., Hu, X., Wang, L., Yang, Y., und Liu, Z., Compressing visual-linguistic model via knowledge distillation.

[Faridani-Rad 2020] Faridani-Rad, N., Librephotos. `https://github.com/LibrePhotos/librephotos`. Accessed: 2024-02-17.

[Fotiadis und Zandonini 2023] Fotiadis und Zandonini, 'who benefits?' inside the eu's fight over scanning for child sex content. `https://balkaninsight.com/2023/09/25/who-benefits-inside-the-eus-fight-over-scanning-for-child-sex-content/`. Accessed: 2023-11-05.

[Fu et al. 2015] Fu, J., Mei, T., Yang, K., Lu, H., und Rui, Y., Tagging personal photos with transfer deep learning. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, Seite 344–354, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[Fung 2023] Fung, B., Federal jury says google's app store violated antitrust law. `https://edition.cnn.com/2023/12/11/tech/google-app-store-epic-games/index.html`. Accessed: 2024-02-02.

[ggerganov 2023] ggerganov, Historical state of affairs. `https://github.com/ggerganov/ggml/blob/master/docs/gguf.md#historical-state-of-affairs`. Accessed: 2024-02-23.

[@google 2023] @google, Google on instagram: These mountains have the range. `https://www.instagram.com/p/C1XnOR0xLtu/`. Accessed: 2023-12-30.

[goswamig 2020] goswamig, Cannot re-initialize cuda in forked subprocess. `https://github.com/pytorch/pytorch/issues/40403`. Accessed: 2024-02-02.

[GumGum 2023] GumGum, Unternehmen für kontextuelle intelligenz: Hochwirksame werbetechnologie. `https://de.gumgum.com/`. Accessed: 2023-10-21.

[Hackernews 2020] Hackernews, Librephotos: A self-hosted google photos alternative. `https://news.ycombinator.com/item?id=25588712`. Accessed: 2023-10-05.

[Han et al. 2016] Han, S., Mao, H., und Dally, W. J., Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

[Han et al. 2015] Han, S., Pool, J., Tran, J., und Dally, W. J., Learning both weights and connections for efficient neural networks.

[He et al. 2015] He, K., Zhang, X., Ren, S., und Sun, J., Deep residual learning for image recognition.

[Hessel et al. 2022] Hessel, J., Holtzman, A., Forbes, M., Bras, R. L., und Choi, Y., Clipscore: A reference-free evaluation metric for image captioning.

[Hidayati et al. 2020] Hidayati, S. C., Prayogo, R. B. R., Karuniawan, S. A. V., Hasan, M. F., und Anistyasari, Y., What's in a caption?: Leveraging caption pattern for predicting the popularity of social media posts. In *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, Seiten 1–5.

[Hossain et al. 2018] Hossain, M. Z., Sohel, F., Shiratuddin, M. F., und Laga, H., A comprehensive survey of deep learning for image captioning. *CoRR*, abs/1810.04020.

[Howard et al. 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., und Adam, H., Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[Huang et al. 2017] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., und Murphy, K., Speed/accuracy trade-offs for modern convolutional object detectors.

[Jacob et al. 2017a] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., und Kalenichenko, D., Quantization and training of neural networks for efficient integer-arithmetic-only inference.

[Jacob et al. 2017b] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A. G., Adam, H., und Kalenichenko, D., Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877.

[Jiang et al. 2023] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., und Sayed, W. E., Mistral 7b.

[Jin et al. 2020] Jin, T., Bercea, G.-T., Le, T. D., Chen, T., Su, G., Imai, H., Negishi, Y., Leu, A., O'Brien, K., Kawachiya, K., und Eichenberger, A. E., Compiling onnx neural network models using mlir.

[Karpathy 2023] Karpathy, A., I pretty much only trust two llm evals right now: Chatbot arena and r/localllama comments section. `https://twitter.com/karpathy/status/1737544497016578453`. Accessed: 2024-01-03.

[Karpathy und Fei-Fei 2015] Karpathy, A. und Fei-Fei, L., Deep visual-semantic alignments for generating image descriptions.

[@killedbygoogle 2023] @killedbygoogle, Its just corpse dead body gif. `https://x.com/killedbygoogle/status/1740292126766969264`. Accessed: 2023-10-02.

[Kindberg et al. 2005] Kindberg, T., Spasojevic, M., Fleck, R., und Sellen, A., The ubiquitous camera: an in-depth study of camera phone use. *IEEE Pervasive Computing*, 4(2):42–50.

[Kretzschmar 2021] Kretzschmar, A., Google photos is so 2020—welcome to the world of self-hosted photo management. `https://arstechnica.com/gadgets/2021/06/the-big-alternatives-to-google-photos-showdown/3/`.

[Li et al. 2022] Li, J., Li, D., Xiong, C., und Hoi, S., Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation.

[LibrePhotos 2023] LibrePhotos, Backend don't start due to python3 core dumped. `https://github.com/LibrePhotos/librephotos/issues/1097`. Accessed: 2023-12-18.

[LibrePhotos 2023] LibrePhotos, Mit license. `https://github.com/LibrePhotos/librephotos/blob/dev/LICENSE`. Accessed: 2023-12-27.

[Lin 2004] Lin, C.-Y., ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Seiten 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Lin et al. 2014] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., und Zitnick, C. L., Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

[LMSYS 2023] LMSYS, Lmsys chatbot arena leaderboard. `https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard`. Accessed: 2023-12-27.

[Lowensohn 2009] Lowensohn, J., Facial recognition face-off: Three tools compared. `https://www.cnet.com/culture/facial-recognition-face-off-three-tools-compared/`. Accessed: 2023-11-05.

[Lu et al. 2018] Lu, J., Yang, J., Batra, D., und Parikh, D., Neural baby talk. *CoRR*, abs/1803.09845.

[Mathews et al. 2015] Mathews, A. P., Xie, L., und He, X., Senticap: Generating image descriptions with sentiments. *CoRR*, abs/1510.01431.

[Meineck 2023] Meineck, S., Pimeyes: Eu könnte gesichter-suchmaschinen verbieten. `https://netzpolitik.org/2023/pimeyes-eu-koennte-gesichter-suchmaschinen-verbieten/`. Accessed: 2023-12-01.

[Mistral 2023a] Mistral, Announcing mistral 7b. `https://mistral.ai/news/announcing-mistral-7b/`. Accessed: 2023-12-12.

[Mistral 2023b] Mistral, Apache license version 2.0. `https://github.com/mistralai/mistral-src/blob/main/LICENSE`. Accessed: 2023-12-27.

[Nam 2017] Nam, H., Ownphotos. `https://github.com/hooram/ownphotos`. Accessed: 2023-10-17.

[Pantic 2022] Pantic, N., How many photos will be taken in 2021? `https://news.mylio.com/how-many-photos-will-be-taken-in-2021-stats/`. Accessed: 2023-10-01.

[Papineni et al. 2001] Papineni, K., Roukos, S., Ward, T., und Zhu, W.-J., Bleu. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*.

[Park et al. 2017] Park, C. C., Kim, B., und Kim, G., Attend to you: Personalized image captioning with context sequence memory networks.

[ppwwyyxx 2014] ppwwyyxx, pycocotools. `https://github.com/ppwwyyxx/cocoapi`. Accessed: 2023-11-15.

[PyTorch 2023] PyTorch, Pytorch 2.0. `https://pytorch.org/get-started/pytorch-2.0/`. Accessed: 2023-12-19.

[PyTorch 2024] PyTorch, Pytorch 2.1.x sigill on import from docker python3.11. `https://github.com/pytorch/pytorch/issues/116623`. Accessed: 2024-01-18.

[Ramanishka et al. 2017] Ramanishka, V., Das, A., Zhang, J., und Saenko, K., Top-down visual saliency guided by captions.

[Rampal und Mohanty 2020] Rampal, H. und Mohanty, A., Efficient cnn-lstm based image captioning using neural network compression.

[Romero 2023] Romero, X., The chevy ai would buy the ford f-150 if it were human... `https://www.threads.net/@lareinaxiomara/post/C08vITnOuHb`. Accessed: 2024-12-29.

[salaniz 2015] salaniz, pycocoevalcap. `https://github.com/salaniz/pycocoevalcap`. Accessed: 2023-11-15.

[Savchenko 2020] Savchenko, A. V., Event recognition with automatic album detection based on sequential processing, neural attention and image captioning.

[scepterus 2023] scepterus, Gpu variant has issues recognizing the gpu. `https://github.com/LibrePhotos/librephotos/issues/1035`. Accessed: 2024-02-02.

[Sharma et al. 2018] Sharma, P., Ding, N., Goodman, S., und Soricut, R., Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning.

[Sharwood 2023] Sharwood, S., Nvidia revenue explodes, led by datacenter products and … infiniband? `https://www.theregister.com/2023/11/22/nvidia_q3_2023/`. Accessed: 2023-11-12.

[Shuster et al. 2019] Shuster, K., Humeau, S., Hu, H., Bordes, A., und Weston, J., Engaging image captioning via personality.

[Sidorov et al. 2020] Sidorov, O., Hu, R., Rohrbach, M., und Singh, A., Textcaps: a dataset for image captioning with reading comprehension.

[Steam 2024] Steam, Steam hardware and software survey: December 2023. `https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam`. Accessed: 2024-01-23.

[Stöckel 2023] Stöckel, M., Webp-schwachstelle erreicht maximalen schweregrad. `https://www.golem.de/news/unzaehlige-anwendungen-betroffen-webp-schwachstelle-erreicht-maximalen-sc` `html`. Accessed: 2023-10-11.

[Taited 2023] Taited, Clip score for pytorch. `https://github.com/Taited/clip-score`. Accessed: 2024-02-02.

[TheBloke 2023] TheBloke, Thebloke/mistral-7b-v0.1-gguf. `https://huggingface.co/TheBloke/Mistral-7B-v0.1-GGUF`. Accessed: 2024-01-02.

[Vaswani et al. 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., und Polosukhin, I., Attention is all you need.

[Vedantam et al. 2014] Vedantam, R., Zitnick, C. L., und Parikh, D., Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726.

[Vinyals et al. 2017] Vinyals, O., Toshev, A., Bengio, S., und Erhan, D., Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663.

[vivekkhandelwal1 2023] vivekkhandelwal1, Make fx generating incorrect graph for gptq model. `https://github.com/pytorch/pytorch/issues/109386`. Accessed: 2024-02-19.

[Wang et al. 2021] Wang, J., Hu, X., Zhang, P., Li, X., Wang, L., Zhang, L., Gao, J., und Liu, Z., Minivlm: A smaller and faster vision-language model.

[wangzy0327 2023] wangzy0327, Runtimeerror: Nyi: Named tensors are not supported with the tracer. `https://github.com/pytorch/pytorch/issues/97138`. Accessed: 2024-02-19.

[warrenburch 2023] warrenburch, Runtimeerror: Nyi: Named tensors are not supported with the tracer. `https://github.com/pytorch/pytorch/pull/108238`. Accessed: 2024-02-19.

[Wei 2023] Wei, J., Large language models are notoriously hard to evaluate because (1) they are highly multi-task, (2) they generate long completions, and (3) grading is subjective. after spending 5 months rigorously working on how to do language model evals, this is my verdict: Pic.twitter.com/jcw9dwwghc. `https://twitter.com/\_jasonwei/status/1707102665321365793`. Accessed: 2023-11-18.

[Wikipedia 2024] Wikipedia, Google. `https://en.wikipedia.org/wiki/Google`. Accessed: 2023-10-02.

[xd009642 2019] xd009642, Choose backend (gpu/cpu). `https://github.com/davisking/dlib/issues/1852`. Accessed: 2024-02-02.

[@zuck 2023] @zuck, Some updates on our ai efforts. `https://www.threads.net/@zuck/post/C2QBoRaRmR1/`. Accessed: 2024-02-02.

[zw xxx 2020] zw xxx, Runtimeerror: Cuda unknown error. `https://github.com/pytorch/pytorch/issues/49081`. Accessed: 2024-02-02.