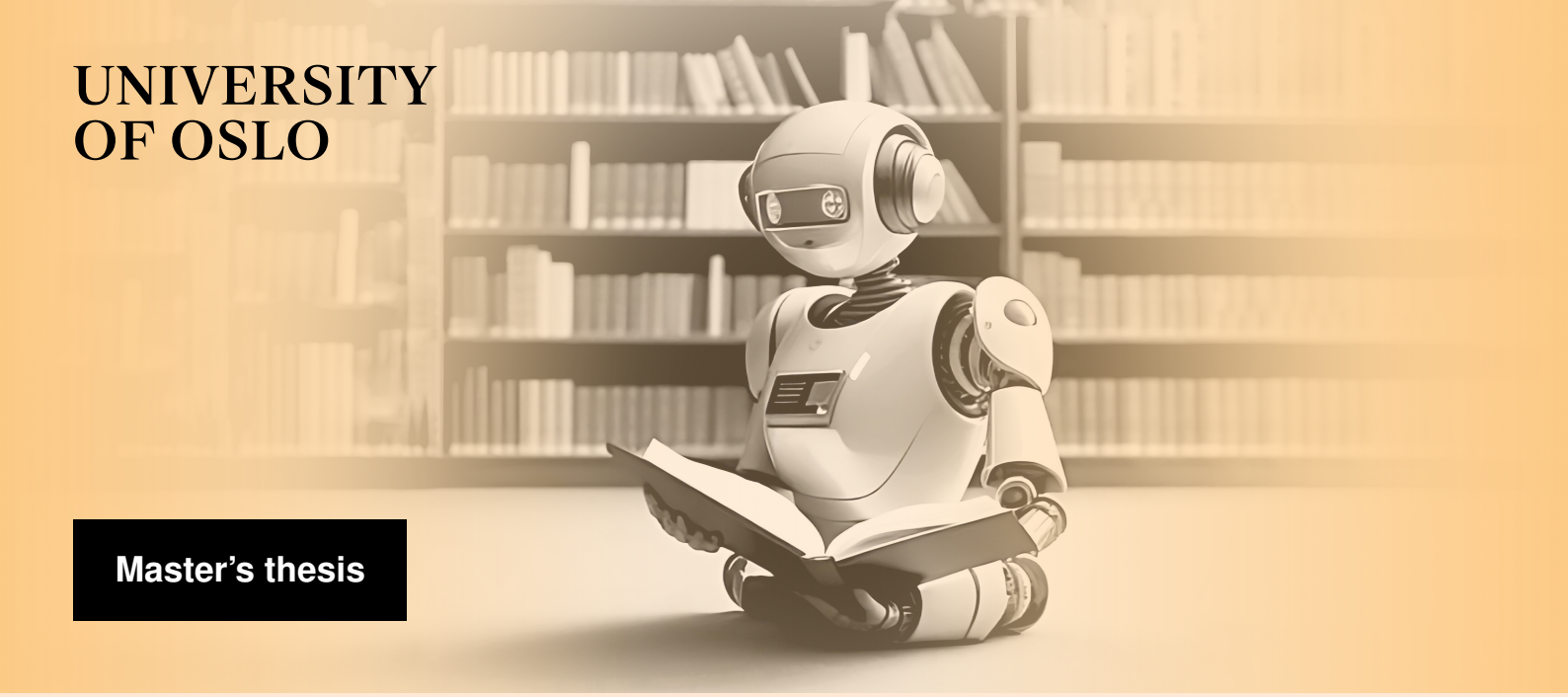
UNIVERSITY
OF OSLO



Master's thesis

Exploring the Why in AI

Investigating how Visual Question Answering models can be interpreted by post-hoc linguistic and visual explanations

Emil Christopher Gjøstøl Strømsvåg

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Emil Christopher Gjøstøl Strømsvåg

Exploring the Why in AI

Investigating how Visual Question Answering
models can be interpreted by post-hoc
linguistic and visual explanations

Supervisors:
Andrea Storås
Michael Riegler
Kyrre Glette

Abstract

With the increase in accuracy and usability of Artificial Intelligence (AI), especially deep neural networks, there has been a big demand for these networks. These methods are implemented in various domains to increase productivity, create new industries, and enhance people's lives. However, these networks are often large and complex, which does not give insight into the prediction process. In order to make the models more functional and be able to improve them, humans need to understand how they reason. This work studies explanatory models and how they can bring value and insight into how the underlying fully developed model interprets data. The experiments specifically examine how Visual Question Answering (VQA) models can be explained in both the visual and linguistic domains.

Two distinct methods are proposed to bridge the gap between models with high accuracy and interpretability. The first model combines the task of VQA with the Explainable Artificial Intelligence (XAI) method Faithful Linguistic Explanations (FLEX). The second method encodes extracted image features into the text prompt of a Large Language Model (LLM). Quantitative experiments are used to find the insights necessary. The experiments are conducted using the language model, which is explained using visualizations of the model's transition score, and a proxy model explained by Local Interpretable Model-agnostic Explanations (LIME). The main finding of this research is that larger and more complex models, like an LLM, can be explained by smaller methods added after the primary model has completed training. These models can combine complex methods with layers of explanation that bring valuable insights with no cost to the accuracy of the primary model.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence i, 1, 2, 6–8, 10, 13, 14, 16–19, 34, 40, 44, 47, 49–51, 77, 124

ANN Artificial Neural Network 25, 26

BART Bidirectional Auto-Regressive Transformers 33, 46

BERT Bidirectional Encoder Representations from Transformers 33, 45, 46

CNN Convolutional Neural Network 4, 11, 28–33, 39, 42, 43, 54, 56–69, 72, 73, 79, 83, 115, 117, 118, 121, 122, 124

CUDA Compute Unified Device Architecture 67

FLEX Faithful Linguistic Explanations i, 4, 5, 11, 43, 56–69, 82, 117, 121, 122

GPT Generative Pre-trained Transformer 7, 33

GPU Graphics Processing Unit 9, 48, 67, 71, 87

Grad-CAM Gradient-weighted Class Activation Mapping 42, 58

ILSVRC ImageNet Large Scale Visual Recognition Challenge 26

k-NN k-Nearest Neighbors 21, 22, 41

LIME Local Interpretable Model-agnostic Explanations i, 12, 42, 43, 69, 76–78, 83, 86, 104, 106, 107, 111–113, 115, 118–120, 144

LLaMA Large Language Model Meta AI 4, 8, 9, 40, 47, 48, 75, 81, 103, 112, 118

LLM Large Language Model i, 3–5, 7–11, 37, 42, 44–49, 54–56, 69–71, 73–77, 79, 80, 82, 83, 87, 95, 104, 106, 107, 112–116, 118–120, 122–124

LoRA Low-Rank Adaptation 70, 71, 82, 87, 90, 97, 112, 114, 118

LSTM Long Short-Term Memory 32, 33, 43, 61, 62, 64, 76, 122

NLP Natural Language Processing 31, 45, 47

PCA Principal Component Analysis 22

RAM Random Access Memory 87

ReLU Rectified Linear Unit 25, 47, 63

RNN Recurrent Neural Network 31, 32, 39, 43, 61, 122

ROI Region of Interest 73, 74, 98, 122

SDG Stochastic Gradient Descent 77, 78, 104, 105

SHAP SHapley Additive exPlanations 42, 76, 77

SVD Singular Value Decomposition 22

SVM Support Vector Machines 21, 104

VGG Visual Geometry Group 72–74, 81

ViT Vision Transformer 33

VQA Visual Question Answering i, 1–4, 6, 7, 10, 11, 43, 44, 55–58, 60–64, 68, 69, 79, 82, 95, 107, 111, 115, 117, 118, 120, 124

VRAM Video RAM 87, 96

XAI Explainable Artificial Intelligence i, 1, 2, 4, 6, 10, 13, 17, 40, 41, 43, 49–51, 54, 60, 76, 106, 112, 113, 115, 124

Acknowledgements

I want to thank my supervisors, Andrea Storås, Michael Riegler, and Kyrre Glette, for all your help, motivation, and interesting discussions. I am thankful that it was you guys that followed me on this journey – without your encouraging presence, this project would not be the same.

All the experiments conducted during this work have been run on the machine learning nodes at the AI HUB hosted at the University of Oslo. These are now becoming a part of the Norwegian AI Cloud (NAIC). Therefore, I would like to thank these nodes for their service, and may you have a prosperous life serving new and interesting assignments.

The header image of the cute robot on the front page is made with the help of the generative text-to-image model Stable Diffusion. The image is meant to illustrate a large language model, reading lots of books to gain knowledge. It is wonderful to live in a time where a stressed student can make a beautiful image without having any artistic qualities.

This thesis has introduced me to the depths of a very fascinating field of research. The field of AI and XAI is developing at an extraordinary speed, which is both exciting and important to be aware of, and I am curious to follow the coming advancements. Open-sourced AI methods have been essential during the development of this work.

Astonishing advancements are now made in both AI and XAI, and their discoveries help contribute to further democratizing these helpful tools. It is beneficial for interested individuals like me to have models and documentation openly available, and it contributes to the transparency of the extreme advancement.

I want to give a special thanks to all the lovely people at Blindern Studenterhjem. You have given me the best student experience that I could ever dream of. Last but certainly not least, I would like to give a huge printed hug to my family and Victoria for motivating me and supporting me throughout this entire voyage.

Chapter 1

Introduction

1.1 Background and Motivation

Artificial Intelligence (AI) have made remarkable achievements in various fields, but their lack of interpretability and opacity have brought the question of trustworthiness to the forefront of discussion. The black-box nature of AI models inhibits user trust and understanding, which can be critical in decision-making scenarios and deploying models in real-world applications. This lack of transparency limits the ability to address bias, identify model limitations, and provide accountability.

Explainable Artificial Intelligence (XAI) is both a research area and a set of techniques to address these weaknesses by providing methods and techniques to explain the decisions and predictions made by AI models. Motivation for XAI methods arises from the increased deployment of AI systems, especially in high-stakes domains such as finance, criminal justice, and healthcare, where justification is vital. Additionally, XAI is important in consumer-facing decision-making applications using AI, where users may not have the technical expertise to understand the inner workings of an AI system.

As many XAI methods strive to provide locally accurate explanations, that is, explanations faithful to the underlying model, they often include low-level features [1]. However, often these features are challenging to convey to a non-technical user or do not provide an intuitive explanation. A user can gain a fuller picture of a model's reasoning by having higher-level explanations that come from multiple modalities [2]. The multimodal task of Visual Question Answering (VQA) combines text

questions in natural language and corresponding images to get a model to output a response based on these two modalities. These multimodal tasks are closely related to human cognition, as understanding and describing visual scenes with language is a fundamental attribute of human perception.

The combination of the VQA task and XAI is an important area of research since it has the potential to bring the most intuitive explanations to researchers and users. By describing the contents of images in natural language and being able to answer questions regarding images and text, users can gain a better insight into these complex models and determine if they can be trusted in critical assignments.

This work explores how VQA systems can be explained using visual and linguistic justifications. As complex models often achieve the highest accuracies, the experiments in this work explore how smaller, transparent, and explainable methods can be attached to a fully developed primary model. These models can provide valuable insights during development and in the real world at no cost to the predicting model's accuracy.

1.2 Problem Statement

To be truly trustworthy, machine learning and computer vision must be understood. Researchers need to understand the model's inner workings to make improvements and uncover biases in the dataset. Users and domain experts using the models must be confident that the model makes accurate predictions and uses meaningful features when evaluating the task. The goal when designing models should be that they are able to achieve state-of-the-art performance without compromising the interpretability and explainability of their predictions. The more complex the models become in the pursuit of greater accuracy, the more difficult it becomes to interpret them. This also applies to multimodal models, as they combine information from multiple modalities into one prediction. However, since multimodal explanations can provide valuable and intuitive considerations, they are an important area of research.

This thesis will examine how explanatory models in the linguistic and visual domain can contribute to gaining new and important insights into the functioning of the AI methods.
Formally, the goal can be written as follows:

- To what extent can VQA with explanatory models in different domains provide additional insights into the underlying data?

Although the following research questions are specified, they should be viewed as guidelines for the experiments rather than defining rules. This is because the overarching goal is to investigate how explanatory models can provide additional insights into different modalities. The goal is, therefore, to explore this topic in more detail to gain new insights rather than to answer a specific problem.

Since the task of VQA combines both linguistic and visual understanding, the visual part benefits from being explained, as it contains significant information regarding the task. A locally accurate explanation ensures that the reason is based on the underlying model's evaluation. However, for an explanation to be locally accurate, some explanatory models work on features on a low level without explaining the task as a whole. As humans often evaluate the contents of an image as a whole instead of low-level features, the explanation model could benefit from using locally accurate features and extrapolating so that they describe the image on a high level.
More precisely, the research question for the visual domain can be formulated as follows:

- Will the answers given by a VQA system be more intuitively explained with additional locally accurate image descriptions?

With the knowledge from answering these questions, better explanations can be made, which makes it easier to develop and interact with models interpreting images.

To have an explanatory model in the linguistic domain, it should only evaluate text features. Since Large Language Models (LLMs) are trained on large text corpora, they can learn connections and gain valuable insights into how language works.
Specifically, the research questions in the linguistic field can be formulated as follows:

- To which degree can an LLM fine-tuned on a new modality bring new insights from its pertaining?

- What insights can additional explanatory methods bring from an LLM after training is complete?

The insights gained from answering these questions can be used to create LLMs that bring knowledge from previous training to a specific task while being explained by supplementary explanatory methods. LLMs with additional methods that give an understanding of how they work helps researchers and users gain a more intuitive understanding of how these complex models work and interpret data samples.

With these research questions answered, the knowledge gained from this work will make it easier to develop models that can give insight into how models understand broad concepts in vision and language. Further on, this will help humans understand what these complex models see and value, potentially leading to increased trust.

1.3 Scope and Limitations

In order to answer the research questions and find a solution to the problem posed, the scope of the work must be focused. This section describes which methods are used, their scope, and where the main limitations lie. Two different models are proposed and described, one taking on the explanation in the visual domain and the other in the linguistic domain.

For the visual domain, the FLEX-VQA method is suggested. This is a fusion of an XAI method called Faithful Linguistic Explanations (FLEX) [3] with the VQA task. The FLEX-VQA method achieves the ability to both answer questions and describe images using locally accurate visual features. The features are extracted from the feature maps of the Convolutional Neural Network (CNN), which are pre-labeled with descriptive words. Finally, a language model converts the words from the feature maps into natural language descriptions.

In the linguistic area, an LLM is used in this work. The model is based on the Stanford Alpaca [4, 5], based on Large Language Model Meta AI (LLaMA) [6]. This Alpaca model is adapted to the multimodal VQA task, giving it the name Alpaca-VQA. Since it is an LLM that has been pre-trained on a large corpus, it can use its extensive knowledge base to gain new insights and connections, thus expanding the available dataset for fine-tuning.

Since training an LLM requires a large number of computational resources, it is neither feasible nor ethical to train it from scratch for

the experiments in this work. Fortunately, pre-trained LLMs are openly available and can be fine-tuned to the specific task. Since the subject of this work is not to create a state-of-the-art LLM, the experiments were limited to using a single LLM. One of the measures used to accomplish that the LLM can be analyzed and evaluated with as little bias as possible is that it is trained to only output a single answer to a question. LLMs have the ability to answer in long sentences, but the quality of natural language can be challenging to evaluate [7]. Therefore, this work limits the answer length to test the answer directly with ground truth.

Since the initial implementation of the FLEX framework uses a variant of VGG16 [8, 9], it was decided to use the VGG16 model to also extract image features into the LLM prompt. Other methods could have yielded better results, such as object recognition models. Still, by having the same image extractor in both FLEX-VQA and Alpaca-VQA, their explanations could be compared more easily.

The experiments performed in this work are trained using a single dataset. Although multiple datasets could have examined more general findings, the scope was narrowed to focus on complementary explanatory methods. This is because these explanatory methods do not focus on the specific dataset but rather on how the primary model uses the samples.

1.4 Research Methods

To effectively answer the research questions, this work incorporates quantitative studies to collect the necessary results. Quantitative research methods provide a structured and systematic approach to collecting and analyzing data, allowing for a rigorous investigation of the research objectives. This work aims to use these methods to gain objective and reliable insights contributing to a more comprehensive understanding of the topic. The models and experiments performed in this research are carefully designed to allow for an unbiased analysis of the subject matter.

The models and experiments conducted in this research are carefully designed to facilitate an unbiased analysis of the results. An unbiased analysis is an approach that minimizes the influence of personal biases or preconceived notions on the interpretation of data. It involves implementing rigorous experimental protocols, ensuring the validity and reliability of measurements, and maintaining transparency throughout the

research process.

By adopting an unbiased analysis approach, this work seeks to ensure that the obtained results are not skewed or influenced by subjective factors. This approach allows for a more objective evaluation and interpretation of the findings, promoting the reliability and generalizability of the results. This will enable researchers to draw meaningful conclusions and contribute to a broader understanding of the subject area while increasing the credibility and validity of research results.

1.5 Ethical Considerations

With more intuitive explanations provided by an XAI system, researchers and users have better insight into the inner workings and arguments of the underlying AI method. This, in turn, will address one of the ethical concerns, which is dataset bias.

Dataset Bias

AI raises a number of ethical concerns, particularly when used in high-stakes domains such as healthcare, finance, and criminal justice. The concern is not specifically the error of the AI model but rather the bias of datasets. Since AI is used to condense complex datasets into understandable predictions, it becomes an inherited ethical issue if models learn unfair biases. Datasets used in deep learning are often large and complex, and AI models are used to extract knowledge from the dataset. Therefore, it can be difficult for researchers and users to identify biases used in training the AI models, especially when they do not explain why this prediction is correct.

One ethical problem is that AI systems can make predictions that are directed against certain groups of people, such as those from marginalized communities. This could occur if the training data used to develop the system is biased or if the system's decision-making process is not transparent. XAI can therefore be an important tool to gain insights into our own biases and to help researchers and users avoid making decisions based on false premises.

VQA datasets and image caption tasks also raise ethical concerns about bias and fairness. The task of VQA and image caption rely on data in

order to learn to generate textual explanations for visual inputs. However, if the training data is biased in any way, the resulting system will also be biased. For example, if a VQA dataset contains mostly humans with one skin color, the system may not work as well with images of people of a different color. Similarly, if a dataset used for captioning contains a relationship between sexes and professions, the same bias will be carried over to the model.

The work of Hirota et al. [10] analyzed gender and racial bias in five popular VQA datasets and found unfavorable stereotypes in the samples. Various possible solutions to address this issue were explored in their work. This includes not asking questions about race and sex when not required when making a dataset and collecting a more standardized distribution related to race and sex. They also propose an alternative to the manual screening that some VQA datasets use since not all can justify the cost of manual annotation. The proposed solution to this automatic screening, followed by ethical guidance for annotators, and lastly, a feedback platform for users.

Large Language Models

Regarding LLMs, there are some ethical considerations to address. LLMs are trained on a large corpus of text, most often collected from the Internet. The data gathered across the web are mostly written by humans, and the LLMs are trained on this wast dataset to extract a more general understanding of human knowledge and present this with a well-structured syntax. With the advancements and availability of models such as Generative Pre-trained Transformer (GPT)-4 [11], combined with a user-friendly and easy-to-use user interface, like the one used by ChatGPT [12], the public has never before had an advanced AI so accessible in their everyday life. Given that users come from different backgrounds and cultures, the LLM should be able to adapt to its users to close the gap between user and AI alignment. AI alignment is a subfield of AI safety, with the goal of building safe and reliable AI methods [13]. The alignment problem is defined as the task of aligning the goals of the humans creating the system with the goals of the AI system. Outer alignment is the overall goal of the system, and inner alignment ensures that the system behaves as expected in a robust manner [14].

LLMs have billions of parameters and incomprehensible feature space

for humans to understand fully. These models are mostly built today using the Transformer model proposed by Vaswani et al. [15], but because of their size and scale, they are very complex to understand [16].

There have been multiple attempts recently to slow the growth of or to regulate deep neural networks, like LLMs. The goal of this is to have time to address ethical and technical concerns before a new, larger, and better model gets released [17, 18]. The reasons why some argue not to continually make new models are many, but the most prominent concern is to understand better the models already developed. With a better understanding of these models, researchers are better equipped to tune models to better align with the overall goal of humans. An understanding of the inner workings of a model, including an LLM, can also give insight into how to make the model predictions more transparent, fair, and efficient.

Ensuring the ability to detect AI-generated content is a crucial aspect that deserves significant attention. As AI becomes more proficient at autonomously searching the web and generating new datasets, it becomes increasingly important for humans to determine the origin of information and assess its reliability and credibility. These tools play a crucial role in providing transparency and allow users to differentiate between content generated by humans and AIs. By integrating reliable detection tools, individuals can make informed decisions based on awareness of whether the information they encounter comes from AI algorithms or human sources.

Carbon Footprint

Training large AI models takes a considerable amount of energy and therefore produces a lot of greenhouse emissions. In the paper introducing the LLaMA model [6], which is the base model of the Alpaca-VQA model used in this work, they calculate the carbon footprint. This footprint can be estimated by the formula in Equation 1.1.

$$\text{Wh} = \text{GPU hours} \times (\text{GPU power consumption}) \times \text{PUE} \tag{1.1}$$

Where *PUE* is Power Usage Effectiveness, which is set to 1.1. In order to generalize the carbon footprint, the authors of LLaMA use the US national averages carbon intensity factor corresponding to 0.385

kg CO_2eq/KWh, where CO_2eq is Carbon Dioxide Equivalents. By generalizing the carbon emissions per watt, it is easier to compare models trained in different locations. The formula used for calculating the metric tonnes of CO_2eq can be done with the formula in Equation 1.2.

$$\text{Metric tonnes of } CO_2\text{eq} = \text{MWh used } \times 0.385\text{kg } CO_2\text{eq/KWh} \quad (1.2)$$

Following this equation, the researchers estimate the development of LLaMA to use 2,638 MW in total, corresponding to 1,015 metric tones of CO_2eq. This is roughly the same amount as 216 passenger vehicles emit in a whole year [19].

In order to minimize the carbon footprint of the LLM used in this thesis, the weights used are not from the original LLaMA but from the Stanford Alpaca model. The model is fine-tuned using optimizations that freeze the internal weights of the model, and a supplementary weight matrix is trained. This implementation is discussed in further detail in section 3.2. Also, the experiments follow an iterative design, which helps reduce unnecessary computing time and emissions. Following the same formula, the fine-tuning of the Alpaca-VQA model used in this work can be calculated. It is estimated that the total Graphics Processing Unit (GPU) processing time to be 30 hours, with GPUs using a maximum of 250 W [20]. Using the same PUE as LLaMA, the total power consumption is 7.5 kW. Using the carbon emissions from the US, the experiments in this work have emitted 2.888 kg of CO_2eq. However, as the servers were located in Norway, the emissions, in reality, were much less. It is estimated that the average emissions per kWh in Norway in 2022 were 0.019 kg CO_2eq/KWh [21]. Therefore, the actual emissions are 0.143 kg of CO_2eq, which is less than a kilometer driven in a fossil car [22].

As the fine-tuning used to conduct the experiments in this work is dependent on the base model, the emissions should therefore be seen in conjunction with the final model. Therefore it is an ethical concern to keep making larger models that consume more energy, especially when the energy used is not from renewable sources.

To address the issue of large computational needs, processor designers have made specialized chips for machine learning [23–26]. These can assist the development of even better models that also have reduced power consumption. An additional benefit of having reduced power

consumption is that more lightweight devices can train and run models that help democratize machine learning.

In summary, AI, datasets VQA, and LLMs raise important ethical concerns about bias, fairness, and carbon footprint. Mainly these concerns stem from reliance on training data, which may be biased or perpetuate stereotypes. Researchers and users of these tools must consider the ethical concerns and work towards developing XAI systems that are transparent, fair, and respectful of humans and the climate.

1.6 Main Contributions

The experiments conducted in this research project demonstrate the effectiveness of post-hoc explainability models in providing valuable insights into the underlying model without sacrificing accuracy. By employing a proxy model that simulates the behavior of the LLM, a useful understanding of how the LLM works can be gained. The visualization of transition scores further contributes to the intuitive description of how the LLM predicts tokens in a sequence.

This work explores the adaptability of LLMs to new modalities while preserving pre-training knowledge. By adding additional methods to a fully trained model, it is possible to leverage the benefits and accuracy of the initial model while providing users with intuitive justifications for its answers.

The key finding of this research is that larger and more complex models, like an LLM, can be explained by smaller methods added after the model has completed training. These additional explanatory methods add no significant resource use or compute time during inference but provide valuable insights into the model. In addition, these supplementary models do not change how the larger, more complex model works. Therefore, these methods can combine complex models with supplementary layers of explanation that bring valuable insights with no cost to the accuracy of the primary model.

The main contributions are an understanding of how smaller and interpretable methods can be added to a model that is finished training. These methods provide valuable insights into how the main model may work, even though the methods are not interpreting the main model itself but a proxy model. All the code used in the experiments conducted in this

work is open and available.[1,2]

1.7 Thesis Outline

This work is divided into five main chapters. All the references used are listed after the last chapter. The thesis is structured in the following way:

- **Chapter 1: Introduction** - This is the introduction to this project and thesis. It is designed to be intuitive to follow and convey this research project's motivation and overall goal.
- **Chapter 2: Background** - This chapter establishes the foundation and context for the subsequent thesis chapters. It provides the necessary background knowledge and related work for understanding the research conducted in this thesis. It covers the motivation for the research, the importance of XAI, relevant technologies such as AI and machine learning, model explanation techniques, evaluation metrics, and a summary of related work in the XAI field.
- **Chapter 3: Methodology** - This chapter delves into the details of two proposed methods in this work, both aiming to explore the implementation of explanatory methods in different domains using machine learning models. The first method, FLEX-VQA, combines a VQA architecture with the FLEX framework to provide explanations originating from the visual domain translated into natural language. The second method involves an LLM combined with a CNN, where image features are translated into text for explanation. The chapter highlights the significance of these models' multimodal capabilities and discusses the specific techniques employed. The Alpaca-VQA model, chosen for its pre-training and efficiency, is fine-tuned using the LoRA technique.
- **Chapter 4: Experiments and Results** - In this chapter, the Alpaca-VQA model is tested on two different dataset sizes, starting with an investigatory experiment on a smaller dataset to gain initial insights. The model is then trained on a larger dataset of 20,000 samples, and the results are analyzed. Using explanatory post-hoc methods, such

[1]https://github.com/2ec/alpaca-lora
[2]https://github.com/2ec/FLEX-VQA

as a proxy model explained by Local Interpretable Model-agnostic Explanations (LIME) and visualizing transition scores, uncovers biases in the dataset and provides supplementary information about the model's behavior. The chapter also explores linguistic biases by testing a language-only version of the model. The key takeaway is that smaller explanatory methods can be added to larger models after training, providing valuable insights without compromising accuracy or computational resources.

- **Chapter 5: Conclusions** - This chapter concludes this thesis and summarizes the findings from the previous chapter. Limitations of the methods used are discussed, as well as future works are suggested.

Chapter 2

Background

This chapter provides an overview of the background knowledge and related work needed for understanding the research performed in this thesis. The chapter begins by explaining the motivation for this research and the importance of XAI. The chapter also presents existing technologies relevant to the research, such as AI, machine learning approaches, and model explanation techniques. It discusses the current state-of-the-art in the XAI field, including recent advancements and research trends. Additionally, the chapter summarizes the most relevant evaluation metrics, which are essential for assessing the effectiveness of AI and XAI techniques.

Finally, this chapter provides an overview of the related work done in the XAI field, focusing on the most relevant studies and research findings. It examines the approaches and techniques used in these studies and discusses their strengths and limitations. The chapter then discusses the key concepts and principles of XAI, including the various types of interpretability and the methods used to achieve it.

In essence, this chapter lays the foundation for the subsequent chapters of this thesis, providing the necessary background and context for understanding the research presented.

2.1 Artificial Intelligence

AI is both a research topic and a group of approaches aimed at giving computer programs the most intelligent possible approach to solving a given task. In academia, it became a research topic in 1956 after a

conference called the Summer Research Project on Artificial Intelligence, which was hosted by John McCarthy and Marvin Minsky, which is considered to be where the term originated and is often credited to McCarthy [27, 28]. Since this first conference, the field has evolved with the help of an interdisciplinary environment of computer science, mathematics, statistics, psychology, neurology, and linguistics.

Overall, this section of the thesis provides a brief overview of the history of AI and sheds light on the importance of the field in modern society. It forms the basis for the further discussion of relevant AI concepts in the following sections of this work.

Throughout history, humans have had the desire to wield the power of creating artificial creatures. The concept of artificial beings that are able to observe, evaluate and react to their surroundings can be dated back to the story of Talos in Greek mythology [29]. The concept has since been featured in countless fiction novels and movies, such as Frankenstein, R.U.R, HAL9000, and Metropolis [30–33].

Humanity has since taken the dream of artificial beings with consciousness from the realm of fiction and driven the evolution of AI as a research field forward with impressive results. AI has become an integral part of people's everyday lives, fueling advances in science and technology, including social media, computational photography, voice assistants, and chatbots [34–36]. Doctors can now use AI to interpret medical images, and artists are using AI to create new art [37, 38].

2.1.1 A Short History of AI

The origins of modern AI can be traced back to the 1940s and 1950s when pioneers such as Alan Turing and John McCarthy laid the foundations for today's AI research methods.

Alan Turing was a British mathematician and pioneer of computer science, and he is most notably known for his contributions to the development of modern computer theory. In 1936, Turing presented his theory of computation, which proposed a system later coined as a Turing machine, which is an abstract machine that manipulates a strip of tape using only simple rules. Despite its limited rules, they can be combined to interpret any proposed computer algorithm. The theory was revolutionary at the time and laid the foundation for the development of

modern computers and later became known as the Church-Turing thesis [39, 40].

One of the first to use Turing's theories in practice was McCullough and Pitts and their formal design of artificial neurons in 1943 [41, 42], which was the first step towards the method now known as the perceptron. In 1950, Turing proposed a behavioral test, originally called the imitation game and later named the Turing test. This behavioral test was set up so that a human would speak to two separate entities, the human judge knowing that one was another human and the other a machine. Then the human would talk to those two participants, not knowing whether the responder was the human or the machine, and then try to decide who the machine was. The computer would pass the Turing test if the human judge could not distinguish between human and computer. This test is relevant today with tools such as ChatGPT [12], where the boundary between human and machine-produced natural language disappears. Turing's work laid the foundation for modern computers and artificial intelligence. His ideas are still relevant today, and his impact on the field of computer science cannot be overstated. The Church-Turing thesis, in particular, had a significant impact on the research field of studying computers and giving an understanding of the limitations of computer systems. Turing's work also laid the foundation for the development of artificial neural networks and their use in machine learning.

The research field of artificial intelligence has gone through several ups and downs over the years. These ups and downs have mostly been caused by increased popularity and interest in founding new technologies in the field, followed by disappointment when the development didn't deliver the desired results. Disappointment led to cuts in funding, which stagnated development. The AI industry has experienced two significant cycles of reduced interest, often referred to as the two AI winters. The first AI winter was in the late 1970s, and the second was in the late 80s to early 90s [43]. Since the beginning of 2012, interest in AI has increased, driven in particular by advances in machine learning, deep learning, and increased computing resources. With the advent of deep learning and big data, AI has seen a resurgence, and unprecedented advances have been made in the field in recent years. This has led to the development of intelligent systems that can take on a variety of tasks that were once thought to be

an exclusive domain of humans, such as driving cars and interpreting images.

As there exist many ways to solve a problem, researchers have divided the AI field into two main paradigms that take different approaches to achieving artificial models. The two paradigms are named symbolic and connectionist AI.

Symbolic Artificial Intelligence

Symbolic Artificial Intelligence refers to a collection of AI methods based on high-level symbolic representation of problems. It is also known as rule-based AI, classic AI, and good old-fashioned AI (GOFA) [44].

In the period between the 1950s and mid-1990s, symbolic representation was seen as dominant over the other main representation, connectionism, which will be discussed in the next sub-chapter. This was because symbolic representation came closer to the thought human way of learning symbolic representations rather than analyzing neuron activation in the brain and therefore had an advantage over connections [45]. The goal was to provide machines/computer code with human knowledge and patterns of action by defining rules in programs. The main idea behind symbolic AI is to define a high-level symbolic representation that creates the building blocks for the intelligence of the program. In addition to these pre-programmed rules, an expert system is used, which then checks which rules are fulfilled and makes a decision on this basis.

The advantage of symbolic AI is that because of the pre-programmed rules, it is more transparent about what underlies the decisions, and, given the same assumptions, the same result is obtained every time. Because of the rule-based nature of the algorithms, they work well even on data with little variation. They also work well for datasets with few examples because programmers can use *a priori* knowledge of the problem when defining the rules.

Symbolic AI has disadvantages in that irregular variations cannot be accounted for other than creating new and additional rules to cover all variants that may be encountered. This is a disadvantage in systems that process data from the real world, as they often contain large variations. It is, therefore, often practically impossible to write rules to cover all variations unless the task is severe and well-defined.

Connectionist Artificial Intelligence

Despite developing in the background of Symbolic AI during the first decades of its academic history, new methods have made connectionist AI remarkably popular in recent times. Connectionism is based on the idea that the interconnection of several small nodes, which learn over time, forms intelligent judgment systems. This approach is often compared to how the brain works, a connected network of neurons. The most well-known example of such a method is the artificial neural network, which is built up of many processing nodes, often called perceptions [41].

The perceptron is inspired by biology and mimics a biological neuron. This perceptron takes data as input, weights it according to its learned importance, and then uses a transfer function to provide an output. These small artificial neurons are connected in layers and learn from sample data fed into the model. The model learns by adjusting weights that define the importance of input to make a correct assessment.

Since 2012, models that can train on increasingly large datasets have improved significantly, in large part due to advances in computational power. Connectionism has given a new boost to AI as a field and tool in real-world applications, and they are constantly being renewed to solve new problems.

The main benefit of connectionism is that the model can learn more complex relationships than symbolic AI since the weights in the neurons can be updated as new examples are trained. This allows the method to extract information from datasets with large variations between individual samples.

The main downside, however, is that it's often difficult for humans to gain insight into the formation of judgments, as the process is learned by adjusting weights in each of the numerous neurons. The models, therefore, can learn connections that do not make sense to humans and can be impossible to explain intuitively. Then, even if the inner workings of the model can be explained, it will bring no value to the human evaluating the explanation.

Another disadvantage of these models is that they often become more accurate with more data and more complex model structures, making accurate models even harder to interpret. Because of the difference in how humans reason and some deep artificial networks learn, new research fields like XAI try to make models more transparent and understandable.

Combination of Symbolic and Connectionist AI

With the rapidly growing demand and development of new AI tools, there is increasing pressure to make connectionist models more transparent by combining them with symbolic AI methods, known as neuro-symbolic AI [46]. The advantage of combining these two paradigms is that these combined models utilize the properties of connectionist models to train on large datasets with significant variations, in addition to learning symbolic representations of the dataset, making it easier for humans to understand the basis for the model's decisions.

In the book *Thinking Fast and Slow* by Daniel Kahneman [47], a model of human thinking is discussed. It is proposed that human thinking is composed of two systems, one fast and intuitive and the other a slower, more deliberate system. The book argues that the symbiosis of these two systems makes human thinking more robust and reliable. In this context, the neuro-symbolic AI comprises a fast symbolic pattern recognition system, and a connectionist deep learning model takes care of the more deliberate reasoning.

This type of combined model can train on large datasets of, for example, images while learning symbolic representations from question-and-answer pairs to become familiar with linguistic terms like colors and shapes. The hope is that this will lead to a more general understanding from the model's perspective by learning multimodally and being able to learn variations in the dataset using fewer examples while being reasonably transparent.

The integration of models across modalities is further explored in this work. It is an interesting research area where the combination of connectionist and symbolic approaches can make AI models more transparent, easier to understand, and more meaningful to work without sacrificing accuracy and applicability.

2.1.2 Machine Learning

The field of artificial intelligence includes a subgroup subset known as machine learning. Because these two groups are often conflated in the media, reasonable to define machine learning as a subset of AI. However, some argue that machine learning has diverged from AI, with overlaps, overlaps as illustrated in Figure 2.1 [48]. Nevertheless, this thesis follows

the conversation most agree on and treats machine learning as a sub-field of AI, as shown in Figure 2.2.

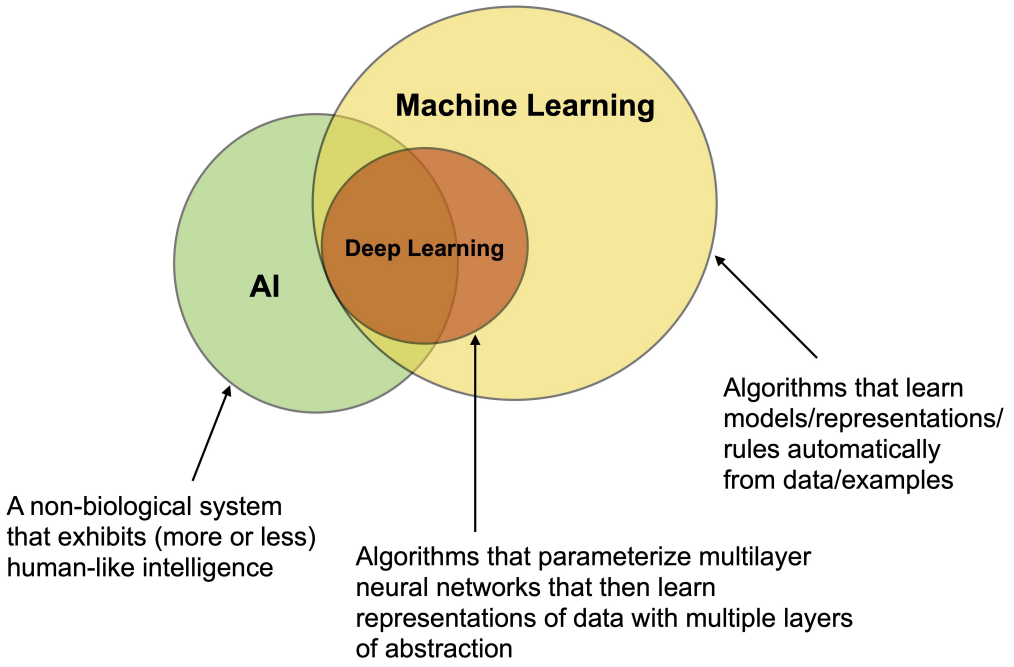


Figure 2.1: Overview of an alternative structure for AI, machine learning, and deep learning. This structure argues that AI is the pursuit of developing non-biological systems with human-like intelligence and can be achieved with methods that do not use machine learning or deep learning, such as symbolic representations in a shallow architecture. This representation is not de facto and is not used in this work.
Figure by Sebastian Raschka [48]

Machine learning is often broken down into three sub-areas, which are discussed and explained in this sub-section, explaining how they work, under what circumstances each sub-area can be used to advantage, and the challenges each branch faces are highlighted.

Supervised Learning

The first category within machine learning is known as supervised learning. This method is conceptually similar to using flashcards for memorization. The participant draws a card and reads the question while keeping the answer hidden. Only after the participant answers the question, the correct answer can be revealed. This is, in essence, the same approach used in supervised learning.

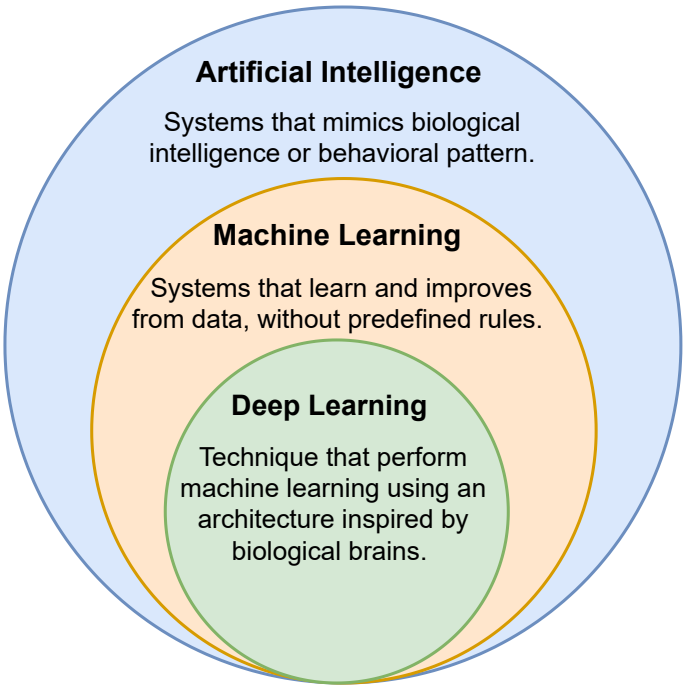


Figure 2.2: Overview of the structure between artificial intelligence, machine learning, and deep learning. This is the structure used in this thesis.

In supervised learning, a dataset with pre-labeled examples, also known as a fully labeled dataset, is used to train algorithms to predict labeling for unlabeled input data. This is typically accomplished by training on a larger dataset with labels for each input, passing the input through a network, and using the label to inform the algorithm what the input data should be labeled as. In this way, the method learns the relationship between the input and the correct label. The method learns these correlations by adjusting the weights within the network to produce a result that is as close as possible to the true label.

Since the ground truth label is known for each entry in the dataset, the model's output is compared to this ground truth label. The accuracy of the similarity between these two is measured using a loss function. A loss function is a calculation that computes the loss of a single model prediction. The loss is a measure of how far from the ground truth the prediction was and is used to adjust the model to predict better next time. Weights are adjusted using a technique called backpropagation to minimize this loss, and training continues until the weights are appropriately adjusted to achieve a satisfactorily low loss. These weights

are then used later when testing real-world inputs, for which ground truth is often unavailable and must be estimated.

Within supervised learning, there are two main subcategories: classification and regression.

Classification is a method used when the output variable belongs to a category, such as a color or an object, or to indicate the probability that an input belongs to one or more categories. Classification either predicts categorical class labels or classifies data by building a model. A typical classification method can be used to sort emails into spam or not spam or to identify the bird species in an image. Classification models can include methods such as neural networks, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN) [49, 50], random forest, decision trees, one-vs-rest, and naive Bayes. Section 2.1.3 will go into more detail about neural networks, as this is a central method of machine learning and is relevant to this work, as it is used later in this thesis by the first proposed architecture in Section 3.1.4.

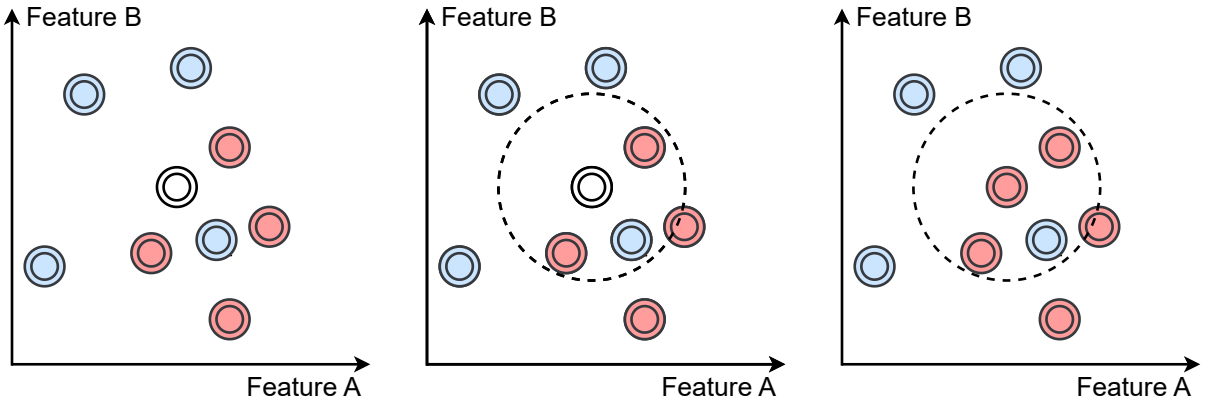
The second approach within supervised learning is called regression. This method is used when the output should be a real or continuous value. Regression is used to find the correlation between dependent and independent variables. There are many different models within regression, such as linear regression, logistic regression, and polynomial regression. It is common to use hyperplanes that fit the available data. Hyperplanes are defined as a subspace of a vector space and have one less dimension than the vector space they are inside. These planes, therefore, are used to make a cut in the feature space in order to separate data samples. Regression is often used to make forecasts based on past data, e.g., estimating real estate prices for an area or salary growth.

Unsupervised Learning

Unsupervised learning is a technique that uses raw data from a given dataset to identify patterns and relationships in unlabeled data, facilitating the grouping of each item into appropriate clusters. The essence of this approach is that it can autonomously analyze and group data without requiring human intervention to pre-label the data. Therefore, it is well suited for use in data analysis, image recognition, and image segmentation.

The main grouping models used in unsupervised learning are cluster-

ing, association, and dimension reduction. Clustering aims to divide raw data into groups or clusters based on similar characteristics. An example of a cluster split is shown in Figure 2.3, demonstrated by the k-NN method. Association rules, on the other hand, are a rule-based approach that attempts to uncover relationships between variables in a dataset. These associations can, for example, be used in recommendation models, where companies can identify correlations between customers who purchase different items and use this knowledge to suggest similar products to new customers.



(a) Samples are distributed in a feature space. The colors represent class, and the white is a new sample. A new sample will be classified based on neighboring samples.

(b) The new sample checks the nearest neighboring samples. In a circumference, noted with a dotted circle, the 4 closest neighboring samples are found.

(c) The classes of the 4 closest samples are counted, and the majority class is chosen. Of the 4 samples, one is blue, and three are red. The new sample is therefore classified as red.

Figure 2.3: Example of how the clustering algorithm k-Nearest Neighbors computes the class of a new sample.

Dimensionality reduction is a technique for reducing the number of dimensions in a dataset. Although many machine learning methods benefit from additional data, this is not always the case [51, 52]. Data samples can often contain insignificant variables or noise, making model training ineffective and making the dataset in practice smaller regarding useful information. This technique can be used to create a dataset containing more distinct features and also visualize and analyze data. When reducing the dimensions of a dataset, the goal is to reduce the number of learnable parameters while maintaining the integrity of the data being processed. Well-known methods for dimension reduction are Principal Component Analysis (PCA) and Singular Value Decomposition

(SVD). More recently, good results have been obtained with deep neural network-derived autoencoders [53]. Autoencoders can compress data fed to them and produce a representation or encoding that retains much of the underlying data, but the dimensions are significantly reduced. To retrieve this data after encoding, a decoder is required to reconstruct the data. Figure 2.4 depicts an autoencoder with an encoder and decoder.

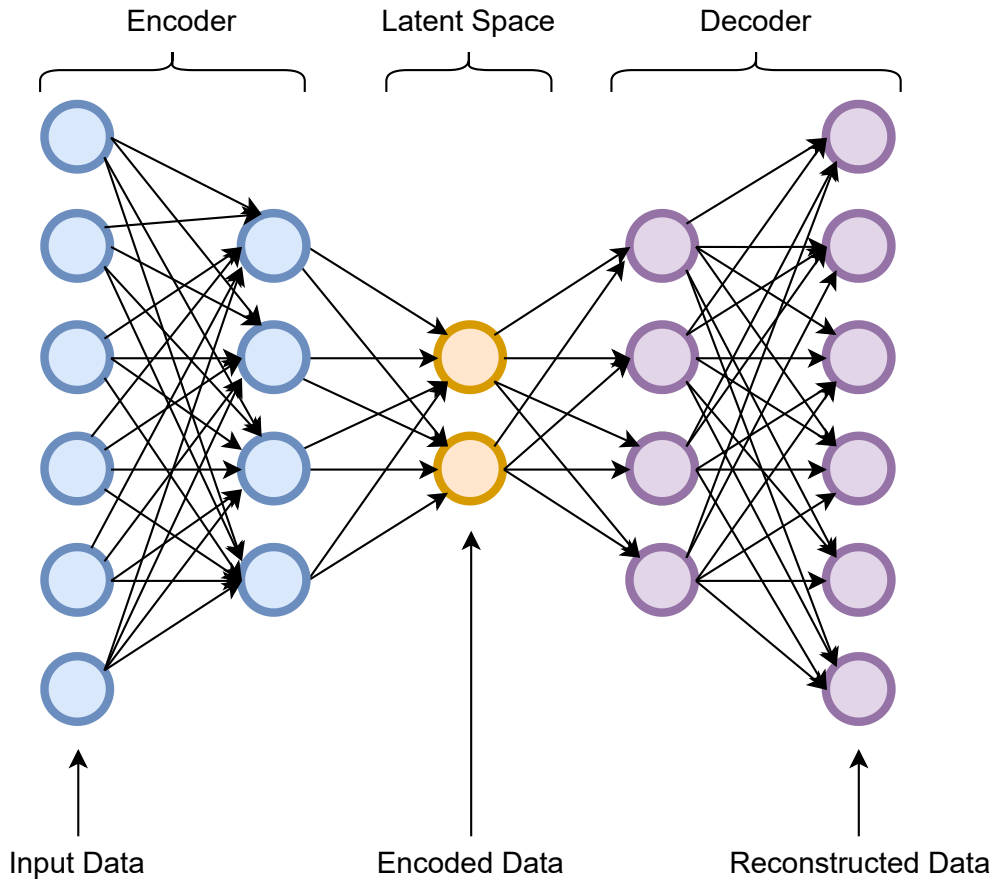


Figure 2.4: A diagram of an autoencoder consisting of an encoder and a decoder network. The encoder network maps the input image into a low-dimensional representation, while the decoder network reconstructs the original image from the low-dimensional representation. By minimizing the reconstruction error between input and output, the autoencoder can learn a compressed representation of the input data.
Figure inspired by Steven Flores [54].

Unsupervised learning has proven useful in computer vision, where it can aid in object recognition, detection, and segmentation, such as in medical datasets. By allowing correlations to be identified and grouped into clusters based on features or traits, unsupervised machine learning

can also be used for anomaly detection, where it can flag atypical data. This method is also often used in recommendation engines, such as those found in online stores, streaming services, or social media.

Semi-Supervised Learning and Hybrids

Supervised learning is often more accurate than unsupervised learning as it can learn from the knowledge provided by humans in the form of labeled data. This approach can also avoid computational complexity since the model does not need to train on irrelevant features for the specific task. However, the downside of supervised learning is that it requires significant human effort beforehand labeling the dataset, which can be time-consuming and costly, particularly when domain experts are required.

A hybrid solution that aims to leverage the best of both supervised and unsupervised learning is known as semi-supervised learning. This approach can be beneficial when there is too much data to label, either because the dataset is large or the cost of labeling is high. Semi-supervised learning typically assumes that data points close together in feature space share labels or have an underlying correlation factor in a lower dimension (manifold hypothesis/assumption). Methods using semi-supervised learning often achieve higher accuracy than those using only supervised or unsupervised approaches separately. This is because they do not need to discard data that is not labeled and can be trained using a researcher's *a priori* knowledge of the dataset or possible solution [55, 56].

Other approaches, such as active learning and weak supervision, can also often be used when a fully labeled dataset is unavailable for training. Active learning is a process in which a model can select input data about which it is unsure and ask a human or other expert for the correct answer. For example, such methods can label large data sets with services such as Amazon Mechanical Turk [57]. Samples are selected using a model that attempts to ask for samples believed to have the highest value for further classification. This can be achieved by first asking an expert for a sample from random samples and fitting the model to obtain a decision boundary that separates the classes. After that, the model can iteratively identify examples of high importance since they are near or at this decision boundary. These are the most uncertain examples, making them valuable

for the model to identify. Then the model asks the expert for samples near the boundary. The boundary is adjusted again, and the process is repeated until the model fits the data and satisfactorily separates the samples.

Weak supervision is a method that can be applied in situations where it is preferable to have a large number of sufficiently accurate examples rather than a smaller number that are completely correct [58]. This approach is often implemented by defining some rules or using a knowledge base in advance, which helps the model estimate the probability that an example belongs to a certain category. Weak supervision is often used in conjunction with transfer learning, which involves transferring knowledge from one area to another [59]. Transfer learning has shown promising results because generalizable knowledge of data can often be transferred, eliminating the need to learn from scratch on a new dataset. Transfer learning can be combined with other methods, e.g., supervised learning [60].

2.1.3 Deep Learning and Neural Networks

Neural networks, also known as Artificial Neural Networks (ANNs), are a machine learning method inspired by how the biological brain processes information and learns. Essentially, ANNs approach the goal of extracting information and constructing knowledge by forming a network of artificial neurons that can process input data. These artificial neurons, called perceptrons, are the building blocks of neural networks and are likewise inspired by biological neurons. In this representation, the inputs to the artificial neuron are synapses on the dendrites, the axon hillock inspires the activation function, and the output from the artificial neuron is the biological axon.

The artificial neurons process data by weighting the input data and a predetermined bias before passing it through an activation function, such as a sigmoid or Rectified Linear Unit (ReLU) [61] that produces the neuron's output. When the output of the activation function exceeds a certain threshold, the neuron is activated and sends data to the next layer. No further data is sent if the output does not exceed the threshold.

ANNs, often called multi-layer perceptron (MLP), are the cornerstone of deep learning, which refers to ANNs with more than three layers. The smallest deep network, therefore, has an input layer, two hidden layers

in the middle, and an output layer. These middle processing layers are called the hidden layer because it is hidden from the outside, which the input and output layers are not. An example of such a network with two hidden layers is shown in Figure 2.5 to see.

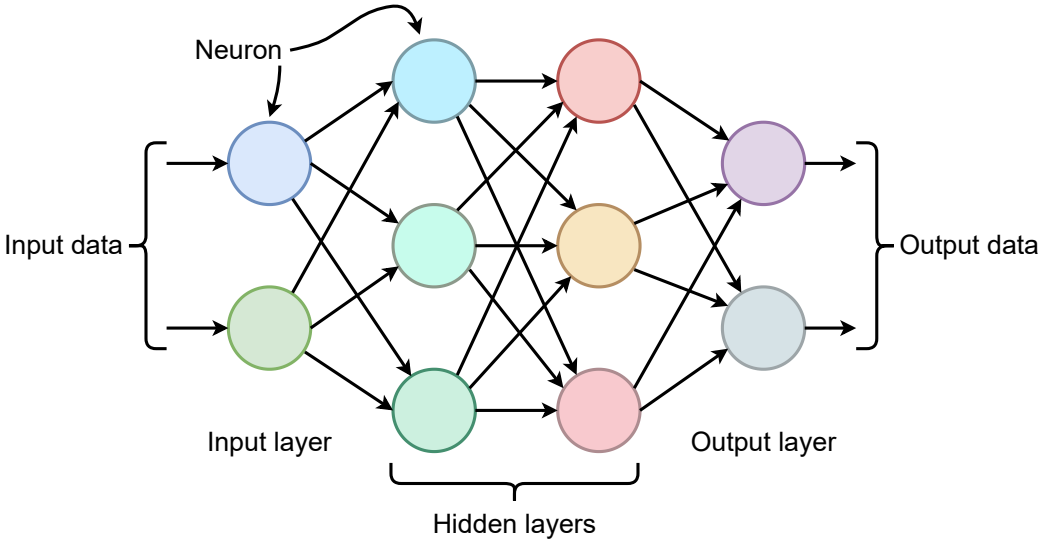


Figure 2.5: A fully connected deep artificial neural network with two hidden layers and two input and output neurons.

Most ANNs are feedforward, meaning that data is entered on one side, passed through the network, and produces a result on the other end. These networks are often trained using backpropagation, where the network output is compared to the ground truth, and the difference between the estimated result and the ground truth is used to adjust the network weights to minimize this difference. The backpropagation technique is also inspired by biological backpropagation, which occurs when a neuron generates an action potential that sends a voltage spike back to the dendrites. Other methods are proposed instead of feedforward combined with backpropagation, such as the proposed forward-forward algorithm by Geoffrey Hinton [62].

Deep neural networks have achieved remarkable achievements, mainly due to their ability to learn from structured and unstructured data. This allows them to learn and generalize knowledge given sufficient amounts and high-quality data [63]. Before computing power allowed deep networks, hand-made feature extractors were commonly used, using human-predefined features such as texture, shape, and color for object recognition tasks. However, when deep learning was first used in the Im-

ageNet Large Scale Visual Recognition Challenge (ILSVRC) competition [64], which is a competition identifying objects from the ImageNet dataset [65], it achieved great results as shown in Figure 2.6. This figure shows the methods and their classification error for each year the competition was arranged. It is clear that deep learning made significant strides in 2012 with AlexNet, which was proposed by Krizhevsky et al. [66]. These methods have improved significantly since then, largely thanks to improved methods and hardware. In 2015, deep learning even surpassed the human ability to classify images in ImageNet with the ResNet model proposed by He et al. [67, 68].

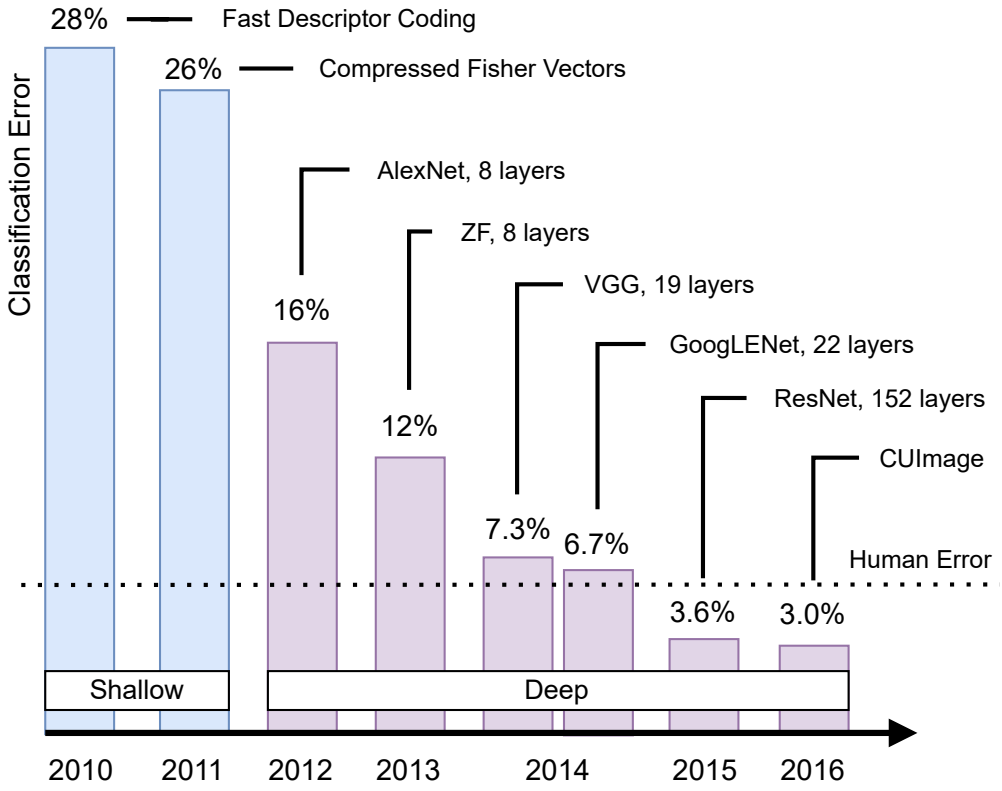


Figure 2.6: Overview of the ImageNet competition winners from 2010 to 2016. The figure shows how deep networks have significantly improved the networks' ability to guess the right label for the images in the ImageNet dataset. In 2015, the deep network ResNet [67] surpassed the average human ability to label these images correctly.
Figure inspired by Gordon Cooper [69].

2.1.4 Convolutional Neural Networks

Inspired by biology alongside regular neural networks, the computer vision method called CNNs is designed to mimic how the brain's visual cortex processes visual information. CNN builds on the idea and technique of neural networks and adds at least one convolutional layer. A convolutional layer allows information about neighboring entities to be obtained from the previous layer. The method has proven particularly effective in computer vision to process and analyze images and video because it can gather context from nearby areas where the convolution is centered. This can help the method extract contextual and semantic relationships that would otherwise have been lost. Both of the proposed methods presented in this thesis use a CNN, namely a VGG-16 model [8], to classify images and extract image features. Therefore, understanding how CNNs are constructed and their advantages and uses are relevant.

The convolution layer works by scanning small filters across the input image, computing dot products between the filter and the image pixels at each location, and creating a feature map. A convolution layer can be considered a flashlight beam aimed at an image. The beam of light allows the viewer can see areas of the image that are in the perimeter of the area where the center of the beam is directed. As the beam of light moves across the image, the viewer can comprehend more of what is in the image. An illustration of a convolution is shown in Figure 2.7.

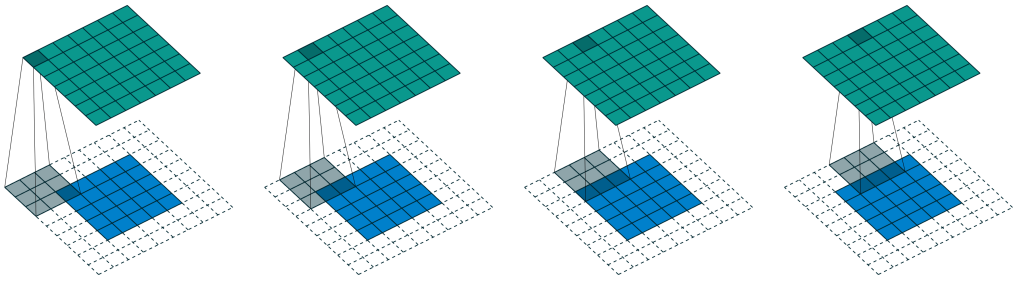


Figure 2.7: An illustration of a convolution where a 3×3 kernel is convolved with a 5×5 input matrix. The convolution illustrated has full padding and a one-step stride.
Figure by Vincent Dumoulin and Francesco Visin [70].

The first CNN that trained with backpropagation was first proposed by LeCun et al. in 1989 [71] and has since had a huge importance in making machines interpret images and videos. A major advantage of using a

convolutional layer is that it can compress information while preserving information and the context around the retrieved information. A very common CNN architecture is shown in Figure 2.8. Also seen in the figure is that a typical CNN has a fully connected neural network. A CNN often uses fully connected layers, unlike a normal neural network, because of the condensation of information with surrounding information, allowing for a network that can take advantage of using many parameters without overfitting.

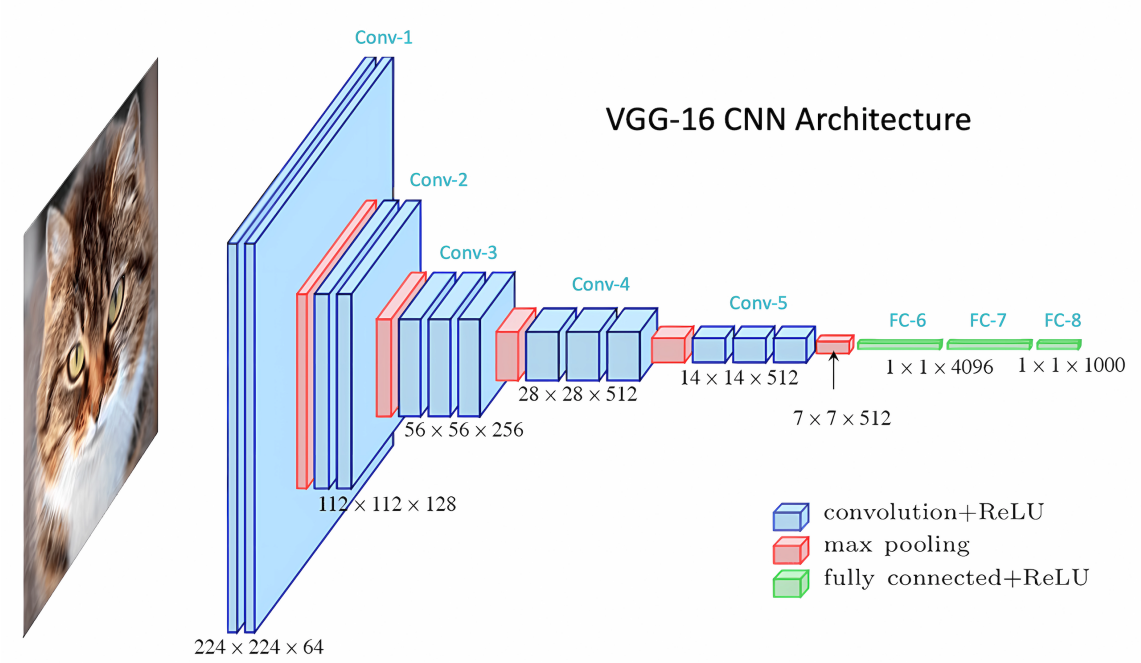


Figure 2.8: Example of a typical CNN architecture, here illustrated by the VGG-16 model proposed by Simonyan and Zisserman [8]. As seen in this figure, the first part of the network convolves the input image and extracts feature maps. These feature maps are then fed into a fully connected artificial network that classifies the image in a given class.
Figure by Satya Mallick [72].

The convolution layers can be applied to an input image as a series of filters, producing a series of feature maps that emphasize different aspects of the image. These filters have weights that are learned and adjusted during training and are able to recognize different types of patterns of increasing complexity and abstraction from the input image. The training process is similar to regular neural networks. It typically updates weights using backpropagation to minimize the loss function and improve the accuracy of the network on the training set. The first layers, closest to

the input image, generally recognize low-level features such as edges, lines, corners, texture, and color. As the convolutional layers process the input information, more complex features can be detected that build an abstraction on the information extracted from earlier layers. Convolution layers farther from the image can recognize more high-level features such as shapes, objects, and scenes. Overall, the convolutional layers allow the features of the input image to be analyzed hierarchically, with the early layers capturing low-level features and the later layers capturing high-level features.

CNNs are typically designed with multiple layers, similar to deep neural networks, and the last layers are typically fully connected. The fully connected layers take the output of the convolution layers, a set of feature maps, and convert them into a single vector of class scores that represent the probability that the input image belongs to each possible class. The main function of pooling layers in CNNs is to reduce the spatial dimensions of the feature maps generated by convolutional layers while preserving the most salient features of the input image. Pooling layers operate on small regions (e.g., 2x2 or 3x3) of the feature maps generated by the previous convolution layers and reduce their size by applying a pooling function, such as max pooling or average pooling, on each region. Max pooling selects the maximum value within each region as the representative value, while average pooling calculates the mean of the region. These operations effectively downsample the feature maps and help reduce the network's sensitivity to small spatial translations and variations in the input. Pooling layers can be inserted after one or more convolutional layers in a CNN. Their hyperparameters, like the size of the pooling area and the stride (i.e., the amount of movement of the pooling window), can be tuned to control the amount of downsampling and the size of the resulting feature maps.

Additionally, pooling layers can prevent network overfitting by reducing the number of parameters and the computation time required for training. The role of pooling layers in CNNs is not limited to downsampling. However, they can also help to capture translation invariance, i.e., the property that the learned features are invariant to small translations in the input. The translation invariance is achieved because the max or average operation selects the most prominent feature in a given region, regardless of its exact position. This property allows the network to collect

more general relationships in the input data, providing a more general understanding of the network. The overall benefits of pooling layers lie in reducing the spatial dimensions of feature maps while preserving their salient features, resulting in more efficient and robust image analysis.

In essence, the different layers of a CNN work together to extract and classify features in images and videos. This makes CNNs particularly effective for tasks such as image classification, object detection, and segmentation, where detecting and locating different features of an image is critical for accurate performance. CNNs have proven highly effective in image classification tasks, outperforming other types of neural networks and traditional machine learning algorithms. They have been used in various applications, including object recognition, facial recognition, and medical image analysis. With the development of larger and more complex networks and the availability of powerful hardware and software, CNNs remains an active area of research and development in machine learning.

2.2 Image Captioning

The methods proposed in this thesis use techniques to extract information from input images and present a text output. The method to make a model give a textual description to an image is often called image captioning. Image captioning is a field of research where the objective is to generate textual descriptions using natural language from the content of an image. This process uses computer vision and Natural Language Processing (NLP) to gather information from images and give a text that describes or analyzes the image's content. Image captioning is, therefore, a multimodal method that has excelled with the advancements in computer vision, helped considerably by CNNs and language models. The motivation for developing this field of research is that its advancements can be utilized in many applications. Image captioning can contribute to improving accessibility for the visually impaired, enhancing search and retrieval systems, assisting in indexing images and video, and making models that can use the combined information in both language and vision to gather information.

Image annotation methods typically include modern deep learning techniques, using CNNs to extract image features and Recurrent Neural

Networks (RNNs) [73], typically Long Short-Term Memorys (LSTMs) [74], to use these extracted image features to generate image captions. RNNs are a class of neural networks designed to work on data sequences, like natural language. LSTMs in particular are designed to combat the problem of vanishing gradients, making them better suited for longer sequences of data. An advantage of the multimodal nature of image labeling methods is that the CNN are able to capture spatial features within a given image. In contrast, the RNN can structure this information by including temporal dynamics and syntaxes using natural language to create a description that matches the image.

2.3 Attention Mechanisms

The method of attention in modern models was proposed by Bahdanau et al. [75] and is a technique that allows the model to search for the most relevant information located in different positions in a sequence. Recent advances in image captioning have seen the use of attention mechanisms, both in the vision and language modalities. Attention mechanisms have allowed the model to selectively focus on different parts of the image when generating captions. By selectively paying attention to different areas of the image, the model is able to capture the fine-grained details needed to generate accurate and descriptive captions. This has significantly improved the quality of captions, where attention-based methods often outperform traditional non-attention-based techniques [76].

The attention technique is motivated by nature and is inspired by cognitive attention mechanisms. The main advantage is that the attention allows the method to highlight some parts of the input data while giving other, less important parts a lower priority. Attention mechanisms aim to carefully learn which parts of the data are most important in context and prioritize those parts.

Attention-like mechanisms have been part of the research field of machine learning since the 1990s. Initially, it was introduced under names like sigma pi units, multiplicative modules, and hyper-networks [77]. The flexibility of attention-based techniques comes from their ability to learn which parts of the input data are most important in context.

Attention weights are most commonly learned during training through backpropagation and gradient descent. The weights are used to give

relevance scores to different words in context for being the following word in the text sequence. Attention can be derived in several ways, notably global vs. local and soft vs. hard. Global and local attention refers to different ways of weighting input features, where a global process weights all input features equally without prioritizing specific parts. Each input feature is considered when computing the output. In contrast, local attention refers to a more selective process in which the input features are weighted differently and are therefore prioritized when computing the output. This prioritized subset is usually determined by a region or window around the previous output or target position, borrowing ideas from how CNNs retrieves surrounding data from an image.

Soft and hard attention refers to the different methods of incorporating attention into a neural network. Soft attention is computed by a weighted average of the input features, with the weights learned during training and typically normalized to a value between 0 and 1. This weighting allows the model to consider multiple input parts simultaneously, gathering information from a more extensive range. On the other hand, hard attention selects a particular input feature to take care of at each time step, effectively forcing the model to make a discrete decision. This attention implementation can make the model less flexible but excel when the input is highly structured and easily segmented into distinct parts.

Some effective methods that use attention are LSTMs, Transformers, and Perceivers [15, 74, 78]. An overview of the Transformer model proposed by Vaswani et al. [15] can be seen in Figure 2.9. The use of autoregressive transformers and attention mechanisms have made it possible to make large language models, such as Bidirectional Encoder Representations from Transformers (BERT), Bidirectional Auto-Regressive Transformers (BART), and GPT [11, 79–83].

In the language part of the model, the attention mechanisms can also be used to selectively focus on a different part of the features extracted from the image. They can therefore generate the textual context that is most relevant to identify contents in the image correctly. This way, the model can describe the essential features of the picture or video using natural language. Some recent models have moved from CNN architectures to extract image features and solely rely on attention mechanisms to gather relevant information in the visual domain. Some of these models are Vision Transformer (ViT), alongside versions of BERT that incorporate

vision, like ViBert [84–86].

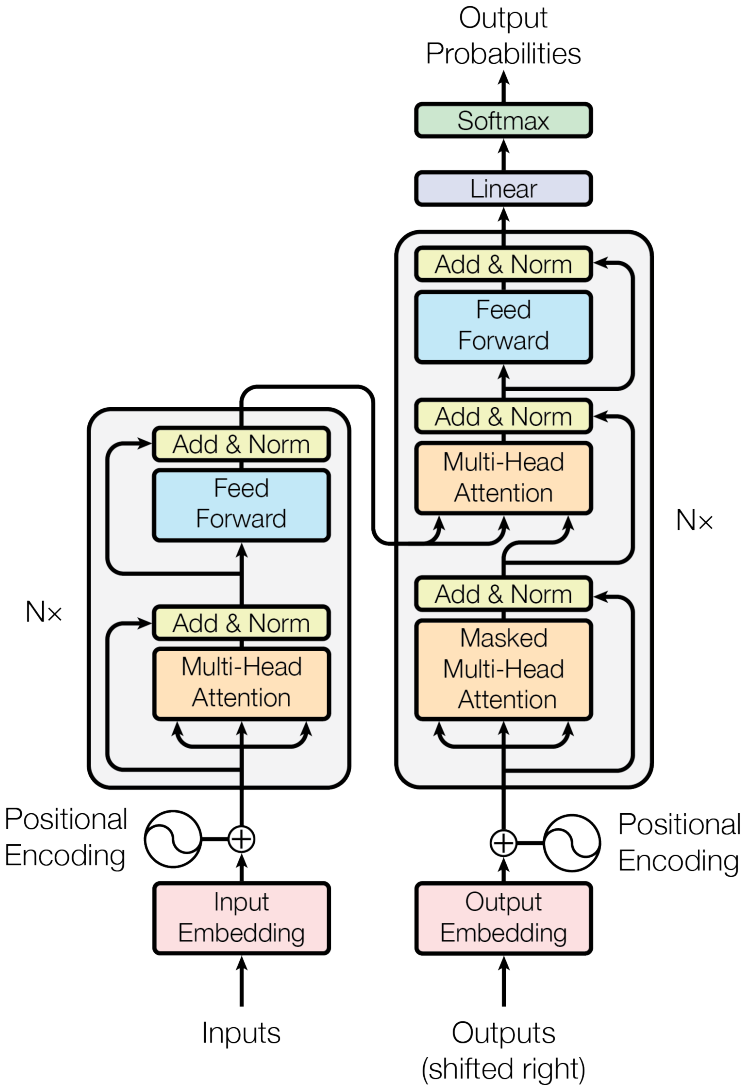


Figure 2.9: Transformer architecture proposed by Vaswani et al. Figure by Vaswani et al. [15].

2.4 Model evaluation

In the field of machine learning and AI, model evaluation refers to the process of measuring the quality and effectiveness of a trained model. These evaluations are essential in developing machine learning systems, allowing researchers and practitioners to determine their models' accuracy and generalization performance. Model evaluation helps to identify any weaknesses or shortcomings in a model's design or training

process and enables researchers to make improvements to enhance the model's performance.

Performance Metrics

Performance metrics are integral to machine learning models, allowing us to measure how well a model performs on a given task. These metrics evaluate the accuracy and effectiveness of a model's predictions by comparing them to the actual values. The choice of performance metrics depends on the problem being solved and the type of model being used. For example, in a binary classification problem, the model's accuracy can be used as a performance metric. In contrast, for multi-class classification problems, metrics like precision, recall, and F_1 score are more appropriate. In general, model evaluation metrics like precision and recall provide information about the performance of a model on a particular class, while accuracy and F_1 score provide an overall measure of model performance. Choosing appropriate performance metrics when developing a machine learning model is essential to ensure that the model's predictions align with the intended use case.

2.4.1 Precision and Recall

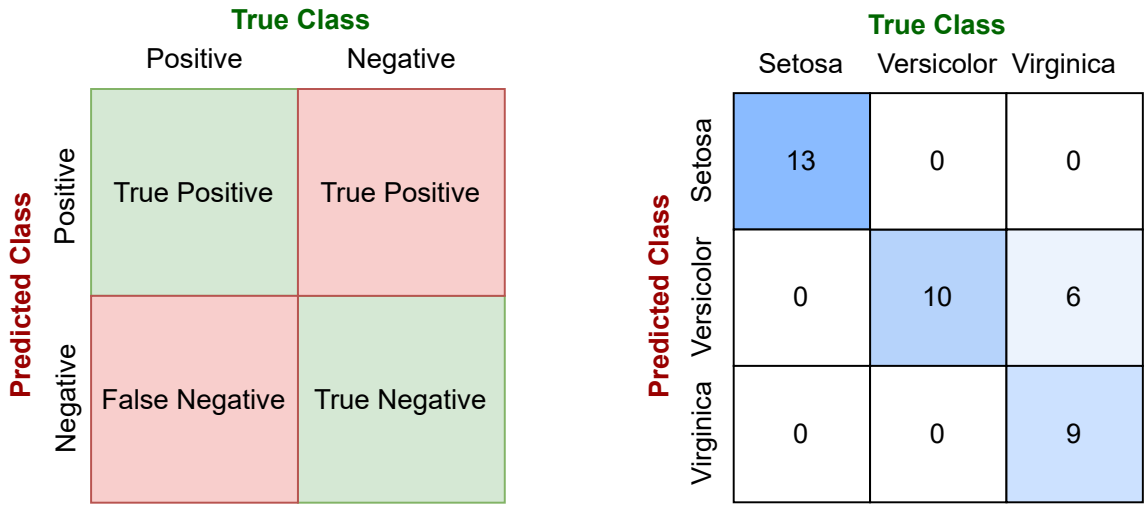
Precision and recall are two important metrics used in classification tasks to evaluate the performance of a model. Both precision and recall are based on the values in the confusion matrix, which is a table that summarizes the performance of a classification model. The confusion matrix contains four values: true positives, false positives, true negatives, and false negatives. True positives and false positives correspond to the model's positive predictions, while true negatives and false negatives correspond to the negative predictions. Figure 2.10 shows an example of a confusion matrix. The precision and recall are defined as follows.

- Precision: the ratio of correct answers among all answers proposed by the model, and it measures how precise the model's positive predictions are. The precision score can be calculated as follows:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.1}$$

- Recall: the ratio of correct answers proposed by the model among all possible correct answers, and it measures how well the model is able to identify positive instances. The recall score can be calculated as follows:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.2}$$



(a) Example of a confusion matrix of a binary classifier. A perfect classifier would have just true predictions.

(b) Example of a confusion matrix of a multi-class classifier. An adapted example from scikit-learn [87].

Figure 2.10: Example of confusion matrices, both for a binary (a) and multi-class (b) classifier.

2.4.2 Accuracy

Accuracy is a commonly used evaluation metric in statistics and artificial intelligence for classification models. It is a metric that is easy to interpret by humans because it measures the percentage of correctly predicted labels out of all the predictions made by the model. Accuracy is a valuable metric when the classes in the dataset are balanced, meaning that the number of instances for each class is roughly equal. However, when the classes are imbalanced, accuracy can be misleading. For instance, in a dataset with 95% of the samples belonging to class A and only 5% belonging to class B, a model that always predicts class A will have an

accuracy of 95% but will not be helpful in practice if the real-world data does not have the same distribution.

Moreover, accuracy does not account for false positives and false negatives, as seen in Formula 2.3. False positives occur when the model predicts a positive label for a sample that is negative, while false negatives occur when the model predicts a negative label for a sample that is positive. In some applications, such as medical diagnosis, false negatives may be more costly than false positives, and accuracy alone may not be an adequate metric for evaluating model performance. Therefore accuracy can be a helpful and easily understandable metric when there is no critical downside predicting false negatives and the classes in the dataset are balanced. When dealing with real-world datasets, the classes are often not balanced, and a more suitable metric can be the F_1 score.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Sample Size}} \tag{2.3}$$

2.4.3 F_1 Score

The F_1 score is a statistical metric used to measure a model's performance on a dataset. It evaluates the performance of a binary classifier, which is a classification system that makes predictions for two possible classes, for example, positive and negative. The F_1 score can be calculated using the formula in Equation 2.4.

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

In the context of question answering by an LLM, the F_1 score can be used to evaluate the accuracy of the model's responses to questions. Specifically, the score measures the harmonic mean of the model's precision and recall.

2.4.4 Perplexity

Perplexity, often referred to as *PPL*, is one of the most common metrics for evaluating language models. In information theory, perplexity measures how well a probability distribution or model predicts a sample. Intuitively, perplexity means being surprised, and in practice, it measures how surprised the model is when it sees new data. The lower the perplexity

score, the better the training. Perplexity is usually only used to determine how well a model has learned the training set.

At its core, a language model is a probability distribution over a set of words, known as the model vocabulary. Considering all of its previous words, the model indicates the probability that a given word will appear in the vocabulary. Usually, the word with the maximum likelihood is selected as the next predicted word in the sequence.

As an evaluation metric, perplexity can be used to compare probabilistic models, and low perplexity indicates that the probability distribution is good at predicting the sample. This probability can be calculated by multiplying a sequence of conditional probabilities for each word by its previous words, which indicates the probability of this sequence. Given a tokenized sequence $X = (x_0, x_1, ..., x_t)$, the perplexity of the sequence X is given in the Formula 2.5.

$$\text{Perplexity}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta\left(x_i \mid x_{<i}\right)\right\} \tag{2.5}$$

In this formula, $\log p_\theta\left(x_i \mid x_{<i}\right)$ is the log-likelihood for the token with index i, given the previous tokens $x_{<i}$.

2.5 Frameworks

This section briefly describes the most important external frameworks used in this thesis.

2.5.1 TensorFlow

TensorFlow is the framework used in the FLEX-VQA method proposed in this thesis. Here the framework is used to train the FLEX model to find connections between images and captions. A more detailed description of the implementation is discussed in Chapter 3.2.

TensorFlow is an open-source machine learning framework developed and maintained by the Google Brain team. It was first introduced in 2015 and has since become one of the most widely used frameworks in machine learning. TensorFlow's core strength lies in its ability to handle large-scale machine learning models easily. It provides a highly optimized execution engine that efficiently computes complex models. This optimization is

achieved through dataflow graphs, representing the computations as a directed graph of nodes and edges. Each node in the graph represents an operation, while the edges represent the data flow between operations. This architecture allows for parallelism and efficient computation, making it an attractive choice for large-scale machine learning tasks.

One of the key advantages of TensorFlow is its versatility. It supports various machine learning tasks, including classification, regression, clustering, and reinforcement learning. It also supports deep learning models, such as CNNs, RNNs, and transformers.

2.5.2 Text Tokenization

Large language models such as GPT-3 [83] and BERT [79] are trained on massive amounts of text data and can generate coherent and contextually relevant sentences, paragraphs, and even longer texts. However, feeding raw text data to such models can be computationally expensive and lead to suboptimal results due to high input variability. Therefore, text preprocessing techniques such as tokenization convert raw text into a form the model can more easily process.

Tokenization is the process of splitting text into smaller units, called tokens, such as words, subwords, or characters. These tokens are then assigned a unique integer identifier and are often converted into fixed-length sequences, which can be fed into the language model. This process allows the model to handle text data more efficiently, reduces input variability, and helps to improve the training and inference times.

Using a tokenizer specific to the language model used is crucial because it ensures that the tokens are consistent with the vocabulary and encoding scheme of the model. For example, using a tokenizer that splits words differently than the one used during the pre-training of the model can lead to inconsistencies and decreased performance.

Various tokenization techniques are available, such as byte-pair encoding (BPE) [88, 89], WordPiece [79], and SentencePiece [90]. BPE is a subword tokenization method that progressively merges the most frequent pairs of characters in a corpus to create a fixed-size vocabulary. WordPiece is a similar technique that uses a predefined list of subword units to tokenize words. SentencePiece is a more flexible method for creating a custom vocabulary that can handle rare and out-of-vocabulary words.

Each of these techniques has its advantages and disadvantages. BPE and WordPiece are efficient and widely used, but they can result in a large vocabulary size and require additional processing steps to handle out-of-vocabulary words. SentencePiece can manage rare and out-of-vocabulary words better but can be computationally expensive due to their adaptive nature.

This work uses a tokenizer built on top of SentencePiece, adapted and trained specifically to the LLaMA model. A more detailed explanation of this tokenizer can be seen later in subsubsection 3.2.2, which details the implemented method.

2.6 Related Work

In this section, some relevant work for this thesis will be presented. Knowing the main takeaways from these works will help understand the context and assessments that will be made when implementing the methods presented in the next chapter. The topics covered in this chapter are different methods to create explainable machine learning models and the development of large language models that lead to the model used in this work.

2.6.1 Explainable AI (XAI)

The rapid growth of AI technologies has led to an increasing demand for transparency and interpretability in machine learning models. While these models have demonstrated impressive performance in various applications, their lack of transparency and interpretability has raised concerns about their reliability, accountability, and potential biases [91, 92]. As a result, the field of XAI has emerged to address these challenges by developing techniques and tools that enable humans to understand how AI systems work and make decisions.

The field of XAI is working on solving the trade-off between performance and explainability. Some approaches specialize in explaining specific architectures, called model-specific. Meanwhile, others, called model-agnostic, try to explain models of different architectures, exploiting inherent properties in neural networks and statistics. Examples of inherently explainable models include decision trees, Bayesian classifiers, logistic re-

gression, linear models, and k-NN [49, 50, 93]. These algorithms are interpretable and, as a result of this, more explainable by design. This interpretability results from their internal structure, and computations follow clear rules or formulas that are manageable to comprehend. However, models such as deep neural networks can work better than less complex methods on larger datasets. Because of their complex structure, with many hidden layers and weights trained on large datasets, it is difficult, if not impossible, for humans to understand what the model evaluated when choosing a prediction. Researchers in XAI have developed model-specific and *post-hoc* model-agnostic techniques to understand better these complex methods to explain the underlying model prediction. The term post-hoc comes from Latin and means *after this*. This thesis uses the term to describe a method applied after a machine learning model is developed. These methods try to bridge the gap between high accuracies and explainability. In theory, this allows us to use the model best fit for the task, regardless of complexity, without the expense of not understanding the model's predictions.

Different explanations try to give additional insights into the predictions. A local explanation looks at one specific model decision and explains what was important in the input data to provide this prediction. On the other hand, a global explanation looks at the whole model's process of making decisions and sees how the different attributes contribute to making a decision. These global explanations can also be built from multiple local faithful explanations. A third type is contrastive explanations that utilize local features to explain the difference between instances. This allows insight into the model's inner workings on a more global scale based on local predictions.

One typical disadvantage of model-agnostic explanations is that their insights and descriptions are not always faithful to the underlying model they try to explain. This can happen if an explaining model is trained to look at the underlying model's input, inner workings, and output and learns the correlation between them. The problem is that correlation does not imply causation, and the explaining model can give a deceiving explanation that can look correct at first. Developing explanatory methods that find causations rather than correlations is an ongoing research topic.

An overview of influential methods proposed in XAI is presented in this subsection.

LIME

In pursuing a method that helps the explaining model to be locally faithful to the underlying model, Ribeiro et al. [94] proposed LIME. This model-agnostic method explains any method by learning an interpretable, less complex model locally around a specific prediction. This approach assures that the explanation is locally accurate and represents the actual inner workings of the model. In the same paper, they also introduce a method for explaining the global attributes of the model by framing the task as a submodular optimization problem. The technique is called SP-LIME (Submodular Pick LIME). With this approach, they can achieve global explanations that are locally accurate and faithful to the underlying model in a non-redundant way. LIME will be used later in this thesis as an explanation method adapted to an LLM.

SHAP

Making non-redundant explanation features faithfully and efficiently is not easy. Lundberg et al. [95] proposed a unified framework for interpreting predictions made by the underlying method. This framework, called SHapley Additive exPlanations (SHAP), assigns each feature a value of importance for a specific prediction. The framework utilizes the class of additive feature attribution methods and estimates the Shapley values [96] from cooperative game theory for that prediction. With this approach, they achieve more effective explanations to compute and have better consistency with human intuition than previously proposed methods.

Grad-CAM

Visually explaining features that contributed to an image prediction can be essential in gaining trust in a model. Selvaraju et al. [97] proposed a technique for making CNNs more explainable and transparent by producing visual explanations for the underlying model. The method is called Gradient-weighted Class Activation Mapping (Grad-CAM), a generalization of CAM proposed by Zhou et al. [98], and uses the gradients in a single backtrack of any target concept. Therefore it does not need any architectural changes or retraining of the underlying prediction model to produce a localization map highlighting the crucial regions in the image for predicting the given concept, often called a saliency map in computer vision.

FLEX

While visual-only explanations can give the user insight into which areas of the image were essential in making the decision, they tell little to nothing about why those regions were important. On the other hand, linguistic descriptions give the user an essential understanding of the model's evaluation when predicting. Wickramanayake et al. [3] propose Faithful Linguistic Explanations (FLEX) to merge saliency maps with locally accurate linguistic descriptions. In this approach, they look at the gradients through layers and identify the most critical activations in the single decision. The advantage of looking at different layers is that alongside getting an explanation that is faithful to the underlying model, they also extract features the CNN identifies at each layer. A CNN may represent high-level concepts, like a "car", at the last layer while identifying features such as texture and color at earlier layers. Using the activations at all essential layers, LIME achieves an image caption that explains all the essential parts of the prediction using sentences. FLEX maps words to neurons in the CNN by looking at high activations of that neuron combined with a word from the caption during the training. For this, they are using a CNN and two stacked LSTM [74] cells. The FLEX framework is used as a basis for one of the two proposed methods in this thesis. A more detailed description of the specific implementation for this work is discussed later in Chapter 3.1.

Visual Question and Answering (VQA)

In order to make the linguistic abilities of computers more robust, Agrawal et al. [99] proposed a new dataset called VQA. This dataset provides images from the COCO dataset [100], paired with free-form, open-ended questions and answers corresponding to the content of the images. These questions and answers target different areas of the images, including underlying context and background details. This dataset aims to make models that can learn multimodal visual and linguistic domain knowledge to get a more general and complete understanding of the world. Models that have done well in this dataset are frequently made up of CNNs, to acquire the visual knowledge, combined with an RNN for linguistic understanding.

Even though VQA is not strictly XAI, it is still relevant regarding transparency to the user. VQA is an important research area because it

provides AI systems with human-readable explanations of their decision-making processes. Because of this ability to answer questions regarding images, some have called this the visual Turing test [101]. By providing a natural language explanation of why a specific answer was generated for a particular question about an image, VQA models can help improve AI systems' transparency and interpretability. This can be particularly important in domains such as healthcare and finance, where trust and transparency are critical for ensuring that AI systems are making reliable and safe decisions. Therefore, it is important to continue developing and improving VQA models to provide accurate and interpretable answers to questions regarding images. Both the methods implemented and investigated in this work will be based on VQA, using text to describe the contents of images utilizing different approaches.

2.6.2 Large Language Models (LLMs)

LLMs are neural networks with billions or more parameters trained by self-supervised or semi-supervised learning on large amounts of text. They originated around 2018 and have performed competently on a variety of tasks. LLMs are typically general-purpose models that excel in various roles, with their performance and range of capabilities depending on the number of resources devoted during training. These models demonstrate considerable general knowledge about the world and can learn associations that make the model "memorize" numerous facts and contexts during training.

LLMs are pre-trained on large text datasets and can be characterized by four parameters: the size of the model, the size of the training dataset, the training cost, and the post-training performance. These variables are related by simple statistical laws called scaling laws.

LLMs serve not only to teach AIs human languages but to understand proteins, write software code, and help students. These models are trained with vast amounts of text fed into the AI algorithm through unsupervised learning, allowing the model to find valuable connections in the language. Through this method, an LLM learns words, their relationships, and the concepts behind them. LLMs can also be tailored for specific use cases, including through techniques such as fine-tuning or prompt tuning, which feed the model small bits of data that must be focused on to train it for a

specific application.

However, a disadvantage of LLMs is that they can suffer from a phenomenon called hallucinations. Generative models can hallucinate because they contain vast amounts of data and organize the information in an unsupervised way. These models tend to make self-confident claims about facts not justified by their training data, which appears plausible but is not factually correct. Because of their size, they can also be challenging and computationally expensive to interpret. An ethical concern about the size of these models is that they are also computationally intensive during training and inference since they are trained on large datasets. As a result, these models have a larger carbon footprint than smaller models. However, there are ways to make these models smaller and faster, which are discussed in this chapter and also provide an overview of the development of large language models.

BERT

Bidirectional Encoder Representations from Transformers (BERT) is a large-scale neural language model that has made significant contributions to the field of NLP [79]. BERT is a pre-trained model that uses a transformer-based architecture that has grown in popularity in recent years due to its success in several NLP tasks. Introduced by Google in 2018, BERT is trained on a large corpus of text using an unsupervised learning approach. The model is pre-trained on a task called Masked Language Modeling (MLM), in which a small percentage of the words in a sentence are masked randomly, and the model is trained to predict the original word based on the context of the sentence. This method can be viewed as a "fill in the blanks" task, often called a cloze test [102], of the training sentences, keeping some words invisible to the model during training and helping the model generalize to new and unseen data. In addition, BERT is trained on an NSP (Next Sentence Prediction) task, similar to the GPT models. The model is given pairs of sentences and asked to predict whether the second sentence continues the first.

The main innovation of BERT is its ability to understand the context and provide contextualized word embeddings. Unlike previous word embeddings, which were static and did not change based on the context of the sentence, BERT can offer different embeddings to the same word depending on its context.

BART

Lewis et al. at Facebook presented an LLMs named Bidirectional Auto-Regressive Transformers (BART) [80]. Like BERT, it uses a transformer-based architecture and is trained on a large corpus of natural language. However, unlike BERT, BART is unique because it integrates a bidirectional and auto-regressive architecture, which makes it well-suited for text generation and summarization tasks. This model is a pre-trained sequence-to-sequence model that can be fine-tuned for various natural language processing tasks. BART is designed to handle auto-regressive, non-auto-regressive, generation, and comprehension tasks. The model is pre-trained on a large corpus of text using a denoising autoencoder objective, which requires the model to reconstruct original text from corrupted versions. Using a denoising autoencoder in pre-training helps reduce hallucinations by training the model to distinguish between real and fake input. The authors demonstrate that BART outperforms several state-of-the-art models on various natural language generation and comprehension tasks, including summarization, question answering, and text classification. They also show that BART can be fine-tuned with relatively little data and can generalize to new domains. However, instead of using Masked Language Modeling and Next Sentence Prediction like BERT, BART is pre-trained using a denoising autoencoder (DAE) objective.

The DAE objective involves corrupting an input sequence by randomly deleting or swapping tokens and training the model to reconstruct the original sequence. This approach allows BART to handle more complex tasks such as text summarization, sentence generation, and machine translation.

GPT

GPT (Generative Pre-trained Transformer) is a set of LLMs developed by OpenAI [81]. Like BERT and BART, GPT is a transformer-based model but uses only an autoregressive architecture. It is pre-trained on large datasets and tuned for specific tasks. When GPT-2 was released, it was trained on a much larger dataset with significantly more parameters than GPT-1 [82]. This allowed it to generate more coherent and realistic text than its predecessor. GPT-3 is the third model in the GPT series and was released by OpenAI in 2020. It was trained on an even larger dataset than GPT-2 and had even more parameters, making it one of the largest

language models. GPT-3 could generate even more readable and human-like text than its predecessors and perform various NLP tasks without explicit training [83]. At the time of writing, the last published GPT model was GPT-4, which is still an autoregressive model but also includes multimodality [11] by making it able to interpret images. To reduce the effects of hallucinations on a generated output, GPT implements a combination of methods such as filtering and sampling.

These models have proven to be versatile and powerful tools for NLP. GPT models have significantly influenced the NLP field, and many researchers and developers have used them as a starting point for their projects. Fine-tuned chatbot versions of GPT-3 and GPT-4 have been made available for public interaction under the name ChatGPT [12].

LLaMA

The AI department of Meta, previously Facebook, released a modified architecture for an LLM called LLaMA [6]. These models are created to compare to other LLMs, such as GPT-3 [83], Chinchilla [103], or PaLM [104], while keeping the number of parameters considerably smaller. In the paper where Hoffman et al. propose the Chinchilla model, they also present insight into how models scale the best in conjunction with the size of the available dataset. The authors of LLaMA use this insight to make the model with fewer parameters perform well by training it on more tokens. The dataset that the LLaMA model is pre-trained on is publicly available and disclosed, which makes it compatible with open-source. An overview of the data the LLaMA models are trained on can be seen in Table 2.1. However, even though these datasets are publicly available and the pre-training distribution is disclosed, the public has yet to have access to the complete dataset actually used. Some sources have been scraped from the web by the authors, and the specific criteria used are not known at the time of writing.

The architecture of LLaMA is based on the transformer [15], with various improvements inspired by other LLMs. Some of these improvements are pre-normalization, inspired by GPT-3 [83], which improves the training stability by normalizing the input of each transformer sub-layer instead of the output. Like the PaLM model, they also use a SwiGLU as the activation function, first introduced by Shazeer at Google [105], instead of ReLU. Shazeer showed it to improve the perplexities of transformer-

LLaMA Pre-trained Data

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 2.1: Overview of the datasets, their sampling proportion, number of epochs trained with 1T tokens, and their disk size. These datasets are all publicly available and are compatible with open sourcing.

based models. To make the self-attention mechanisms in the transformer position-agnostic, the authors implement a method called Rotary Position Embedding (RoPE) introduced by Su et al. This method allows both flexibilities of sequence length and faster convergence in fine-tuning compared to normal self-attention. With the 13 billion parameter version of LLaMA, the authors show that this model outperforms the GPT-3, with 175 billion parameters, on several evaluation metrics. Because the LLaMA model is ten times smaller than GPT-3, it can also be run on a single GPU. These steps towards smaller, capable models benefit both the carbon footprint and inference compute budget and permit democratizing LLMs by making a model that can run on consumer hardware.

Alpaca

The work in this thesis uses this Alpaca model to interpret images. More of how this model is implemented is discussed later in Chapter 3.2.

With Taori et al., Stanford University released an open-source fork of this LLaMA model, with some modifications, called Alpaca [4, 5]. The Alpaca model is a fine-tuned model, with a 7B LLaMA model as the base model, trained on 52 thousand instruction-following tasks generated by OpenAI's text-DaVinci-003 model [106] using techniques from the Self-Instruct paper proposed by Wang et al. [107]. Figure 2.11 shows an overview of the Alpaca training procedure and how the dataset was built. By using text-DaVinci-003 to generate instructions, the team was able to train the model and generate the dataset for a significant cost reduction

compared to traditional methods, totaling only $600, with $500 using OpenAI API to create the dataset and $100 to rent 3 hours on 8 Nvidia 80GB A100 GPU cards. Reducing the cost of training an LLM, comparable to models much more expensive to develop, helps democratize these models and make them available to more people for less cost and energy.

To address the ethical issue of not knowing whether a text is generated by an AI, the team also implemented the method proposed by Kirchenbauer et al. [108]. This method embeds "green" or marked tokens into the generated text, which are invisible to humans but can be detected by an algorithm analyzing a short span of these tokens. The authors argue that the watermark can be embedded with negligible impact on text quality and can be detected with an efficient open-source algorithm without access to the language model API or parameters. The watermark works by choosing a random set of green tokens before generating a word and using a soft watermarking rule to encourage the input of green tokens during sampling. The authors propose a statistical test to detect the watermark with interpretable p-values and derive an information-theoretic framework to analyze the sensitivity of the watermark.

Given the need for less computational resources, leading to a reduced carbon footprint, and the ability to disclose the generated text as computer-generated, the Alpaca model represents a favorably ethical and transparent option for users of a system compared to other LLM. As such, it is an appropriate choice as a starting point for this thesis. This model has many of the benefits that LLMs can provide, like knowledge of vasts amount of data, while still being able to customize it to a specific task through fine-tuning. This process enables the model to adapt to a particular assignment while retaining the knowledge acquired from its initial training data.

2.7 Problem and Application

The increasing use of AI has led to significant advances in various areas. However, understanding their decision-making processes becomes increasingly difficult as AI systems become more complex and opaque. The need for XAI and more transparent machine learning models has become imperative to address this problem, especially in applications where the consequences of wrong decisions can be severe, such as

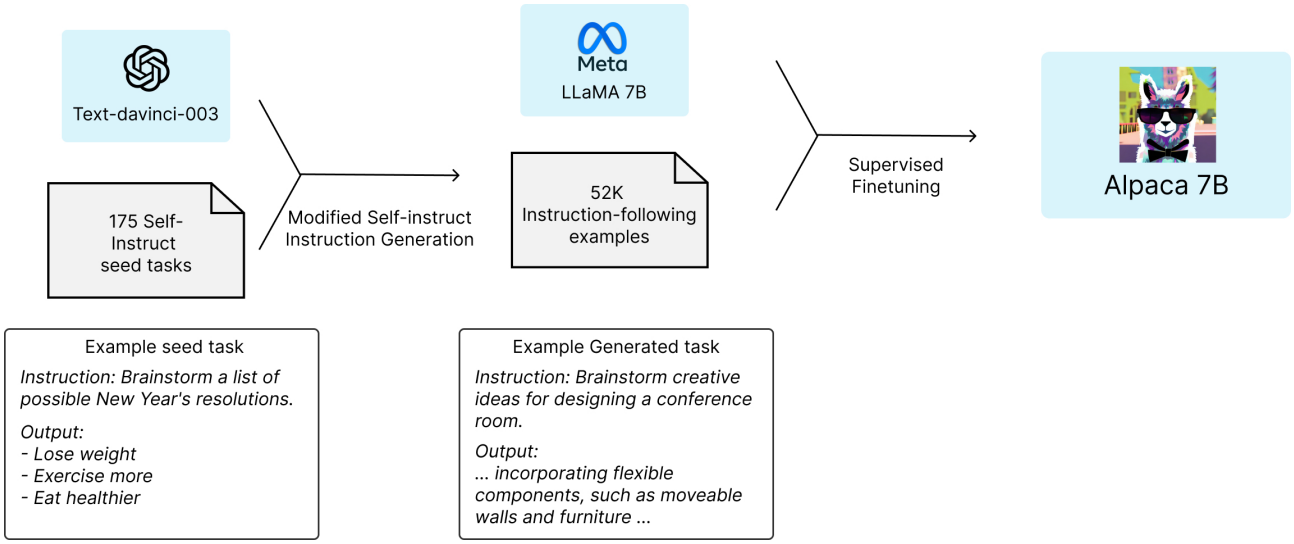


Figure 2.11: Overview of the Stanford Alpaca training procedure. Figure by Taori et al. [4]

healthcare, finance, and autonomous vehicles. This chapter aims to provide an overview of the problem and application of XAI, which is crucial for building transparent and trustworthy AI systems.

2.7.1 Problem

Although deep networks have made significant positive achievements in areas such as object detection [109–112], image annotation and captioning [113–116], their complexity makes it more difficult to understand why these networks predict what they do. People, both researchers and users, of these systems need to be able to know how these black-box algorithms work to gain confidence [117–119], improve the models and apply these networks in new ways and domains [120–124]. Researchers and model architects can understand at a higher level how information flows in the network compared to everyday users by using their technical insight. However, as the architecture grows more profound and complex, and the training datasets are getting larger, it can be challenging to understand which parts of the input data contributed to making the decision [125].

Interpretablity vs. Accuracy

The models with the highest accuracy for large data sets often have such complex architectures that even domain experts have difficulty interpreting their decisions [126]. Meanwhile, smaller, less complex

architectures often have lower accuracy and generalize worse than their more complex counterparts. An example of this was shown in a case study to predict the risk of death for patients with pneumonia to help medical staff prioritize [127]. The researchers found the most accurate model to be a neural network, outperforming less complex models such as logistic regression. A rule-based system was also evaluated, and while this is a simpler model than neural networks, it is interpretable by design. This rule-based model to investigate the underlying dataset showed that a patient suffering from pneumonia and asthma had a lower probability of dying than only having pneumonia and was, therefore, less important to treat. The model drew this conclusion because patients in the training set with both pneumonia and asthma were usually prioritized first, was given medical treatment, and therefore had a higher survival rate [128]. Because of this insight from the rule-based method, more complex models, such as neural networks, were concluded to be too risky in real-world decisions.

In pursuing models with higher accuracies, the primary way to achieve this is with even more complex models and larger training datasets [129]. This brings the trade-off of an interpretable and explainable model vs. a more accurate model [130] to the forefront of discussion.

Explainable AI

XAI is an interdisciplinary research field that aims to make AI models more transparent, interpretable, and accountable to human users. It combines techniques from machine learning, human-computer interaction, cognitive science, and other related disciplines to develop methods and tools for explaining the behavior and decisions of AI models. The need for XAI arises from the fact that many machine learning models, especially deep neural networks, are often viewed as black boxes, meaning that their internal workings and decision-making processes are intricate for humans to comprehend. As a result, these models' lack of transparency and interpretability can create distrust among users, limit their adoption, and raise ethical concerns, especially in high-stakes applications such as healthcare, finance, and criminal justice.

XAI can either be used to design models that are interpretable by design or adapt explanation methods to models already developed. The goal of XAI is, therefore, to build models and methods that can achieve great results without a reduction in transparency

2.7.2 Application

With better explanations that are intuitive and faithful to the underlying prediction model, humans can be more precise when improving the model. This also gives the ability to use the model with confidence that the prediction is based on correct decisions.

When machine learning methods are utilized as a tool in the real world, it is essential that everyone in the process of using the tool can trust the model. In a medical setting, both the doctor and the patient must have confidence that the conclusions drawn are based on a reasonable and trustworthy basis. If the model is correct, but the clinician does not trust the underlying model or the patient has no explanation for the model's conclusion, the model is not being used as intended and is, therefore, a useless tool.

This is also the case in other domains, such as finance. Here the bank can utilize large models that look at the loan applicant using big data. The bank can profile the customer alongside fellow citizens in the same demographic and use a model to predict whether or not that customer should receive a loan. In this case, finding all biases in the data set can be difficult, as it can be challenging to distinguish correlation from causation in demographic analysis [131]. Here the bank must have an explanation alongside the predicted output of the loan to see better if the model made a trustworthy decision.

One advantage of better understanding the model's evaluations in a prediction is that it can highlight biases in the dataset. If these biases are known and understood, they can be combated. Ribeiro et al. [94] showed that it could be hard to discover biases when the predictions are correct without knowing the reasoning. They experimented with a model trained on images of huskies and wolfs and first presented the model's prediction without explanations to participants. The participants were then asked to determine if they trusted the model. Thereafter the model's explanation for the predictions was presented, and participants again had to tell if they trusted the model now. In both instances, the participants were also asked if they thought snow was a potential feature of importance. An image of a husky misclassified as a wolf can be seen in Figure 2.12, alongside the regions important in making the decision. The results can be seen in Table 2.2, and it can be seen that a considerable amount of the participants lost trust in the biased model when they were

presented with the explanation. They then noticed that the decision was based only on whether the snow was visible in the image. This shows that models with an intuitive explanation are more likely to gain the trust of their users. Trustworthy explanations open the ability for users who do not have insight into the making of the model to still able to detect biases in the dataset and improve the model.

	Before explanation	After explanation
Trusted the biased model	10 out of 27	3 out of 27
Thought snow was an important feature	12 out of 27	25 out of 27

Table 2.2: Overview over the trust by participants in the *Husky vs. Wolf* experiment by Ribeiro et al. [94]. The majority lost trust when explained that a classifier considered snow in the background the most important feature when classifying images of huskies and wolves.

Figure 2.12: Husky classified as a wolf, alongside an explanation of what the model considered important.
Figure by Ribeiro et al. [94].

Today's machine learning models are already better than humans in domain-specific tasks, like chess [132]. Silver et al. [133] proposed the reinforcement learning algorithm AlphaZero, which learns to play chess, shogi (Japanese chess), and Go, only by playing with itself. This method does not get any domain knowledge except the game rules and achieves

performance better than humans. Because this algorithm has not seen humans play these games, it does not have human biases and playing flaws in it, and professional game players can learn new moves and techniques by playing with it. As performance better than humans is a goal many researchers strive for when it comes to machine learning, it is more important than ever that these algorithms give explanations on what they do and why so that humans can learn new aspects never before thought of, as well as detect flaws that restrict performance, both in humans and machines.

2.8 Summary

This chapter provides an overview of the background and motivation for using the concept of XAI in various applications. The discussion focuses on the challenge of understanding decision-making processes in complex AI systems and the necessity for more transparent machine learning models, particularly in applications where the consequences of erroneous decisions can be severe.

The chapter also delves into the history of AI and its two main paradigms: symbolic and connectionist AI. Moreover, it lays out various machine learning techniques, including supervised, unsupervised, and semi-supervised learning, as well as deep learning and neural networks.

The key takeaways from this chapter are:

- The importance of XAI in building transparent and trustworthy AI systems for real-world applications.
- The diverse techniques used in machine learning, why they are used, and how they differ.

The topics discussed in this chapter provide insight into the importance of designing systems transparently. An understanding of the data and models used helps researchers build better and more appropriate systems, and users can better understand why those systems predict what they do.

This work will use a combination of a CNN, an LLM, and XAI to conduct its experiments. These experiments will investigate further how these methods can be understood and explained.

The following chapter will therefore introduce some methods that address some of the challenges presented in this chapter. Two different approaches are presented, and both have multimodal capabilities. Both models are based on the VQA task, where one model uses a traditional VQA architecture, with an explanation method that explains visual features using text. The other model also describes images using text but is based on an LLM. The next chapter is, therefore, a summary of methods used to answer the research goals of this thesis.

Chapter 3

Methodology

In this chapter, the two methods proposed in this work are presented in more detail. The overall goal of these two methods is to explore how different machine learning models can implement explanatory methods in various domains that still provide value to the user.

Different model architectures represent data in separate ways. This chapter highlights how insights into these representations can give a researcher or user a new understanding of the data and models used. Both models presented in this chapter have multimodal capabilities, specifically in visual and linguistics. The way they achieve this multimodality differs.

The first model uses a classical VQA architecture, with the addition of the FLEX framework. Using this framework, the model can label feature maps in the CNN to get information from the visual domain translated into natural language. The explanation originates, therefore, from the visual domain, translated into text. Although no results from this method are examined due to hindrances outside the scope of this work, the process is discussed thoroughly.

The second method introduces an LLM combined with a standard CNN. The image features of the CNN get translated into text, and the explanation happens in the text domain.

Therefore, these two models discussed in this chapter will investigate explanations originating from different domains.

3.1 FLEX-VQA

This section will delve deeper into implementing the first proposed method, FLEX-VQA. The name originates from the framework FLEX, originally introduced by Wickramanayake et al. [3], combined with visual question and answering (VQA).

3.1.1 Overview

To make VQA models easier to understand for the user, the explanation can originate from either of the two modalities of the model, either the language model, the image model, or both. The method presented in this experiment will address this task using the cross-modality explanation method FLEX. An overview of the FLEX model has already been described shortly in sub-chapter 2.5.2 on page 40. This method is relevant to this work because it explains the network's visual reasoning of why a picture should be in a given class using natural language. The FLEX framework combines a CNN, which is pre-trained on the specific task and does not need any further training that alters the CNN to give it explainable abilities. An overview of the original FLEX framework can be seen in Figure 3.1.

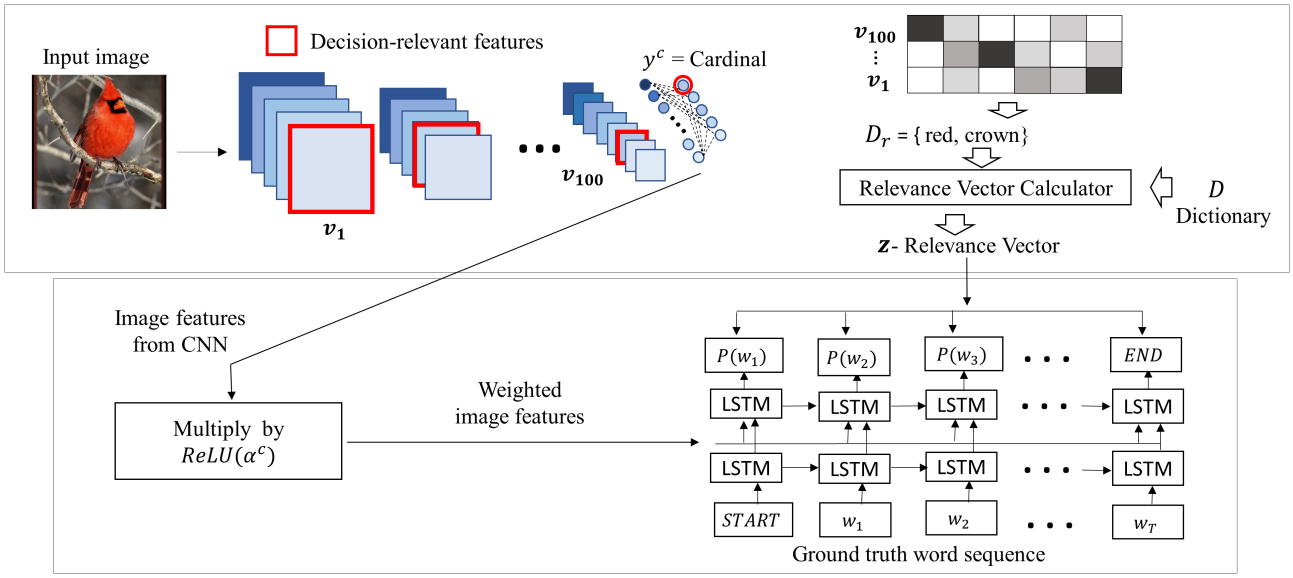


Figure 3.1: Overview of the FLEX Framework.
Figure credit: Wickramanayake et al. [3].

3.1.2 The motivation for this method

By using FLEX as a starting point, the purpose of the method in this experiment is to adapt it to have VQA abilities. In order to make this adaption, the dataflow shown in Figure 3.2 is proposed. Using this technique, a user can get an answer for a question regarding an input image that is both in natural language and faithful to the underlying CNN model, making it more explainable and transparent. The explanation is faithful to the underlying model since it uses the actual gradients in the CNN to find the words used to answer the question. This feature ensures that if the underlying CNN model has learned features that correlate with the answer but are not the intended or logical features to pay attention to, these flaws get communicated to the user. Making a model that can explain what it considers important to a user, also when flawed, is important when making a system where a user can make a factually based decision in trusting the answer.

3.1.3 Original FLEX in more detail

First, the FLEX framework works by having a CNN predict the class of an input image. Given the image and its predicted class c, the method finds a subset of the most important feature maps from each layer in the pre-trained CNN. To find the global activation score of each feature map, it sums up each neuron in the layer and then repeats this process for each layer in the CNN. After that, it sorts the layers in order of importance and chooses a subset of n layers so that the subgroup has a combined importance over a predefined threshold. When a subset of the most important visual features is collected, FLEX connects these visual features with linguistic features.

FLEX uses the Grad-CAM technique proposed by Selvaraju et al. [97] to do a backward pass through the feature maps of the CNN to find the largest class activation maps when predicting a class. This information is then used to map words to important visual feature extractions in the network. This mapping of words combines the separable feature layers salient layers of the Grad-CAM method with natural language to make better explanations.

The way FLEX achieves explainability is to find a connection score between a word (w) to the elements of the important feature maps (F) so

that the word (w) best describes the important visual element (v) in each feature map. With a given natural language description of the ground truth (D_I) of an image (I), FLEX uses the corresponding description for every image and builds a dictionary that contains all words in every description seen in the training data. This dictionary (D) is used when describing new images during testing. Creating this combined dictionary is conceptually similar to how language models generate a dictionary of tokens used for tokenization. For each word (w) FLEX calculates the co-occurrence score for each feature ($v \in F$), using the Dice score [134, 135], as seen in Equation 3.1. The word with the highest co-occurrence score gets associated with the visual feature (v).

$$score(w, v) = \frac{2 \times occur(w, v)}{count(w) + count(v)} \tag{3.1}$$

When the dictionary D_I for each image and the dictionary for all images D is computed, the dictionary for the individual class labels can be found. This class label dictionary (D_c) contains the combined dictionary of each image of a specific class and can be defined as $D_c = \bigcup D_I$, where the class label of I is c. Lastly, the dictionary D_r is the words extracted recursively from the CNN. When an image is provided, and the class is predicted, FLEX starts at the last convolutional layer and finds the top-k features from that layer and their associated words. This collection of associated words from feature maps is done recursively until the first convolutional layer is reached, and the union of all these words is named D_r.

To describe decision-relevant features, FLEX computes the relevance of a word w to each of the dictionaries using the formula in Equation 3.2, where D_x is a placeholder for D_r, D_I, and D_c.

$$\text{relevance}\,(w, D_x) = \begin{cases} 0 & w \notin D_x \\ -\log\left(\frac{|D_x|}{|D|}\right) & \text{otherwise} \end{cases} \tag{3.2}$$

When the relevance scores for each dictionary are calculated, the relevance vector z is defined as in Equation 3.3:

$$z = (g(w1), g(w2), ..., g(w\,|D|)) \tag{3.3}$$

Where w_i is the i^{th} word in D and

$$g(w_i) = relevance(w_i, D_r) + relevance(w_i, D_I) + relevance(w_i, D_c)$$

To train the framework to generate a textual description, FLEX uses two stacked LSTMs, where the first LSTM receives the ground truth word w_{t-1}. The second LSTM gets the hidden state of the first LSTM, concatenated with the visual feature vector from the CNN. The output of the second LSTM is encoded to vocabulary space to produce a conditional probability distribution, defined as:

$$P(w_{t-1}|w_{\leq t-1}, I) \tag{3.4}$$

This distribution samples the current word w_t at time step t. The relevance vector z is element-wise multiplied by this probability distribution to produce the relevance loss and is defined as:

$$\text{loss}\left(w_t, I\right) = \max\left(\boldsymbol{z} \odot P\left(w_t \mid w_{\leq t-1}, I\right)\right) \tag{3.5}$$

When no ground truth label is available during interference, FLEX samples from the conditional distribution in Equation 3.4 to get the next word then passed as the input to the LSTM in the next step.

This labeling of visual features can be reminiscent of DenseCap proposed by Johnson et al. [115], and Neural Baby Talk proposed by Lu et al. [136]. The main difference between these methods and FLEX is that FLEX can be added to a network after the architecture is finalized, trained, and evaluated. In theory, the framework is agnostic to the underlying network. However, one limitation is that the visual encoder must have a property in which distinct parts of the image activate distinct parts of the network.

3.1.4 Implementation

Given that FLEX can combine information from the visual domain with natural language, it is an exciting framework to explore in the field of XAI and VQA. A modified architecture to FLEX is presented in this thesis, as seen in Figure 3.2. This proposed architecture has unfortunately not been tested because of technical hindrances outside the scope of this work. A more detailed explanation of these hurdles is described in subsection

3.1.5. However, since much of the available time for this thesis was spent researching this approach and solving the hurdles, an overview of the architecture will still be provided in this section.

Dataset: VQA 1.0 and COCO

The dataset chosen for this experiment is the VQA 1.0 dataset [99] instead of the more balanced VQA 2.0 [137]. This experiment's goal is not to achieve state-of-art accuracy but to test the validity of this method. The VQA 1.0 dataset allows the VQA model to learn biases in the data that will be utilized in this method. Because of its unbalance, the authors of the VQA 1.0 dataset propose to use a k-way classification with 1000 classes. The rationale behind this decision is that the top thousand classes cover 82.67% of all the train and validation answers. Optimizing this way makes the assignment no longer open-ended but a more straightforward task by choosing one class out of a thousand.

This optimization will make the answer easier to compute, reducing training time and making it easier to control if the prediction is correct. Another benefit of this simplification is that the FLEX-VQA method can use the answer as a class, making it more efficient to adapt it to the FLEX framework. This is because the FLEX method builds a dictionary for each class (D_c). In order to make the dictionary D_c, the method will need descriptions for each image. Another benefit of using the VQA dataset is that the images originate from the Common Object in Context (COCO) dataset [100]. This dataset is a large-scale object detection, segmentation, and captioning dataset, meaning that the images in the VQA dataset are already described in COCO.

Proposed architecture

Figure 3.2 shows how the data flow in the proposed method is supposed to work. It closely follows the original FLEX architecture, with adaptations to make it answer questions. It consists of a CNN that extracts visual information and a 2-layered LSTM [74] RNN that combine linguistic information with extracted visual features. The CNN used in this implementation is the same as in the original framework, which is a modified VGG-16. The modification is a technique proposed by [9] et al. called Compact Bilinear Pooling, which reduces feature dimensionality

without sacrificing performance. This ensures a more efficient and faster training and inference time, which also has the potential benefit of reducing the risk of overfitting.

The wanted requirements of the system will need to be defined before any method can be developed. The method should be able to have an image as input alongside a question related to the content of this image. The wanted output is a locally accurate answer to the question and a description that is also locally accurate to the model.

The VQA model

The proposed architecture can be utilized to make a model and pipeline that achieves the required features. The proposed method consists of a standard VQA model with a k-way classification optimization. This architecture is essentially the same as the one presented in the original VQA 1.0 paper, called *deeper LSTM Q + norm I*, which achieves an accuracy of 57.75% on the complete test set. This network comprises an image pipeline (*norm I*) with activations from the CNN being l_2 normalized. The question pipeline (*deeper LSTM Q*) consists of an LSTM with two stacked layers. The image embeddings are then transformed to a dimensionality to match the question embeddings. This is done by a fully-connected layer, which makes the image features a vector with a length of 1024.

When the image and question features have the exact dimensions, they are merged by pointwise multiplication. This fusion vector is then passed through a fully connected neural network with two hidden layers, 1000 hidden units in each layer, and a dropout of 0.5. Finally, the answer with the best fit is found by passing the result of the fusion vector through a SoftMax layer to predict the final class.

FLEX

Since the VQA system is now a classification task, it can be matched with the FLEX framework. The importance score is recursively calculated using the image features in each feature map of the CNN. The formula used in this calculation is shown in Equation 3.6, where Z is the total number of neurons in the feature map A, the predicted label for class c is y^c and a_{ij} is the ij^{th} neuron in A.

$$\alpha^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial a_{ij}} \tag{3.6}$$

In order to remove negative influence and only evaluate scores contributing to the specific class, the importance scores α^c are then multiplied with a ReLU. These positive-only activation scores are then fed into the secondary stacked LSTM of the FLEX framework, which also gets the hidden state from the first LSTM. This stacked LSTM network is separate from the LSTMs of the VQA system, as it is trained only to output descriptions, whereas the one in the VQA is used to encode the question features. The rest of the FLEX framework is the same as described in Section 3.1.3. The main difference between the original FLEX and this method is that instead of the CNN predicting a class, the classification method now predicts an answer using the VQA model.

Training and testing

Given that FLEX is a post-hoc explanation method, it does not need to be trained simultaneously as the VQA system. This separation ensures that the VQA system can be trained separately from the explanation system and, thereby, can be tuned to achieve satisfactory accuracy before the explanation system is introduced. However, even though the systems are separate, the explanations are still connected directly from the extracted image features and, therefore, will be locally accurate to the CNN method.

In order to train the VQA system, the input image with corresponding questions and their ground truth answers is supplied. The VQA 1.0 dataset contains all these image-question-answer pairs and will be used in this experiment. Then entire VQA model is trained end-to-end using a cross-entropy loss, as described in the VQA 1.0 paper, where it was presented originally.

The FLEX is a separate model that is outside of the VQA model and, therefore, will need to be trained after the implemented VQA method is fully trained. Some optimizations can still be implemented to make this training process faster. The optimization, with a considerable impact on training time, is to precompute the image features when the VQA system is trained. Because FLEX uses the image features for each image together with the predicted answer, these can be saved during VQA training so that they do not need to be fed through the VQA model once more.

To train the FLEX framework to the task at hand, it must have extracted image importance scores, a ground truth description of the image from the COCO dataset, and the predicted answer for the question. The answer

is, in this experiment, regarded as the class label and, therefore, needed when FLEX makes the dictionary D_c. The dictionary D_I contains the image description, and the relevance dictionary D_r is calculated using the importance scores from the CNN.

While testing the combined system, the VQA method still needs an image and related questions. Then the predicted answer is made by the fully connected network. This answer defines the class dictionary D_c that the FLEX framework uses when making a conditional probability distribution. This distribution contains the combined relevance defined by z, which is a function of the relevance for D_c, D_I, and D_r, as described in Equation 3.3 in Section 3.1.3.

The LSTMs gets trained by feeding each word in the ground truth caption into the first layer alongside the hidden state of the LSTM. This is used to compute the next state, which gets concatenated with relevant activation scores from the CNN and is input to the second and last, LSTM-layer. The output from the second LSTM-layer is encoded to vocabulary space to make a conditional probability distribution that will later be used to calculate the probability of predicting that word, given the provided image features. The word with the highest probability is chosen and used as input in the first layer of the LSTM. This word prediction is repeated until the stop word is predicted.

3.1.5 Why this method has no results

To implement the proposed architecture as shown in Figure 3.2, it was natural to use the original code from the FLEX paper as a starting point. The authors of the FLEX paper also released the code used to do the experiments in the article on GitHub[1] to encourage others to iterate on their method. The original experiments use a CNN called Compact-Bilinear Pooling, a classifier proposed by Gao et al. [9]. While the FLEX framework is designed to be model agnostic, the actual implementation of the experiments is tied closely with the CNN, which proved to be a hurdle when trying to recreate the results in the paper and expand on the architecture and features. This method mainly does not provide finished experiments or results because an outdated machine learning framework makes compact-bilinear pooling CNN practically impossible

[1] https://github.com/sandareka/FLEX

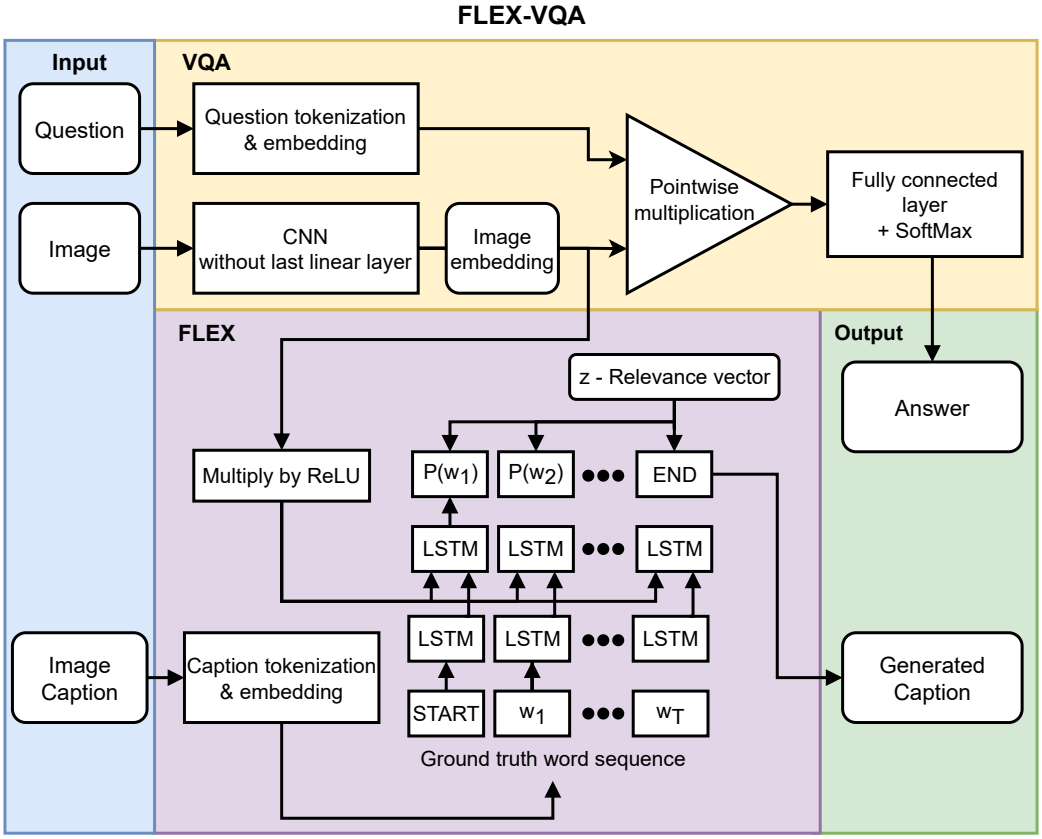


Figure 3.2: Proposal of the data flow and components explored in this experiment.

to execute. Combined with the tight integration between the implemented FLEX method and the CNN method, considerable resources have been put into separating these two methods, customizing the software, and building containers around it, with no success.

The original implementation of FLEX

For FLEX to work, it needs to get important features from the CNN. This is an essential part of the framework and therefore needs insight into how the layers of the CNN are structured. The implemented version of the original FLEX architecture uses the Compact-Bilinear Pooling CNN, which in short, uses kernelization to reduce the number of dimensions of bilinear features to make it more computationally efficient at the cost of having an architecture that deviates somewhat from a more traditional CNN architecture. Therefore, the authors of the FLEX paper have implemented a version of the framework that addresses these special considerations when calculating co-occurrence metrics between words

and features.

Why Caffe was needed

To use the same CNN as the original FLEX in a new context, as in this experiment, would require retraining or fine-tuning this network to suit the images in the given dataset. This would require the Compact-Bilinear Pooling model to train using the new pictures in the specific dataset. Alternatively, a new CNN could be chosen that could be merged with the FLEX framework. The original Compact-Bilinear Pooling came with pre-trained weights from training on the Caltech-UCSD Birds (CUB) [138] dataset. This dataset has images from Flickr and ImageNet containing a subset solely focused on bird species and a relatively small amount of photos (11,788, compared to 14 million images in ImageNet [65]). Therefore, the original weights are best fit to classify this CUB dataset and will require retraining outside this specific task. Because of how the original FLEX framework is structured, it would need to be substantially rewritten for a new CNN to be used in place of the Compact-Bilinear Pooling. Therefore the best way forward was to use this CNN, like the original method.

Getting Caffe to run

To train the Compact-Bilinear Pooling on a new dataset, the underlying machine learning framework it was built in had to be installed. This framework is named Caffe and is a deep learning framework developed by Berkeley AI Research (BAIR) / The Berkeley Vision and Learning Center (BVLC). It's a precursor to the framework Caffe2, which was initially built by Facebook and is now merged with PyTorch. Because Caffe was forked into Caffe2, the development of the original stagnated. Developers implemented new features and compatibility in Caffe2, while the original Caffe has not received an update since 2017. Because of the rapidly evolving nature of software and hardware since 2017, it proved to be a nontrivial task to get Compact-Bilinear Pooling written in Caffe to run on the hardware available during this project. The FLEX framework is implemented in TensorFlow version 1.7, which is also considered outdated at the time of writing. However, in contrast to Caffe, the teams at TensorFlow have made tools and helpers to run outdated code and translate it into current versions.

To get the compute benefits necessary for deep learning, the Caffe framework will need to run on GPUs. The framework supports GPUs from Nvidia with Compute Unified Device Architecture (CUDA) version 5 through 8 [139, 140], as well as AMD GPUs. The GPUs in the compute cluster available to run experiments for this thesis are Nvidia RTX 2080 Ti, Nvidia A100, and AMD Vega 10 XL/XT, which run CUDA version 11.7, 11.5 and ROCm version 4.5.0 respectively [141]. The mismatch between Caffe's highest supported CUDA version and the version available on the hardware need to be addressed. To not break compatibility with other services on the compute cluster, containerization was needed. Luckily containers are being actively deployed with Caffe versions with GPU support, most notably from Nvidia and AMD.

At the time the implementation of the architecture started, communication was established with the IT staff at the University of Oslo, which is responsible for the available computing cluster.

At that stage, only an experimental implementation of containerization was available through the container engine Podman [142]. Using this container engine, the Nvidia container could not attach the Nvidia GPUs. However, the AMD ROCm container could access the AMD GPUs. Unfortunately, this AMD container did not have the correct driver compatibility to train the CNN. Since Podman did not result in a container that could run successfully, a different container engine was installed, namely Apptainer, formerly known as Singularity. This new engine could see all the Nvidia GPUs, but not the AMD ones. After the initial errors were ironed out, a Caffe container from Nvidia was successfully installed. Although successfully installed, the container had trouble running the training examples in the FLEX code repository. When an error message was solved, a new one arose. Therefore no definite problem could be addressed, but rather several error messages pointing to possible driver incompatibility. The most likely explanation for the error messages was an incompatibility issue between the Caffe version in the container and the hardware it was tested on. After extensive testing and error-solving in most of this thesis time frame, the decision was taken that this technical problem was outside the scope of this experiment, and this method was discontinued for this project.

How these problems could have been mitigated

The main limitation of the chosen approach was getting the Caffe framework to work. However, the proposed method could still be implemented with a new approach. Because this method could bring new insight into how VQA systems are interpreted, exciting experiments could still be carried out. Some suggested improvements based on the experience gained implementing this method are:

- **Remove Caffe from FLEX**
  To have better compatibility with modern hardware and be more accessible for research moving forward, discontinued frameworks should be updated with more up-to-date versions. At the time of publishing the FLEX paper (2019), the development of the Caffe framework had already been stale for two years. To facilitate peer review and additional implementations so that the proposed methodology can be further developed, researchers benefit from publishing material for others to follow and implement themselves. Removing the Caffe framework from the FLEX framework makes the method more accessible and easier for others to expand on.

- **Update FLEX to TensorFlow 2**
  The FLEX framework is implemented in TensorFlow version 1.7, released in 2018. As of writing this thesis, TensorFlow versions 1.x are considered legacy. However, version 1.15, the latest version 1.x, is supported by TensorFlow 2 through a legacy helper. By updating to a more recent version, the framework can implement more modern features and optimizations, gain compatibility with modern hardware and acquire more users already familiar with the current versions. This upgrade can speed up the implemented FLEX framework through software and hardware optimizations and speed up the development of the framework itself. TensorFlow has migration guides and scripts to help translate from version 1 to 2 [143], which makes the migration manageable.

- **Make FLEX model agnostic in practice**
  Theoretically, the FLEX framework is separate from the CNN. As proposed in the paper where the framework is introduced, the implementation of the FLEX framework would benefit from being adapted to be agnostic to the underlying method. The

initially implemented method has methods to extract image features specific to the CNN used, making it dependent on that particular model. Using an agnostic implementation of the gradient search through the CNN feature maps, the method could be extended by allowing testing different CNNs to extract features and find co-occurrences with linguistic features. This could make implementing the technique for new CNNs easier and make more methods explainable.

3.1.6 Summary of FLEX-VQA

To summarize, by implementing the FLEX framework into a VQA method, the combined framework would make interpreting VQA models easier and locally accurate. FLEX has the same benefits as LIME of being locally accurate and model agnostic. Having insight into a locally accurate explanation model would make it possible for researchers and users of an implemented system to see if the underlying model is trustworthy based on what the model deems important.

Unfortunately, no experiments were conducted with this method due to technical difficulties. However, the process described in this section still provides insight into how an interpretable and locally accurate VQA system could be designed. The main takeaways from this section are how a framework could be designed to extract visual explanations in the visual domain described in the linguistic field. The FLEX framework searches through the CNN and finds important features that are then described using natural language. In order to make the FLEX framework easier to be developed further by future research, the model should be made agnostic to the CNN in practice, facilitating researchers to use the image encoder they want to study.

The following section will introduce an experiment where, in contrast to the previous, the visual features are first translated to the linguistic domain and explained using linguistic explanation techniques.

3.2 Alpaca-VQA

The experiment in this section will explore an approach to design an LLM that can interpret images, reason about their content, and be able to

answer general text questions. There are many ways a multimodal LLM could be developed, but this experiment aims to have a system with an explanation method in the linguistic domain. The rationale is that the previous experiment, described in Section 3.1, was explained in the visual domain and translated into text. The investigation in this section will do the opposite. The image features are extracted and translated to text, where the explanation system works on the text directly.

This section presents the implementation details of a VQA version of the Alpaca model. First, an overview of the reasons for choosing Alpaca for this experiment will be given. A more detailed description and rationale for the specific implementation of this model are then discussed, such as how the model has gained visual capabilities and how image features are implemented into the language model.

3.2.1 Overview

To make an LLM that can see the world and explain what it sees, it was outside the scope of this experiment to train an LLM from scratch, both regarding time and resources. When training, an LLM requires vast quantities of good-quality data. Training the model requires large compute clusters that consume lots of energy, cost much to facilitate, and produce unwanted greenhouse emissions. Therefore, an LLM that was already pre-trained was needed for this experiment to be fine-tuned further to answer the research goals.

Alpaca-LoRA

Even though the Stanford Alpaca model was trained using significantly less computing power compared to other LLMs, it is still advantageous to further lower the necessary compute budget to train and fine-tune models. A technique that was proposed by Hu et al. called Low-Rank Adaptation (LoRA) [144] addresses this issue. This technique manages this task by adding pairs of rank-decomposition weight matrices, commonly called update matrices, on the current weights and only trains these newly added weights. This approach has many advantages, most importantly accelerating the training while reducing memory consumption. Other benefits of these added matrices are that the original weights are kept frozen, making the LoRA model less prone to catastrophic forgetting. This

catastrophic forgetting can happen when large connectionist networks, like deep neural networks for LLMs, are trained on multiple tasks in sequence. In these scenarios, the models are prone to forgetting the earlier tasks [145]. Having the original weights available in a frozen state while training on a new task keeps the learned representations, as they are not changed. The added weights are typically less than 10 percent of the original trainable parameters, making training domain-specific models more feasible and faster. Because the fine-tuned matrices are so small, these LoRA weight matrices can be changed on the fly depending on the task. This allows for a system where a large base model, pre-trained on a large text corpus, is tuned to different requirements with smaller LoRA matrices that still benefit from the larger base model when answering. When using LoRA matrices, they also allow training and interference on consumer hardware, like regular desktop GPUs. It is even possible to do interference on a single-board computer like the Raspberry Pi, although at a significant speed decrease [146].

3.2.2 Implementation

This implementation has the goal of answering the research questions stated in 1.2. Specifically, to investigate if the model can be explained after it has finished training and if it still can bring knowledge from pre-training to the new modality.

In order to answer these questions, a LoRA version of the Stanford Alpaca version was used to carry out the experiments.

The code used as a starting point was a fork of Erik J. Wang's Alpaca-LoRA [147], which was further modified to fit the experiments in this section. The most notable modification to this model is to make it cross-modal by allowing it to take both text and images as input.

Make a language model see images

An image encoder had to be implemented to enable the model to interpret images. To allow the LLM to analyze image data, the visual features had to be extracted and translated into the linguistic domain, which could be merged with the input prompt. To make a system that more efficiently can change image encoders without changing the LLM, the design of the image-to-text method is to encode features as text strings. The proposed

dataflow is shown in Figure 3.3. This architecture allows the language model to be agnostic to the image encoder, allowing it to use a CNN or vision transformer that is fine-tuned to the task.

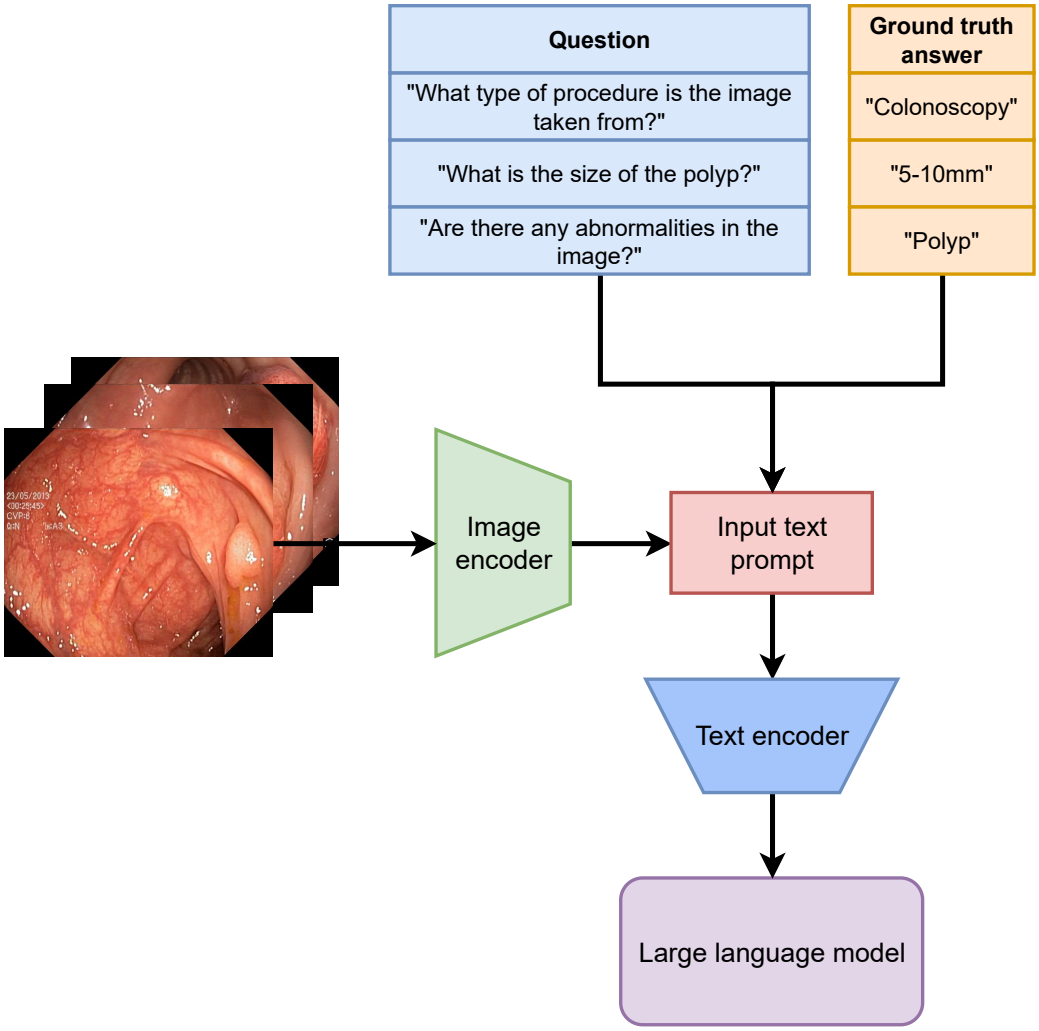


Figure 3.3: Overview of the proposed dataflow to make large language models interpret images. The images in this Figure are from the HyperKvasir dataset by Borgli et al. [148]. The rest of the diagram is by the author of this work.

Specifically, the image encoder first implemented was Visual Geometry Group (VGG)-16, proposed by Simonyan and Zisserman at Oxford [8]. The reason for implementing this CNN is that it is a known network that performs relatively well on the ImageNet dataset, where the correct label is in the predicted top five in 90.38% of the samples in the test set [149]. The VGG-16 model used for the experiments is pre-trained on the ImageNet [65] dataset. The model presumably performs better if pre-trained on the

specific dataset used, namely the HyperKvasir dataset [148] introduced by Borgli et al. However, the experiments will explore the proposed approach's feasibility rather than achieve optimal accuracy.

Given that the VGG-16 network is pre-trained on ImageNet, it outputs a probability for each of the 1,000 classes in the dataset. To accomplish the task of extracting valuable features from any image, regardless of whether CNN has been pre-trained for the task, the implemented method extracts the label and probability of the 100 ImageNet classes with the highest probability. With this approach, the image feature extraction will find a consistent number of features in an image, sorted with the feature with the highest probability first. The features with the highest probability will likely have a feature map similar to the predicted class in ImageNet. Therefore, even if the class label from ImageNet is not connected with a correct label from the HyperKvasir dataset, there is a high likelihood that the feature extraction will still be practical to extract image features.

The VGG-16 CNN was initially developed for the image classification task, not object classification. It consists of multiple convolutional layers followed by fully connected layers, and it does not have any built-in mechanisms for handling Region of Interest (ROI) operations. Therefore, since VGG-16 is designed as an image classifier, it only outputs the classes it recognizes in the image as a whole, which means it does not perform particularly well for localization tasks. Object classifiers, like R-CNN [109], Faster R-CNN [110], and variants of YOLO [111, 150–156], among others are made to output ROI, which would allow the image encoder also to encode locations of detected objects within the image. These object classifying qualities from an image encoder will most likely give the LLM image data of a higher quality to work from. However, the experiments will test the feasibility of the implementation using VGG-16.

Images to text prompt

To encode the image features into a format that the LLM can interpret, the image features must be converted into a form that can be embedded into a natural language question-and-answer format. The dataset used to train the Stanford Alpaca model follows a *question-input-answer* format, as shown in Figure 3.4. This format is structured so that the question comes first, followed by an optional input to the prompt with detailed information that can be used to answer the question. Additional up-to-

date information can therefore be embedded in the text prompt if the model's training set does not contain information to answer the question. This input function enables the LLM to answer new questions even after completing the training.

To make the new task of interpreting images while keeping the prompt text format close to the original, image features are incorporated into the *input* section of the original prompt. By not changing the original structure of the prompt, the model is better suited to utilize the knowledge gained from the pre-training. The feasibility of incorporating image features in text prompts has been successfully explored by Yang et al. in the paper MM-REACT [157]. The team explores methods to make ChatGPT a multimodal model, including interpreting images. The MM-REACT model achieves this by encoding images using an X-Decoder model proposed by Zou et al. [158]. This model features ROI capabilities using dense captioning, which outputs class labels and coordinates of the corners of bounding boxes. These extracted features are then concatenated into a text string that the LLM receives as an input prompt. The modified Alpaca model used in this experiment uses a similar approach to the one in MM-REACT by encoding image features as text. When using the VGG-16 model, the final text prompt to the Alpaca-LoRA model is the same as shown in Figure 3.5.

```
Below is an instruction that describes a task, paired with
an input that provides further context. Write a response
that appropriately completes the request.

### Instruction:
{instruction}

### Input:
{input}

### Response:
```

Figure 3.4: Overview of the original text prompt to the Stanford Alpaca model [4, 5], with additional input.

```
Below is a question that describes a task, paired with an
input that provides image features from an encoded image.
The form of the image features is (label, probability).
Write a response that appropriately completes the request.

### Question:
{question}

### Encoded image features on the form (label, probability):
{Top 100 most probable ImageNet classes from VGG-16.
 Example: ("dog", 0.73),("cat", 0.53) ...}

### Answer:
```

Figure 3.5: Overview of the modified text prompt to the Alpaca-LoRA model, including extracted image features. The text in curly braces is not part of the prompt but represents the placeholder for the question and extracted image features.

Text encoding

A text tokenization process is needed to break down the raw text data given in the prompt into smaller, standardized units called tokens, as described in Section 2.5.2. This process assures that text models, LLMs, in this case, can interpret linguistic input data. Tokenization is essential to transform unstructured text data into a format the model can process.

LLMs have been trained on specific tokenization schemes that use unique tokenization rules and vocabularies. Therefore, they should use the same tokenizer when pre-training the model to get adequate performance. Suppose a different tokenizer is used to pre-process the text data. In that case, the tokenization output may not be compatible with the language model's vocabulary and encoding scheme, leading to poor model performance and incorrect predictions.

The tokenizer used in this implementation is the one used by the original LLaMA model [6] since the Alpaca model is a fork of this model. HuggingFace [159] makes the specific implementation of the used LlamaTokenizer and is based on SentencePiece [160, 161]. This tokenizer and corresponding detokenizer allow for an unsupervised, end-to-end system without language-specific processing.

3.2.3 Explaining the output

As LLMs are large and complex models, they can be challenging to explain. The model used in this experiment is small compared to other state-of-the-art LLMs, yet it consists of seven billion parameters, making it too large for many XAI methods. Explaining large transformer models is an area of research where many are working on developing new approaches. However, there is still no *de facto* method to explain LLMs.

Transition Scores and Attention

There are still various approaches to getting an explanation from transformers. The most popular method is to extract values from the attention weights of the transformer [162–165]. Research has been done on finding subsets of the attention or even single neurons that are strongly associated with specific tasks, like the *"sentiment neuron"* presented by Radford et al. [166]. Here the researchers trained an LSTM on Amazon reviews to predict the next character of the text. When the model was trained, they made it into a sentiment classifier by adding a linear layer on top of the LSTM's vector units. Available labeled sentiment data trained this linear layer. They noticed that one single neuron significantly impacted the predicted sentiment value. By dialing this single neuron, the sentiment of the text generated by the LSTM could be controlled.

However, the attention weights have proven unreliable as a factual explanation of what the model evaluates on a low-level [167–169].

Although attention weights are not a reliable source of explanation, there still may be exciting insights to gain by analyzing these weights, as demonstrated by the *"sentiment neuron"*. In this experiment, the transition scores of the LLM will be used to gain insight into how the implemented Alpaca-VQA model works. These transition scores are calculated using its attention when the LLM predicts the next word in a sequence from the probability distribution. Although these scores do not give insight into the answer's validity, they indicate how sure the model is predicting the words.

Proxy Model and LIME

Because of the size of LLMs, XAI methods such as LIME and SHAP can be hard to implement to fit these large models or take a lot of time when

finding perturbations on the input text and their corresponding output. Initial experiments with both LIME and SHAP on the Alpaca-VQA model did not manage to make explanations because of the token size, making the run time unreasonably long.

A proxy model was therefore developed to have a model that can shed light on how the input affects the output. As LLMs are often complex and have millions or even billions of parameters, making their internal workings challenging to interpret. A simplified version that is easier to understand and explain can be created by training a proxy model on top of the large language model. This proxy model acts as a "translator" between the complex language model and human interpreters, providing a more interpretable representation of the underlying model's decision-making process. The proxy model can help address the issue of transparency and trust in AI systems. Like many deep neural networks, LLMs are often treated as black boxes, where the input-output relationship is not easily interpretable. By training a proxy model, insights can be gained into how the language model makes decisions, and this insight can be achieved using techniques such as LIME. These explanations can help users understand why the model made a particular decision and provide a level of transparency and trust in the system. Therefore, training a proxy model on an LLM enables us to bridge the gap between the complexity of the underlying model and the need for interpretability and explainability.

The proxy model used to mimic and interpret the Alpaca-VQA model is a Stochastic Gradient Descent (SDG) classifier. The data used to train the classifier includes 14,000 question-answer pairs, with the appropriate answers provided by Alpaca-VQA. The model is trained on the response predicted by the LLM instead of the ground truth to predict the same as the Alpaca-VQA. With 20,000 question-answer pairs created by the LLM, divided into 14,000 in the training set and 6,000 in the test set, the SDG classifier achieves an accuracy of 86% in the test set.

This proxy model is then fitted with the LIME method to make it explainable. Since LIME fits a linear model to the model it explains, there are three stacked models in total in the explanation pipeline, as seen in Figure 3.6. Even though stacked models obscure the actual decisions of the underlying model, some valuable insights can still be gained. Despite fitting a proxy model itself, the LIME method has proven effective in providing insight into the inner workings of numerous models.

Therefore, this experiment will test whether an additional proxy model would provide valuable explanations or insights into the Alpaca-VQA model.

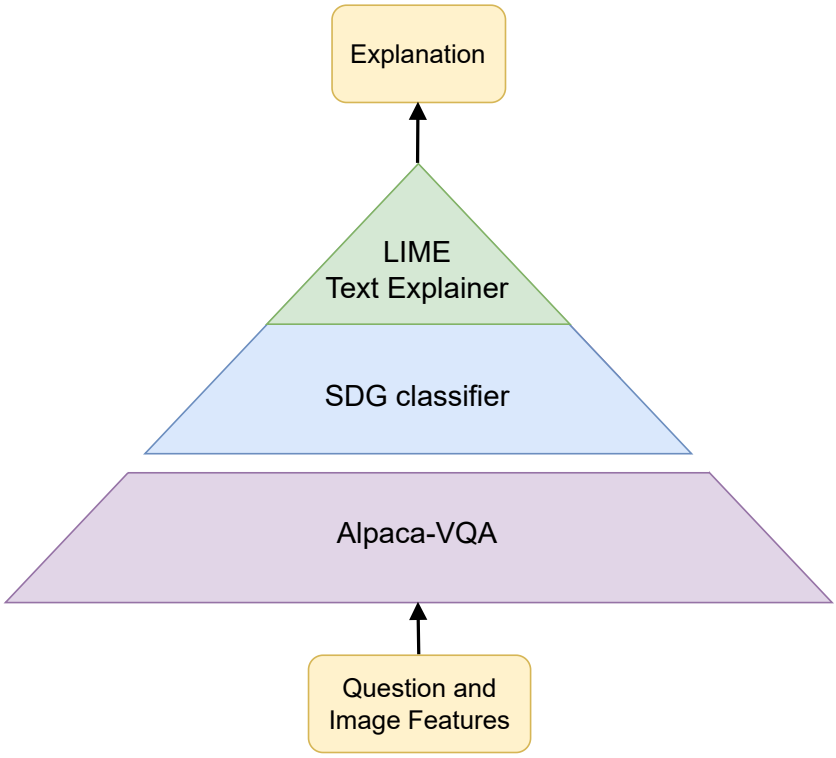


Figure 3.6: This figure represents the explanation pipeline of the Alpaca-VQA model using a proxy model that is explained by LIME. The question and image features are fed into Alpaca-VQA, which predicts an answer. This answer is used to train a SDG classifier that gets explained by LIME.

3.2.4 Dataset

The images used in this experiment are from the *HyperKvasir* developed by Borgli et al. [148], and the VQA extension using these images from the *ImageCLEFmed-MEDVQA-GI-2023* dataset by Hicks et al. [170, 171].

HyperKvasir

The available dataset on the gastrointestinal tract is one of the most extensive datasets, which comprises images and videos. The data was collected during examinations at Bærum Hospital in Norway, including anatomical landmarks and normal pathological findings. A subset of

the images and videos were annotated by at least one experienced gastroenterologist, from Bærum Hospital, the Cancer Registry of Norway, or Karolinska University Hospital in Sweden, together with one or more experienced persons working in the medical field.

The dataset can benefit medical and technical communities exploring semi-supervised and unsupervised methods. It can also help artificial intelligence-based computer-assisted diagnosis systems to provide better patient treatment. The full HyperKvasir dataset is available to the public and is open access[2,3].

ImageCLEFmed-MEDVQA-GI-2023

This dataset extends HyperKvasir by adding multiple modalities to a subset of the images, specifically VQA, Visual Question Generation (VQG), and Visual Location Question Answering (VLQA). The dataset is developed for the CLEF 2023 Medical Visual Question Answering (MedVQA) Challenge and is available to the public [4].

The question-and-answer ground truth is developed with medical partners, and the data include images spanning the entire gastrointestinal tract. Questions and answers include abnormalities, surgical instruments, and normal findings.

Since the experiments in this thesis are based on VQA, the questions regarding VQA in this dataset were used. To test if the LLM could learn some location capabilities, even though the CNN does not output bounding boxes, the questions regarding the location were also included.

Dataset Preparation

The original dataset is structured as a nested JSON file, and the structure can be seen in Figure 3.7. It is structured to have the image ID as the key and related questions and answers as value. The input text prompt to the model must have the image data, question, and answer bundled together in each prompt, as seen in Figure 3.5, the data structure was unrolled.

First, the nested structure was flattened so each row received the appropriate image ID, making the prompts easier to parse. The resulting

[2]https://doi.org/10.17605/OSF.IO/MH9SJ
[3]https://datasets.simula.no/hyper-kvasir
[4]https://github.com/simula/ImageCLEFmed-MEDVQA-GI-2023

structure can be seen in Figure 3.8. However, some questions, particularly regarding localization and color, have multiple correct answers. Instead of teaching the model to score multiple correct answers, the questions were flattened. Evaluating numerous correct answers to a question can be challenging because the model needs to know which answer is correct and which is incorrect. To circumvent this challenge, the answers in the dataset are flattened so that each row contains only one correct answer.

Having all information self-contained would make inputting one row into the prompt easier, as all necessary information is contained in the row. When the LLM predicts an answer, it can be quickly compared to the ground truth, making calculating accuracy more straightforward. This flatting of ground truth answers made the original 36,683 question-answer pairs into 52,051 pairs. The final structure of the data used in this experiment can be seen in Figure 3.9.

3.2.5 Context Window, Cuttoff, and Evaluation Metrics

This section will discuss essential features when training language models: context window, cutoff length, and evaluation metrics.

Context window and cutoff length

The context window of a language model refers to the number of words or tokens considered in predicting the next word or token. The context window is typically linked to both the input prompt and the output since the model refers to the input when generating the output. When training LLMs, the model receives a sequence of input tokens and produces a series of output tokens. Similarly, when fine-tuning a pre-trained language model on a specific task, the context window is also linked to both the input and the output. The input would consist of the prompt defined by the context window, and the output would be the predicted tokens that the LLM predicts based on the tokens in the context window. By adjusting the context window size, researchers can control the amount of context the model uses to generate its predictions, which can impact its accuracy and efficiency.

Cutoff length is a term often used when fine-tuning a pre-trained language model. The cutoff length specifies the maximum length of the prompt used when generating an answer.

Therefore, the context window and the cutoff length are closely related features that should be considered in conjunction. In practice, the context window can be regarded as a way to capture longer-term dependencies between words in a sequence. At the same time, the prompt cutoff length imposes a constraint on the size of input the model should process for a given task.

The updated input prompt shown in Figure 3.5 was used to train the model to interpret images. The prompt cutoff length was updated from the original 256 to 1485 tokens. The original LLaMA model was trained on 2,048 tokens [172], and the original Alpaca model is fine-tuned with 512 tokens as the cutoff length. In this experiment, the Alpaca-LoRA implementation is used. The authors of the model found that 96% of the prompts in the training data could be answered using a cutoff length of 256 tokens.

When deciding on the new token length, including the top 100 classes from VGG-16, the LlamaTokenizer must encode the whole input text, including image features.

When the class prediction score was represented using Numpy's floating number with 32 bits of precision, the average token length of just the image features where 162,654. As this would increase the original token size by more than 63,000%, it was considered too significant to be reasonable for the model. Because the used language model uses the LoRA technique to lower memory consumption, such a considerable cutoff length should still technically be feasible. However, the increased training and inference time was considered too inefficient for this experiment. The prediction score was converted to floats with 16 bits of precision to decrease the image token length. In practice, this would not make a difference in accuracy when fed into the language model, as the predicted class is just a placeholder for the underlying feature maps extracted. The advantage of converting the prediction score is that the average token length of the extracted 100 classes was reduced to 144,654 tokens. The reduced token size is an 11% reduction of tokens from the 32-bit precision representation and would still benefit from a further reduction. To give an additional decrease in the length of the prediction scores, they were rounded to three decimals. This will still provide the language model insight into the extracted class labels and the relationship between the classes while also significantly reducing the prediction score token length.

The final rounded scores have an average length of 1,229 tokens, resulting in a reduction of 99% from the original 32-bit precision score. Combined with a prompt text length of 256, the total cutoff length is therefore calculated to be 1,485 tokens. Thus, this cutoff length is designed to cover 96% of the original prompts while incorporating the 100 most probable classes and their predictions.

Evaluation metrics

The evaluation metrics used in the Alpaca-VQA model experiments are precision, recall, accuracy, and F_1 score. These metrics are defined in detail in Section 2.4.

The precision is used to evaluate the proportion of the correct predicted classes, and the recall is used to see how well the model predicts correct classes. F_1 score is used to find the harmonic mean of the model's precision and recall and is a combined score of these two metrics. This score is used to more easily compare the score of each class in the test dataset. Accuracy is used to see how well the model predicts the correct classes.

Lastly, perplexity is used as a metric to analyze how well the model predicts an answer, given its probability distribution. The features from this perplexity calculation are also used to calculate a prediction score for each token in the answer. This gives insight into how sure the model is about its prediction.

3.3 Summary

In summary, this chapter has discussed the methods used when developing the two approaches to explain VQA tasks. These two models investigate explanations originating from different domains.

The first proposed method, FLEX-VQA, combines the FLEX framework with VQA to explain the network's visual reasoning to explain why an answer to an image-question pair is using natural language. The Alpaca model was chosen for this experiment as it was already pre-trained and required less computing power than other LLMs. The LoRA technique was used to fine-tune the Alpaca model further, which adds pairs of rank-decomposition weight matrices on the current weights and only trains these newly added weights. This approach accelerates the training

while reducing memory consumption, making training domain-specific models more feasible and faster. The image features of the CNN are translated into text, and the explanation happens in the text domain of the LLM using LIME and transition scores.

The next chapter will present the results achieved with the Alpaca-VQA, as the FLEX-VQA method was unfeasible to run because of outdated frameworks. The results gained from the Alpaca-VQA will be explained, discussed, and compared.

```
[
  {
    "ImageID": "clb0lbwzadoyc086u0brshvx5",
    "Labels": [
      {
        "Question": "Are there any abnormalities in the image?",
        "AnswerType": "Text",
        "Answer": [
          "Polyp"
        ]
      },
      {
        "Question": "What color is the abnormality?",
        "AnswerType": "Text",
        "Answer": [
          "Red",
          "Pink"
        ]
      },
      ...
      {
        "Question": "Where in the image is the abnormality?",
        "AnswerType": "Text",
        "Answer": [
          "Upper-left",
          "Lower-left",
          "Center-left",
          "Lower-center"
        ]
      },
      {
        "Question": "Where exactly in the image is the instrument located?",
        "AnswerType": "segmentation",
        "Answer": "clb0lbwzadoyc086u0brshvx5_mask.png"
      }
    ]
  },
  {
    "ImageID": "cla820gl5s3vv071u18ipbr2h",
    "Labels": [
      {
        "Question": "Are there any abnormalities in the image?",
        "AnswerType": "Text",
        "Answer": [
          "Ulcerative colitis"
        ]
      },
```


Figure 3.7: The original JSON from the ImageCLEFmed-MEDVQA-GI-2023 dataset. It comprises 6683 question-answer pairs related to images in a nested structure, with the image ID as the key and the question-answer pairs as values.

	Question	AnswerType	Answer	ImageID
0	Are there any abnormalities in the image?	Text	[Polyp]	clb0lbwzadoyc086u0brshvx5
1	Are there any anatomical landmarks in the image?	Text	[No]	clb0lbwzadoyc086u0brshvx5
2	Are there any instruments in the image?	Text	[Biopsy forceps]	clb0lbwzadoyc086u0brshvx5
3	Have all polyps been removed?	Yes/No	[No]	clb0lbwzadoyc086u0brshvx5
4	How many findings are present?	Number	[2]	clb0lbwzadoyc086u0brshvx5
...	...	...	...	...
36678	What type of polyp is present?	Text	[Not relevant]	clb0lbwypdods086u2txd5l85
36679	What type of procedure is the image taken from?	Text	[Colonoscopy]	clb0lbwypdods086u2txd5l85
36680	Where in the image is the abnormality?	Text	[Center, Upper-right, Lower-right, Center-righ...	clb0lbwypdods086u2txd5l85
36681	Where in the image is the anatomical landmark?	Text	[Not relevant]	clb0lbwypdods086u2txd5l85
36682	Where in the image is the instrument?	Text	[Center, Upper-left, Center-left, Lower-center]	clb0lbwypdods086u2txd5l85

36683 rows × 4 columns

Figure 3.8: The ImageCLEFmed-MEDVQA-GI-2023 has the image-IDs flattened. In this structure, each row has a question-answer pair and a corresponding image ID. Responses are still grouped in a list, making it difficult to evaluate individual responses.

	Question	AnswerType	Answer	ImageID
0	Are there any abnormalities in the image?	Text	Polyp	clb0lbwzadoyc086u0brshvx5
1	Are there any anatomical landmarks in the image?	Text	No	clb0lbwzadoyc086u0brshvx5
2	Are there any instruments in the image?	Text	Biopsy forceps	clb0lbwzadoyc086u0brshvx5
3	Have all polyps been removed?	Yes/No	No	clb0lbwzadoyc086u0brshvx5
4	How many findings are present?	Number	2	clb0lbwzadoyc086u0brshvx5
...	...	...	...	...
36681	Where in the image is the anatomical landmark?	Text	Not relevant	clb0lbwypdods086u2txd5l85
36682	Where in the image is the instrument?	Text	Center	clb0lbwypdods086u2txd5l85
36682	Where in the image is the instrument?	Text	Upper-left	clb0lbwypdods086u2txd5l85
36682	Where in the image is the instrument?	Text	Center-left	clb0lbwypdods086u2txd5l85
36682	Where in the image is the instrument?	Text	Lower-center	clb0lbwypdods086u2txd5l85

52051 rows × 4 columns

Figure 3.9: The ImageCLEFmed-MEDVQA-GI-2023 with image-IDs and answers flattened. This structure has one question-answer pair on each row, allowing for easier calculation of predictions. This is the structure used when training the model in this experiment.

Chapter 4

Experiments, Results, and Discussion

4.1 Intro

In this chapter, the Alpaca-VQA model presented in the previous chapter is tested on two different dataset sizes. First, hyperparameters, their values, and the reasons for choosing these are discussed. Then an investigatory experiment is conducted using a smaller dataset to get initial results for training on more data. These results are analyzed to understand better how the model responds to the data, aiding understanding how the model responds to the data. The Alpaca-VQA model is then trained on 20,000 samples, and its results are discussed. Methods for visualizing transition scores and training a proxy model explained by LIME are implemented when the Alpaca-VQA model has finished training. The insights from these supplementary methods uncovered a possibility that the model did not evaluate the image features when predicting an answer. A language-only Alpaca-VQA model was tested to test this hypothesis, and its findings are discussed. Finally, a more general examination of the findings in this work is discussed.

4.2 Hyperparameters

The Alpaca-VQA model is based on the Stanford Alpaca model and therefore draws inspiration when deciding on hyperparameters from their findings. To finetune the Stanford Alpaca model, the authors suggest

training for two or three epochs. The low number of epochs is to prevent the model from forgetting the previous knowledge and still be able to use the knowledge gathered from the original training corpus, known as *catastrophic forgetting*. Therefore, for the following experiments, the hyperparameters were chosen as in Table 4.1.

The LoRA parameters were selected to be the same as the original Alpaca-LoRA implementation. The transformer architecture contains four weight matrices for the self-attention module. These weights are for *query* (W_q), *key* (W_k), *value* (W_v) and *output* (W_o). In the paper introducing LoRA, the authors find that when applying LoRA to the attention weights for LLMs, specifically GPT-3 on the datasets WikiSQL [173] and MultiNLI [174], they achieved the best results overall by only adapting the W_q and W_v matrices. The developer of Alpaca-LoRA also confirmed these results, and therefore the experiments in this thesis will also use these update matrices. The authors of LoRA also did experiments for GPT-2 and GPT-3 to see which effect the rank r would have on the performance. They found that LoRA performs competitively with small values of r, especially when only adapting W_q and W_v.

As seen in Table 4.2, the LoRA update parameters are only 0.0622% of the original amount of trainable parameters in the Alpaca-VQA model. This considerately reduced amount of trainable parameters using LoRA update matrices results in considerably faster training and lower memory use.

The rationale for deciding on the batch size and micro-batch size values was to make it fit in the available Random Access Memory (RAM) on the GPU it was trained on. Since the current state of the code can only utilize a single GPU, it was necessary to fit the model on the GPU while training. The GPUs used for this experiment was an Nvidia RTX2080Ti with 11 GB of Video RAM (VRAM) and an Nvidia A100 with 40 GB of VRAM. For the initial experiment, the smaller RTX2080Ti was used not to demand more compute resources than needed. The A100 was used when training on the larger datasets, and the batch size was doubled to 256, which benefited from more VRAM. The increased batch size would allow the model to see more of the samples simultaneously during training. The number of epochs was kept to three while the step length was increased, effectively making the model evaluate more often in each epoch.

The size of the validation set was chosen to be roughly 30% of the

training data, and the cutoff length was chosen to be 1485, as this corresponds to 256 question tokens + 1229 tokens from the encoded image, as discussed in Section 3.2.5. This token length should allow the model to see most of the available input data, both text and images when predicting the output.

The temperature parameter influences the probabilities when predicting new tokens. In practice is a measure of how creative the language model should be during text generation. A lower temperature value makes the model choose tokens it is more confident are correct, and a larger value makes it more creative. As this task is to answer a question, the goal is more based on facts than creativity, and therefore the temperature was chosen to be 0.1.

During generation, the Alpaca-VQA model uses beam search with four beams. This means that it generates and searches in four sequences before deciding on the sequence with the highest combined transition score. This sequence is the one that the model is most confident fulfills the task. The values of *top-K* and *top-p* influence how many words are considered in the probability distribution during token generation. A top-K of 40 means it considers the 40 most probable tokens in the distribution. Top-p narrows down this distribution of top-K words to the smallest group that fulfills a cumulative probability above the set value. With these hyperparameters set during generation, the Alpaca-VQA model achieves a reasonable broad distribution of tokens to sample from while maintaining the computation demand reasonable.

If no further details are given in the experiments, the parameters described here are the ones used in the following experiments.

In the next section, an investigatory experiment will be carried out. By testing the model on a smaller dataset, it can be explored how the model responds to the dataset without using extensive computational resources. The results from this experiment will be analyzed and help make the final model a better fit for the task.

4.3 Investigatory Experiment

Before training the model on the complete dataset, a subsection of the available training data was used to see how the model would respond. Running the model on smaller training data makes it possible to get

Alpaca-VQA Hyperparameters	
Hyperparameter	**Value**
Training	
Batch size:	128
Micro batch size:	4
Number of epochs:	3
Learning rate:	0.0003
Validation set size:	30% of training data
Text Generation	
Temperature:	0.1
Cutoff length:	1485
Top-p:	0.75
Top-K:	40
Number of beams:	4
LoRA	
Rank r:	8
Alpha α:	16
Dropout:	0.05
Weight Matrices:	W_q, W_v

Table 4.1: Hyperparmaters selected for the initial experiment with Alpaca-VQA, fine-tuning the LoRA matrices on the ImageCLEFmed-MEDVQA-GI-2023 dataset.

initial results quickly, providing insight into where the method could be improved.

The investigatory experiment was conducted using a subset of the original *ImageCLEFmed-MEDVQA-GI-2023* dataset. The subset was chosen to be on 5000 samples, corresponding to 9,7% of the total dataset. The reason for doing the initial investigatory experiment with this subset is to test the feasibility of the method and implementation without using excessive time and computing resources.

4.3.1 Results

The graph in Figure 4.1 shows the training and evaluation loss for the training session over 3 epochs, with 8 evaluation steps during training. As seen by the graph, the model has a decrease in loss midway before it flattens out without a further significant decrease. This can suggest that the model is able to fit the data in the training set. However, since it does

LoRA Training Parameters	
Parameter	Value
Number of trainable LoRA parameters:	4,194,304
All parameters in Alpaca-VQA:	6,742,609,920
Trainable percent:	0.0622%

Table 4.2: Overview of the number of trained parameters using LoRA.

not continue to decrease, it indicates that the model has already learned the relevant parts midway through the training. It can also be seen that the evaluation loss follows the training loss closely, indicating that the model does not overfit the available data. The loss flattens towards the end, suggesting that the chosen hyperparameters were reasonable for the size of this training data. However, changing hyperparameters such as learning rate or LoRA parameters may improve the model. By increasing the size of the LoRA update matrices, the model could learn more features, which could increase the fit.

It can also be seen that evaluation loss closely follows the same trend as the training loss. The loss on the validation data is expected to be lower than the training data, which is often called the *generalization gap*. When the training and evaluation loss follow each other, separated by the generalization gap, it can indicate that the model did not overfit the available data. If the model were to overfit, the gap between training and evaluation loss would tend to increase at the end. Training loss would continue to decrease, and evaluation loss would flatten out or possibly increase.

In order to see how well the model scores on the test set of 5,000 samples, a classification report can be seen in Table 4.3. The columns are class labels, the precision, recall, and F_1 scores for each class, and the number of occurrences of each class in the test set in the column *Support*. At the bottom of the table, the accuracy and average score are calculated. This classification report shows that the model has an overall accuracy of 26% and is the most accurate in classifying the classes "0" and "Not relevant".

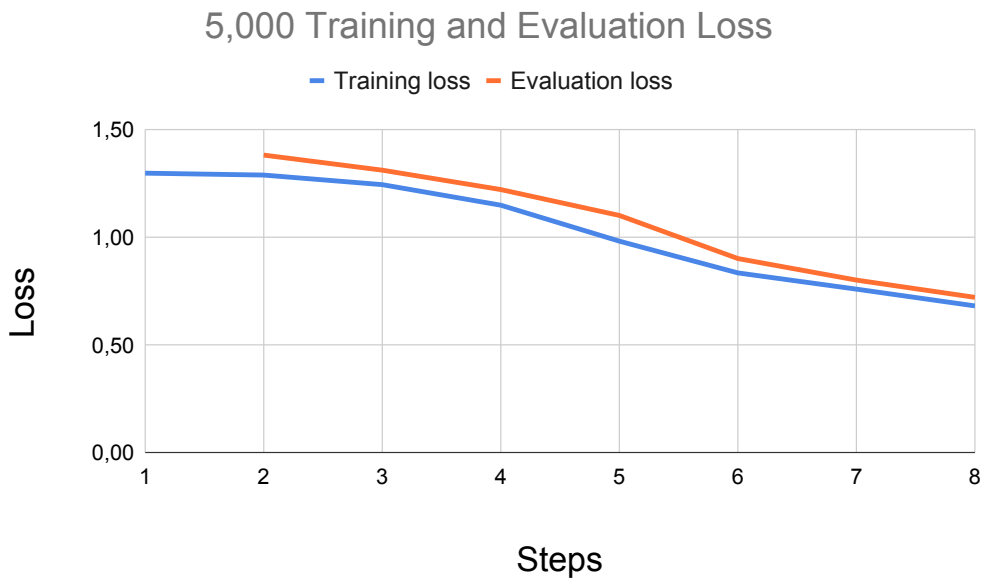


Figure 4.1: Graph over training loss for the initial experiment on 5,000 samples.

Table 4.3: Classification Report: Investigatory Experiment

Class	Precision	Recall	F1-score	Support
0	0.51	1.00	0.68	298
1	0.00	0.00	0.00	215
11-20mm	0.00	0.00	0.00	16
2	0.00	0.00	0.00	63
3	0.00	0.00	0.00	5
5-10mm	0.00	0.00	0.00	14
< 5mm	0.00	0.00	0.00	18
>20mm	0.00	0.00	0.00	19
Biopsy forceps	0.00	0.00	0.00	9
Black	0.00	0.00	0.00	2
Blue	0.00	0.00	0.00	1
Brown	0.00	0.00	0.00	3
Cecum	0.00	0.00	0.00	7
Center	0.00	0.00	0.00	222
Center-left	0.00	0.00	0.00	167
Center-right	0.00	0.00	0.00	166

Continued on next page

Table 4.3 – continued from previous page

Class	Precision	Recall	F1-score	Support
Colonoscopy	0.00	0.00	0.00	144
Gastroscopy	0.00	0.00	0.00	51
Grey	0.00	0.00	0.00	1
Lower-center	0.00	0.00	0.00	190
Lower-left	0.00	0.00	0.00	125
Lower-right	0.00	0.00	0.00	150
Metal clip	0.00	0.00	0.00	1
No	0.00	0.00	0.00	492
Not	0.00	0.00	0.00	0
Not relevant	0.27	0.89	0.41	1,057
Oesophagitis	0.00	0.00	0.00	51
Orange	0.00	0.00	0.00	3
Paris iia	0.00	0.00	0.00	25
Paris ip	0.00	0.00	0.00	22
Paris is	0.00	0.00	0.00	21
Pink	0.00	0.00	0.00	158
Polyp	0.00	0.00	0.00	65
Polyp snare	0.00	0.00	0.00	6
Red	0.00	0.00	0.00	125
Tube	0.00	0.00	0.00	43
Ulcerative colitis	0.00	0.00	0.00	43
Upper-center	0.00	0.00	0.00	184
Upper-left	0.00	0.00	0.00	156
Upper-right	0.10	0.13	0.12	142
Violet	0.00	0.00	0.00	1
White	0.00	0.00	0.00	108
Yellow	0.00	0.00	0.00	19
Yes	0.09	0.10	0.10	343
Z-line	0.00	0.00	0.00	47
Grey	0.00	0.00	0.00	2
Accuracy			0.26	5,000
Macro Average	0.02	0.05	0.03	5,000
Weighted Average	0.10	0.26	0.14	5,000

4.3.2 Analysis

In order to investigate why the model does not have a higher accuracy and only gets a few of the classes correctly classified, its answers should be examined. As seen in Table 4.4, of the 5,000 test samples, the model answers with class "Not relevant" in 70.14% of the instances. It can also be seen that it only answers with five class labels on the test samples. This could be a result of the unbalanced dataset and the fact that it has not learned to generalize well.

Answers given by investigatory model	
Class Label	Count of answers given
Not relevant	3,507
0	583
Yes	385
Not	343
Upper-right	182

Table 4.4: Answers given by the investigatory model on the test set of 5,000 samples.

It is helpful to analyze the dataset used in training to investigate why the model often answers with "Not relevant". As seen in Figure 4.2, the majority of correct answers in the dataset are of the class "Not relevant". This dataset can be balanced by either over-sample the non-majority classes or under-sample the abundant class, making the model learn that "Not Relevant" is not the best answer in 21,7% of all questions. Going further in the following experiments, the original dataset will be modified. The class "Not relevant" is removed, and the rest of the dataset stays the same. The rationale behind not doing any additional sampling modifications is to reflect the natural occurrences of question-answer pairs in the original dataset. Since this dataset is based on real-world examinations in a hospital, the number of occurrences in the dataset could reflect the real-world occurrence of these findings. In contrast, if this dataset were made synthetically and did not have biases from real-world examinations, there would be more reasonable to modify the number of occurrences of the classes to get a better fit.

The Alpaca-VQA model will be trained on a larger dataset in the next section. This is to explore how well it can fit the available data and which

insights can be gained.

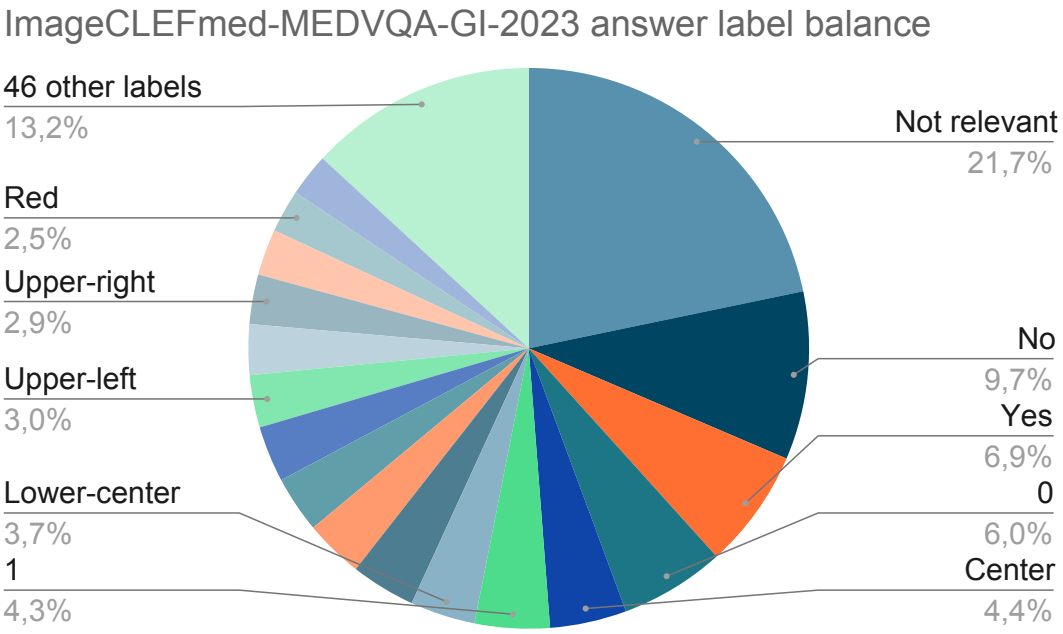


Figure 4.2: Overview of the answer label balance in the ImageCLEFmed-MEDVQA-GI-2023 dataset. The category "Other" is a collection of 46 labels that occur in less than 2,5% of the samples.

4.4 Main Experiment

In this experiment, the model is trained on four times as much data as in the investigatory experiment. As noted in the analysis of the previous experiment, the dataset used in this experiment is the same as the original, only with the class "Not relevant" removed. Removing this class completely also allows the model to see more relevant data since many of the chosen samples are no longer from the "Not relevant" class. By removing this class, the 20,000 samples will contain more question-answer samples that are relevant to this classification problem.

The dataset was split as shown in Figure 4.3, with the original dataset split in two, where the first part was used to train the model, and the second part was used to test and train the proxy model. The reason for leaving half of the dataset unused during the training of the model was to preserve unseen data that could be used to train the proxy model and

to test how the model would perform on unseen data. While the model presumably would be able to fit the new task of seeing images given more data, it will still be possible to evaluate if this method is able to interpret image data. How the preserved test data is used to train the proxy model will be discussed later in subsection 4.5.3.

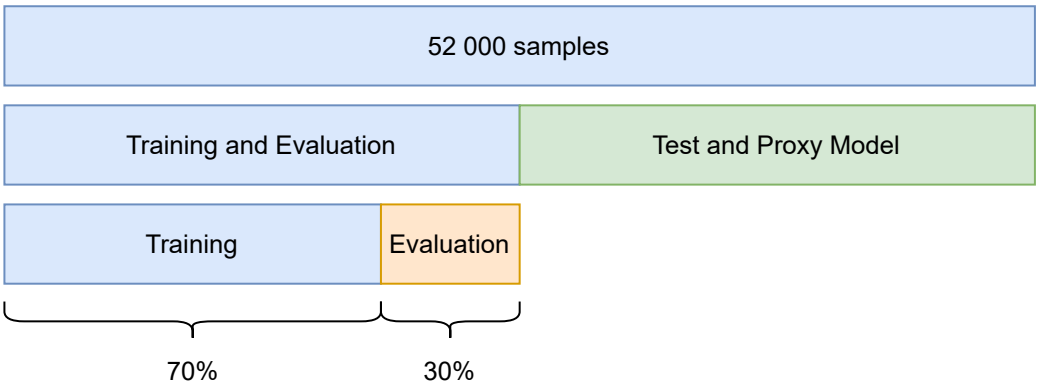


Figure 4.3: Illustration on how the dataset was split into training, evaluation, and testing data.

While training the model on considerably more data would be interesting, the time and computing cost for this experiment would not justify the possible better-trained system, resulting in a more generalized and nuanced model. The experiments carried out with the Alpaca-VQA model aim to investigate how an LLM would perform on a VQA task with image features encoded in the text. The final dataset used for training in this experiment contains 20,000 samples, and the class distribution can be seen in Figure 4.4.

The hyperparameters used in this experiment are the same as in the previous experiment and can be seen in Table 4.1, with a batch size of 256, as previously discussed. While the dataset used in this experiment contains four times more samples than in the previous experiment, the image feature extraction is still the extracted top 100 features from the VGG-16 model. Due to the similarities in the data samples, the hyperparameters are kept the same to see how the Alpaca-VQA model responds to a larger dataset.

Because of the larger dataset, more evaluation steps were used to help the model correct during training. In the investigatory experiment, 8 steps were used during the runtime. In this training session, with four times as much training data, the number of steps was chosen to be every 10^{th}

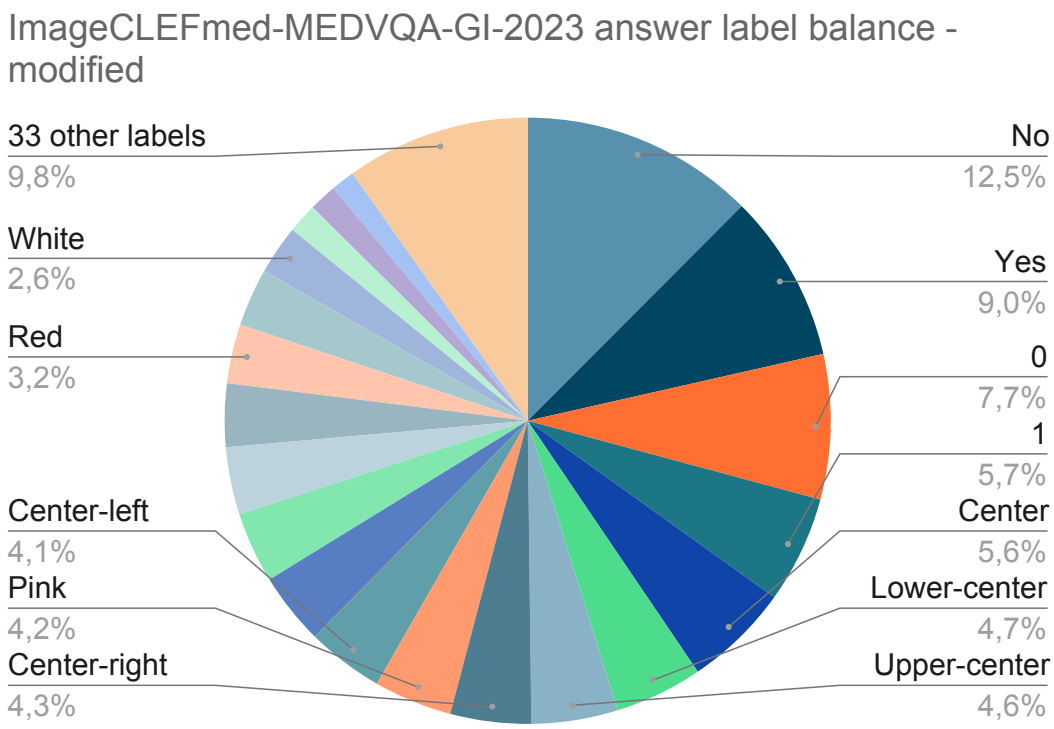


Figure 4.4: Overview of the answer label distribution in the ImageCLEFmed-MEDVQA-GI-2023 dataset, modified with the class "Not relevant" removed. The category "Other" is a collection of 33 smaller labels in the training data and is only used in this visualization, and is not an actual class.

training step, resulting in 46 evaluation steps in total.

The Alpaca-VQA model on 20,000 samples was trained on an Nvidia A100 with 40 GB VRAM. It took fifteen and a half hours to run, including image encoding, fine-tuning, and evaluating the model.

4.5 Results

This section evaluates the results for the Alpaca-VQA model trained on 20,000 samples, where 70% was used as training data and 30% as evaluation. The training results will be discussed alongside the classification report, transition scores, proxy model, and a blinded model.

First, the training and evaluation loss graph during this training session can be seen in Figure 4.5. In this graph, it can be seen that the model converges relatively fast and stabilizes during the training run.

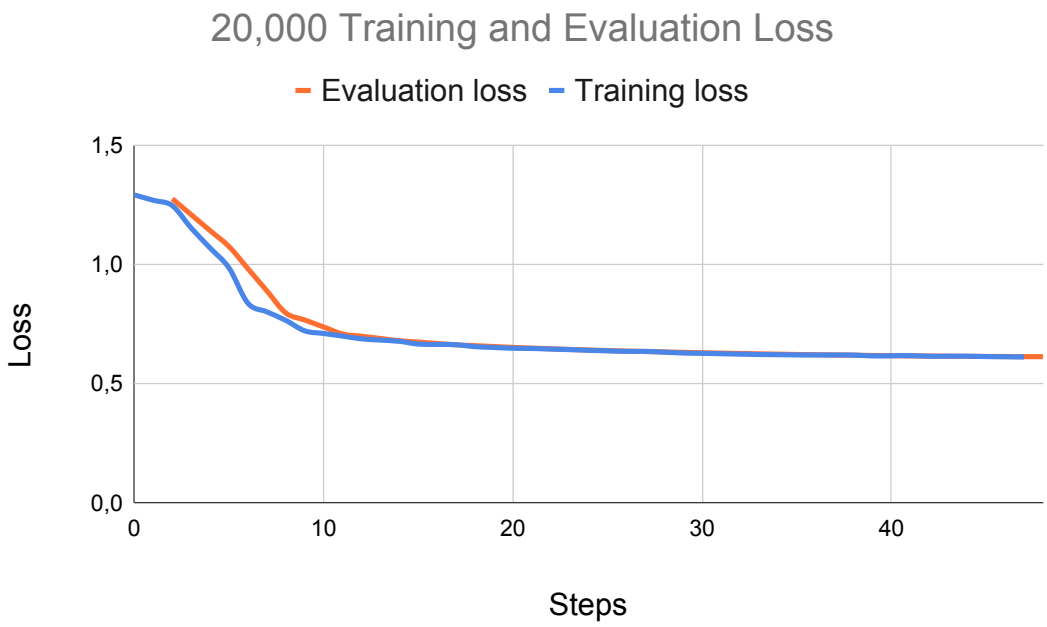


Figure 4.5: Graph over training loss for 20,000 samples.

This can indicate that the model was able to fit the available training data and fine-tune its LoRA update matrices to the training data. Regarding the number of evaluation steps, it can be seen that the evaluation loss is approximately the same as the training loss at step 10. At this step, it can also be seen that the loss for both training and evaluation flattens out. This can indicate that the model has fitted its parameters to the available data and did not have more quality data to learn from.

4.5.1 Classification Report

In order to test how the model was able to fit the training data, a test set was used. The test set was from the same dataset as the training data but naturally did not overlap with the training and evaluation data.

As the test result should be used later in order to fit a proxy model, the test set was chosen to be 20,000 samples. These samples should not be confused with the ones used for training and evaluation, as the dataset was split, as previously shown in Figure 4.3.

The classification report is shown in Table 4.5. As noted in this table, the Alpaca-VQA scores an accuracy of 29% on the test set. The number of occurrences in each class shows that this test set is also unbalanced and

has approximately the same distribution as the training set.

A simplified classification report can be found given that Table 4.5 is a large table that can be hard to interpret. As the F_1 score is a harmonic mean of the precision and recall, it can be used to indicate how well the model performs in each class. By removing the classes where the F_1 score is 0 from Table 4.5, the result is a simplified representation that can be seen in Table 4.6. From this table, it can be more clearly seen that the classes the trained model performs well on have a high presence in the training set, as previously shown in Figure 4.4. Logically, the model performs better on data it has seen and not as well on data it has not been exposed to. From the difference between the full Table 4.5 and the simplified Table 4.6, it can also be seen that despite having relatively high occurrence in both training and testing sets, the Alpaca-VQA model falsely categorizes classes with localization and color labels. One possible explanation for this is that the image encoder, the VGG-16 model, does not have localization abilities, as it is not designed to output ROI or multiple objects with bounding boxes. As it is not fine-tuned on the images in this dataset but rather pretrained on ImageNet, the predicted classes may not correlate with colors.

Table 4.5: Classification Report on Test Set - Model trained on 20,000 question-answer pairs.

Class	Precision	Recall	F1-score	Support
0	0.54	0.92	0.68	1,551
1	0.52	0.16	0.25	1,132
11-20mm	0.00	0.00	0.00	97
2	0.00	0.00	0.00	306
3	0.00	0.00	0.00	13
4	0.00	0.00	0.00	2
5	0.00	0.00	0.00	2
5-10mm	0.00	0.00	0.00	100
< 5mm	0.00	0.00	0.00	55
>20mm	0.00	0.00	0.00	71
Biopsy forceps	0.00	0.00	0.00	43
Black	0.00	0.00	0.00	9
Blue	0.00	0.00	0.00	4
Brown	0.00	0.00	0.00	10

Continued on next page

Table 4.5 – continued from previous page

Class	Precision	Recall	F1-score	Support
Cecum	0.00	0.00	0.00	33
Center	0.00	0.00	0.00	1,121
Center-left	0.10	0.14	0.12	831
Center-right	0.11	0.43	0.17	861
Colonoscopy	0.98	0.05	0.10	761
Gastroscopy	0.00	0.00	0.00	242
Green	0.00	0.00	0.00	3
Grey	0.00	0.00	0.00	16
Ileum	0.00	0.00	0.00	2
Injection needle	0.00	0.00	0.00	1
Lower-center	0.00	0.00	0.00	938
Lower-left	0.00	0.00	0.00	616
Lower-right	0.11	0.06	0.08	769
Metal clip	0.00	0.00	0.00	11
No	0.66	0.46	0.54	2,505
Oesophagitis	1.00	0.00	0.01	240
Orange	0.00	0.00	0.00	16
Pale Pink	0.00	0.00	0.00	1
Paris iia	0.00	0.00	0.00	95
Paris ip	0.00	0.00	0.00	101
Paris is	0.00	0.00	0.00	125
Pink	0.38	0.04	0.07	819
Polyp	0.25	0.96	0.39	311
Polyp snare	0.00	0.00	0.00	30
Purple	0.00	0.00	0.00	1
Pylorus	0.00	0.00	0.00	1
Red	0.29	0.27	0.28	629
Tube	0.00	0.00	0.00	218
Ulcerative colitis	0.00	0.00	0.00	250
Upper-center	0.00	0.00	0.00	944
Upper-left	0.00	0.00	0.00	764
Upper-right	0.11	0.38	0.16	729
White	0.00	0.00	0.00	520
Yellow	0.02	0.06	0.03	79

Continued on next page

Table 4.5 – continued from previous page

Class	Precision	Recall	F1-score	Support
Yes	0.59	0.98	0.74	1,778
Z-line	0.00	0.00	0.00	220
Brown	0.00	0.00	0.00	4
Grey	0.00	0.00	0.00	7
Purple	0.00	0.00	0.00	3
Accuracy			0.29	20,000
Macro average	0.09	0.08	0.06	20,000
Weighted average	0.30	0.29	0.24	20,000

Table 4.6: Simplified Classification Report on Test Set, only non-zero F_1 scores - Model trained on 20,000 question-answer pairs.

Class	Precision	Recall	F1-score	Support
0	0.54	0.92	0.68	1,551
1	0.52	0.16	0.25	1,132
Center-left	0.10	0.14	0.12	831
Center-right	0.11	0.43	0.17	861
Colonoscopy	0.98	0.05	0.10	761
No	0.66	0.46	0.54	2,505
Oesophagitis	1.00	0.00	0.01	240
Pink	0.38	0.04	0.07	819
Polyp	0.25	0.96	0.39	311
Red	0.29	0.27	0.28	629
Upper-right	0.11	0.38	0.16	729
Yellow	0.02	0.06	0.03	79
Yes	0.59	0.98	0.74	1,778

The classification reports are just a score on the top-level prediction of the model and do not give insight into why or how the model classifies labels. The next two subsections present different approaches to gaining insight into how the model predicts its answers. Having ways to present how the model works can aid in determining how much a user can trust the answers given by a system.

4.5.2 Visualizing Transition Scores

This subsection explores the transition scores given by the Alpaca-VQA model when it generates its answers. These scores can provide valuable insights into how the model processes and generates its responses. This insight can help understand which words or tokens in the input are most influential when generating an answer. Transition scores represent the attention or importance given to the different parts of the input sequence when generating new tokens in the output. Visualizing the transition scores can also reveal patterns and biases in the model's attention mechanism, like if the attention biases specific tokens.

As the Alpaca-VQA model uses the transformer architecture from HuggingFace, implementing the transition scores is as simple as Listing 4.1

```
# Generate output tokens
generation_output = model.generate(
    input_ids=input_ids,
    generation_config=generation_config,
    return_dict_in_generate=True,
    output_scores=True,
    max_new_tokens=256,
    temperature=0.1,
    top_p=0.75,
    top_k=40,
    num_beams=4,
)

# Compute transition scores
transition_scores = model.compute_transition_scores(
    generation_output.sequences,
    generation_output.scores,
    normalize_logits=True
)
```

Listing 4.1: Example of how to generate transition scores

In Table 4.7, examples of transition scores computed by Alpaca-VQA on various questions are displayed. These scores should be seen in conjunction with the distribution of classes in the training set, which was previously discussed and can be seen in Figure 4.4. As the model uses the connections made during training when predicting answers, the transition scores with high probability are usually from one of the majority classes

Transition Scores			
Question	Ground Truth	Predicted Token	Transition Score
Are there any anatomical landmarks in the image?	No	No	21.20%
Where in the image is the instrument?	Upper-left	Lower - left	50.49% 99.76% 40.16%
How many polyps are in the image?	0	0	90.38%
What type of procedure is the image taken from?	Laparoscopy	Comput ed tom ography	53.12% 47.85% 97.61% 99.41%
What type of polyp is present?	Paris is	Par ap lex is	19.25% 41.16% 71.34% 46.92%
What type of procedure is the image taken from?	Colonoscopy	Col on os copy	19.29% 99.80% 100.00% 99.90%

Table 4.7: Examples of transition scores computed by Alpaca-VQA. All questions were asked in relation to an image input.

in the training set. Examples of these, as shown in Table 4.7 are "Lower" and "0". It can also be seen how the Alpaca-VQA model splits up words into tokens, like how *"Computed Tomography"* gets split into tokens that can be used in other words. For example, the token *"comput"* be used in words like *"computer"*, *"computed"*, *"computing"*, and so on. By splitting the words into non-redundant tokens, the tokenizer keeps the dictionary of tokens efficient.

An interesting finding is that the predicted answers are not always from the fine-tuning dataset. In Table 4.7, the predicted answers *"Computed Tomography"* and *"Paraplexis"* does not occur in the dataset Alpaca-VQA is fine-tuned on, namely ImageCLEFmed-MEDVQA-GI-2023. It can also be seen in the case of *"Paraplexis"* that the model correctly

identifies the first token, *par*, although with a low score. Then the model follows up with the tokens it thinks finish the word by predicting *ed* with a 47,85% score. Most likely, *"Paraplexis"* was predicted by having a higher average score in the four beams than the correct answer. In the training data, there are many instances describing polyps using the Paris classification system. This makes the token *par* have a high probability of being the correct start token. However, as there are many labels that start with this token but end in a distinct class, the tokens that follow may not be as easily estimated.

The base model of Alpaca-VQA, the Stanford Alpaca, is a fine-tuned instruction-following model based on the LLaMA 7 billion parameters model. Upon investigation, *"Computed Tomography"* and *"Paraplexis"* are not part of this instruction-following fine-tuning dataset either. Therefore the knowledge of these medical subjects originates from the original LLaMA model. As noted in Table 2.1 on Page 48, the dataset used for pre-training the LLaMA model is disclosed and based on publicly available data, but not made public as the time of writing. Therefore it can not be known exactly where this information stems from, other than from the dataset used for pre-training the LLaMA 7B model.

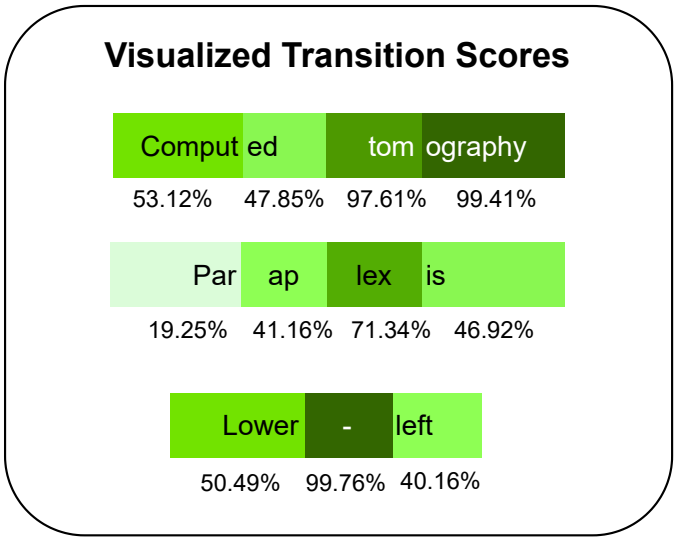


Figure 4.6: The visualized transition scores of Alpaca-VQA.

When using the transition scores calculated, the scores can be seen as a decimal value between zero and one. This value can be used as a weight when applying color to each individual token. Some of the predicted answers from Table 4.7 are visualized using this method in Figure 4.6.

As transition scores can be useful to get insight into how confident the model is when predicting a token, visualizing the scores can make them more intuitive for a user. Instead of only presenting the transition scores as percentages, colors are more often understood by non-technical users. By making more of the model presentable in an intuitive way, more users can gain insight into how it works and where it can be improved. The next subsection will explain another way to make models more transparent and presentable to humans. The experiment uses a proxy model explained by LIME.

4.5.3 Proxy model and LIME

This subsection will describe how the proxy model and LIME method were adapted to give insight into how the Alpaca-VQA model works.

Proxy model

The proxy model used to interpret the Alpaca-VQA model was trained on the predicted outputs of the LLM. These answers were generated using half of the available dataset, as shown in Figure 4.3. The rationale behind this was to see how well the Alpaca-VQA model performed on unseen data that could be used to fit the proxy model so that it mimics the underlying model. The available test set of 20,000 samples with predicted answers by the Alpaca-VQA model was used as a training and testing set for the proxy model. The dataset was split into roughly 70% training data and 30% evaluation data. Because a random split was used, the precise numbers used were 14,351 for the training set and 5,649 for the test set. The proxy model is a SDG classifier implemented by SciKit Learn [175]. The hyperparameters used when training the proxy model are shown in Table 4.8. Mostly the parameters used are the default for this implementation of the model, with the most notable difference being that the loss function is a modified Huber instead of the default *hinge*, giving a linear SVM. The modified Huber is a smooth loss function that brings tolerance to outliers as well as probability estimates, and was chosen because it gave the best results among the available loss functions.

A classification report was made when the SDG classifier, used as a proxy model, was fitted. This report can be seen in Table 4.9, and it can be seen that the model fits the Alpaca-VQA model with a satisfactory

SDG Classifier Hyperparameters	
Hyperparameter	Value
Loss:	Modified Huber
Penalty:	L_2
Alpha:	$1e^{-3}$
Random state:	42
Max iterations:	1000
Class weight:	Balanced

Table 4.8: Overview of the parameters used when training the proxy model.

accuracy of 84%. This score indicates that the proxy model will answer the same as the Alpaca-VQA model approximately 84% of the time, given the same questions and images.

Table 4.9: Proxy Model Classification Report

Class	Precision	Recall	F1-score	Support
0	0.87	0.98	0.92	783
1	0.00	0.00	0.00	111
2-3mm	0.00	0.00	0.00	0
3-5mm	0.99	1.00	0.99	99
Center-left	0.77	0.85	0.81	336
Center-right	0.94	0.74	0.83	1,059
Colonoscopy	0.00	0.00	0.00	9
Endoscopy	0.67	0.94	0.78	185
Laparoscopy	0.14	0.04	0.06	60
Lower-right	0.44	0.94	0.60	98
No	0.98	0.93	0.95	532
Oesophagitis	0.00	0.00	0.00	0
Pink	0.29	0.64	0.40	28
Polyp	0.97	0.77	0.86	343
Polypus	0.23	0.95	0.38	22
Red	0.90	0.97	0.93	178
Upper-right	0.86	0.84	0.85	784
Yellow	0.62	0.73	0.67	74

Continued on next page

Table 4.9 – continued from previous page

Class	Precision	Recall	F1-score	Support
Yes	0.97	0.96	0.96	893
Biopsy	0.00	0.00	0.00	1
Endoscopy	0.45	0.30	0.36	44
Yellow	0.17	1.00	0.29	10
Accuracy			0.84	5,649
Macro average	0.49	0.59	0.51	5,649
Weighted average	0.86	0.84	0.84	5,649

Explaining the proxy model with LIME

In order to interpret this proxy model, the XAI method LIME was fitted. Specifically, the TextExplainer module [176] was used to adapt to the proxy model, and the default parameters for TextExplainer were used. The TextExplainer uses as default an exponential kernel that uses a cosine distance metric and only uses words present in the text as explanations. This restriction helps speed up the explanation process, as only the relevant words are used, and not the whole vocabulary of the LLM.

In order to explain the proxy model, the LIME method first generates a neighborhood of the data by hiding features randomly from the explaining instance. Then the explaining instance uses this neighborhood to learn locally weighted linear models that explain each class. By learning these linear models, the LIME instance can interpret which word has a high impact on the predicted outcome.

Even though the stacked model design of this explanation pipeline can obscure the underlying Alpaca-VQA model, there are still some insights to be obtained. The LIME model is designed to give locally accurate interpretations of which parameters the underlying model uses. Therefore, the LIME model should be transparent to the proxy model. However, the proxy model has no transparency of which features it bases its decisions on. The similarities between the Alpaca-VQA model and the proxy model are that they both are trained on data from the same dataset and sample distribution, with the proxy model ignoring ground truth and using the predicted answer from Alpaca-VQA as its correct answer. Therefore, the proxy model does not give a locally accurate representation

of the inner workings of the underlying model and may weigh the inputs differently than the Alpaca-VQA model. Yet, as the proxy model is trained to mimic the underlying model and fits with an accuracy of 84%, there is still a possibility that the proxy model can give insight into how the LLM work and possibly reveal biases in the dataset.

The outcome of the proxy model being explained by LIME for one instance can be seen in Figure 4.7. Here it can be seen that the LIME instance predicts that the input question is most relevant in answering the question. In this figure LIME highlights words that have a high impact on the outcome. The four most probable classes are visualized to the left in the image. The class "Other" is how the LIME instance classifies the possibility of the predicted class not being in the top four. In the Figure 4.7, it can be seen that the class "Red" is thought to have the highest probability of being correct. In the middle of the figure, the two most probable classes are explained by visualizing the positive and negative impact different words have on the outcome. The figure shows that both "Red" and "Yellow" mostly depend on the same words and do not consider the image features. Additional examples of the proxy model being explained can be seen in Appendix A.

In this experiment with the proxy model explained by LIME, there is a likelihood that the model only considers the question and ignores the image features when predicting an answer. Since this is an indicator that the model detected some bias in the language used in the questions, the next experiment removes the image from the input entirely. Testing the Alpaca-VQA model trained on question-answer pairs, including image features, and removing the image features during testing, can provide insight into whether the proxy model exploited the same biases as the underlying model.

4.5.4 Language-only Alpaca-VQA model

As noted by the authors of the VQA 2.0 dataset [137], it could be observed that some VQA models in their experiments learned biases in the language of the question-answer pairs. This was demonstrated by removing the image altogether from the testing phase and letting the model predict answers only based on the question. Suppose these language-only models perform similarly to the models that also can see images. In that case, it

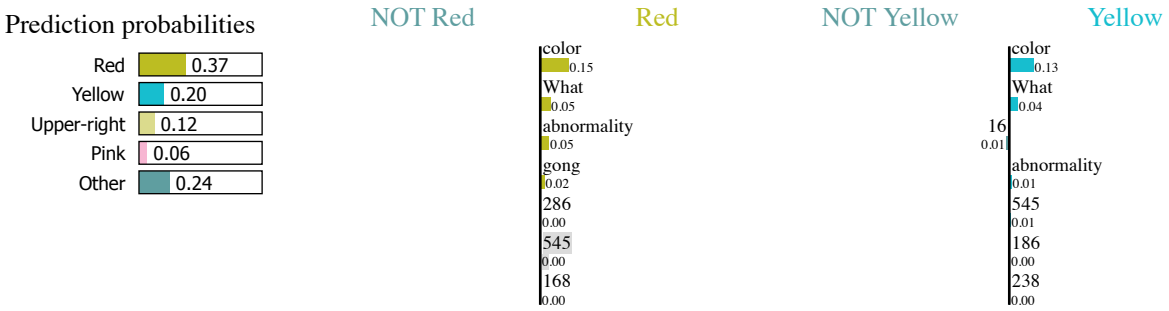


Figure 4.7: The proxy model explained by LIME. The correct answer is *Pink*, Alpaca-VQA predicted *Red*, and this is the explanation of the prediction.

is an indicator that the model has exploited severe language biases and mainly uses these when answering a question. Therefore, it pays too much attention to the language parts of the input and not using the image features. A model that has exploited these language biases will not ground its answers equally on the question and image, resulting in a reduced ability to answer correctly, given that the same question can have multiple images.

The classification report on 20,000 language-only test samples is listed in Table 4.10. The model used in this experiment is the same as in the main experiment, trained on text and images but with the images excluded during testing. As seen in the classification report, the language-only Alpaca-VQA model has an accuracy of 38% on this test set. Compared to the model that could also see images, the language-only model managed to correctly classify more samples without analyzing images.

Generally, a model trained on a larger dataset with more parameters is expected to perform better than one trained on a smaller dataset. This performance is because it has more data to learn from, making it able to capture the underlying patterns in the data more precisely. However,

there are certain situations where a model trained on a smaller dataset may perform better, and it is usually related to the quality of the dataset. Because the data quality is different, this performance discrepancy can be why the model tested on language-only performs better than on the complete VQA dataset on which it was trained on, including images.

One possible reason is that the smaller dataset, without image features, may have a higher signal-to-noise ratio, meaning the useful data patterns are more precise and distinct. A dataset with a higher signal-to-noise ratio makes it easier for the model to learn and generalize well to new data. In contrast, a larger dataset with more parameters may have more noise or variability, making it harder for the model to distinguish the relevant patterns. More noise may lead to overfitting or poor generalization performance. In this example, from the loss curve, there is a possibility that the Alpaca-VQA model did not generalize well to the available data, as the graph flattens relatively early in the training phase.

Another possible reason is that the samples with fewer features better represent the target distribution or application domain and may capture the key patterns and characteristics relevant to the task. In contrast, a larger dataset may contain more diverse or irrelevant data that can dilute the important signal and affect the performance of the model. One rationale is that the language-only model's higher accuracy can be explained by the fact that the image features are considered noise to the language model and that it can not utilize the information given.

Table 4.10: Language-Only Alpaca-VQA: Classification Report on 5,000 question-only samples.
Model trained on 20,000 question-answer pairs, including images. Tested on only questions.

Class	Precision	Recall	F1-score	Support
0	0.70	0.91	0.79	1,551
1	0.57	0.51	0.54	1,132
11-20mm	0.30	1.00	0.46	97
2	0.00	0.00	0.00	306
3	0.00	0.00	0.00	13
4	0.00	0.00	0.00	2
5	0.00	0.00	0.00	2
Continued on next page				

Table 4.10 – continued from previous page

Class	Precision	Recall	F1-score	Support
5-10mm	0.00	0.00	0.00	100
< 5mm	0.00	0.00	0.00	55
>20mm	0.00	0.00	0.00	71
Biopsy forceps	0.08	1.00	0.15	43
Black	0.00	0.00	0.00	9
Blue	0.00	0.00	0.00	4
Brown	0.00	0.00	0.00	10
Cecum	0.00	0.00	0.00	33
Center	0.00	0.00	0.00	1,121
Center-left	0.11	0.87	0.19	831
Center-right	0.00	0.00	0.00	861
Colonoscopy	0.76	1.00	0.86	761
Gastroscopy	0.00	0.00	0.00	242
Green	0.00	0.00	0.00	3
Grey	0.00	0.00	0.00	16
Ileum	0.00	0.00	0.00	2
Injection needle	0.00	0.00	0.00	1
Lower-center	0.00	0.00	0.00	938
Lower-left	0.00	0.00	0.00	616
Lower-right	0.00	0.00	0.00	778
Metal clip	0.00	0.00	0.00	11
No	0.73	0.38	0.50	2505
Oesophagitis	0.00	0.00	0.00	240
Orange	0.00	0.00	0.00	16
Pale Pink	0.00	0.00	0.00	1
Paris iia	0.30	1.00	0.46	95
Paris ip	0.00	0.00	0.00	101
Paris is	0.00	0.00	0.00	125
Pink	0.39	1.00	0.56	819
Polyp	0.00	0.00	0.00	311
Polyp snare	0.00	0.00	0.00	30
Purple	0.00	0.00	0.00	1
Pylorus	0.00	0.00	0.00	1
Red	0.00	0.00	0.00	629

Continued on next page

Table 4.10 – continued from previous page

Class	Precision	Recall	F1-score	Support
Tube	0.00	0.00	0.00	218
Ulcerative colitis	0.25	1.00	0.40	250
Upper-center	0.16	0.14	0.15	944
Upper-left	0.00	0.00	0.00	764
Upper-right	0.00	0.00	0.00	729
White	0.00	0.00	0.00	520
Yellow	0.00	0.00	0.00	79
Yes	0.71	0.80	0.75	1,778
Z-line	0.28	1.00	0.44	220
brown	0.00	0.00	0.00	4
grey	0.00	0.00	0.00	7
purple	0.00	0.00	0.00	3
Accuracy			0.38	20,000
Macro average	0.10	0.19	0.11	20,000
Weighted average	0.31	0.38	0.31	20,000

4.6 Discussion

This research project has aimed to explore how different explanatory methods could provide additional insights into how larger, more complex, and opaque models interpret the underlying data.
For the visual domain, the original question was:

- Will the answers given by a VQA system be more intuitively explained with additional locally accurate image descriptions?

As the FLEX-VQA model did not materialize in results, the visual explanation was done by fitting a proxy model, which was explained by LIME. Considering that the proxy model and LIME were not designed to describe an image but rather to use text as input data, the image features were encoded as text. Therefore, the essential elements could be highlighted by the TextExplainer method in LIME. In subsection 4.5.3, it was shown that the Alpaca-VQA model on this specific dataset did not evaluate the visual features as necessary compared to the question.

Still, the experiment aimed to investigate if these visual highlights would intuitively bring additional important information. The visual representation of the input features of the proxy model was highlighted using a locally accurate XAI method, LIME. These features made it more straightforward to investigate further, leading to the experiment with the language-only Alpaca-VQA model. As concluded in subsection 4.5.4, the model tested on language-only confirmed that the model mostly paid attention to the questions and not evaluating the image features. As this finding was consistent with the proxy model, it provided valuable insight into how the model used the available data. Even though a stacked proxy model design can obscure the underlying model's inner workings, it proved helpful in investigating how the Alpaca-VQA model may have used the input data.

In the linguistic domain, the initial research questions were:

- To which degree can an LLM fine-tuned on a new modality bring new insights from its pertaining?
- What insights can additional explanatory methods bring from an LLM after training is complete?

The first question was demonstrated in the experiment by visualizing the attention scores in subsection 4.5.2. The Alpaca-VQA model used knowledge from the underlying LLaMA model when predicting answers. As these answers from previous training did not occur in the test set, the model did not answer the question correctly. However, the model understood how the question was structured and appropriately responded with a medically relevant term, which could be correct given a different input image. This suggests that the model's *a priori* general knowledge from pre-training could potentially enhance responses when tested on more open-ended tasks or questions. It was fine-tuned to provide a single answer to facilitate the evaluation of the Alpaca-VQA model. This contrasts how many LLMs are trained as conversation systems that generate longer sentences or paragraphs. As the Alpaca-VQA uses LoRA weights for fine-tuning, it performs as well as the original Stanford Alpaca model on the tasks it was initially trained on. This allows the Alpaca-VQA model to work as an extension on an already capable LLM. Consequently, Alpaca-VQA can handle more free-form responses

when given free-form questions, where the knowledge from the pre-training can contribute to a better answer.

To investigate the second research question, the proxy model and visualization of the Alpaca-VQA model were implemented after the LLM completed its fine-tuning. Yet, they proved instrumental in interpreting the decision-making process of the underlying model. Consequently, the conducted experiments in this study shed light on how post-hoc explanatory models can aid in comprehending larger and more complex models without compromising the accuracy that the larger model can provide.

As explored in this chapter, visualizing transition scores and having proxy models explained by XAI methods like LIME can give insight into how the larger underlying model handles the data. Even though one extra model is stacked on top of the explainability pyramid, it can still provide an understanding of how the model functions.

An analogy of this proxy model is how most computer vision systems tackle predicting a person's mood by looking at their face. Instead of trying to analyze every facial muscle that makes up the expression, the system looks at the resulting expression when predicting the mood. Another example may be how humans can determine the species of trees. As taking a DNA sample every time is relatively resource-intensive, most people use features such as the shape of the leaves, the color, and the structure of the bark. In this example, the LLM represents the DNA, the proxy model represents the outside of the three, and LIME represents which features to pay attention to and why these are important. Using a proxy model, the LLM can make a prediction, and some of the details can be abstracted to more high-level features, which can be explained by intuitive methods like LIME.

In the experiment with the proxy model explained by LIME, like in Figure 4.7, it was discovered that the proxy model primarily evaluated the question when predicting an answer. This finding was not definite proof that Alpaca-VQA did not evaluate the image features. Still, it highlighted the possibility that the data could be exploited using biases in the linguistic part of the dataset.

The Alpaca-VQA model, trained on questions and images, was blinded during testing to investigate this finding further. By having the model trained on seeing images, not receiving images, the possibility of it not

using the image features could be explored. The model achieved higher accuracy when testing this language-only data than the one tested with questions and images. This finding suggests that the Alpaca-VQA model trained on the current dataset has learned to exploit linguistic biases, possibly combined with taking advantage of the dataset used being unbalanced.

The transition scores can be calculated by exploring how the LLM predicts the next token. In the experiments in subsection 4.5.2, the transition scores were visualized by studying how these scores progress throughout the predicted answer. A user can gain insight into how the model samples from the available distribution of tokens. Further insights about how the model interprets the dataset can be gained by complementing the resulting transition scores with a proxy model. By leveraging additional models and explanation methods, a user can get a combined interpretation of how larger, more complex, and opaque models may use the available data when predicting outputs. Another advantage of these additional models and explanation methods is that they require much fewer computing resources than training the primary model. This makes these complimentary explanation models have a little-to-no impact during inference, compared to the processing time of an LLM. These explanation methods will, therefore, add additional information on how the model computes an answer, in combination with details on how the model may use the data. These insights can facilitate a user in gaining important information on how the model solves the task.

From the classification reports, it could be seen that the Alpaca-VQA model had relatively low accuracy. During the generation of answers, as seen when visualizing transition scores, the model sometimes predicts answers not present in the fine-tuning dataset. This demonstrates the ability of the LoRA implementation to resist catastrophic forgetting, letting the model continue to learn new features without forgetting what has already been learned. This prediction of answers not present in the test dataset may be why the model does not achieve higher accuracy. As discussed earlier, another reason for the low accuracy may be that the image features encoded into the input text prompt do not bring more signal than noise to the model. The language-only Alpaca-VQA model did achieve higher accuracy than the one using image features, hinting that the visual elements bring more noise than value into the model.

Regarding the FLEX-VQA model, although no experiments were conducted using this model, the method still provides a novel take on the VQA task. Using labeled feature maps from a CNN to generate supplementary captions to an input image would allow for an explanation grounded in locally accurate features. As the goal of VQA is to answer a question as accurately as possible, it only gives insight into a single instance of how the model interprets the task. The XAI method LIME also explains a single, locally accurate sample but can give a global insight into a model by explaining multiple single instances. The way LIME achieves this global explanation is by utilizing a submodular pick algorithm to isolate instances with non-redundant features. These non-redundant instances are explained by LIME, and combined, they give a more global understanding of how the underlying model works. By having the FLEX-VQA model both answer the specific question and supplement with a description of the image, using the locally accurate image features, the model can achieve both a locally accurate and a more global explanation of the image. As the image features are used to make the description, the model uses locally accurate image features to create a descriptive text using natural language that describes the image in a broader way than just answering the specific answer. Given both the answer and the description of a single instance, a user can achieve a complete picture of how the model interprets the task.

4.7 Summary

This chapter presents the results of examining the Alpaca-VQA model, an LLM designed to answer questions about the contents of images. An investigatory experiment was conducted to see how the model responded to the available dataset. The dataset was modified to remove the majority class to make more of the samples relevant to the specific task.

The Alpaca-VQA model was trained on the larger dataset of 20,000 samples. From the classification report, it could be seen that the model had learned biases in the dataset and potentially exploited biases in the questions. These findings were discovered using explanatory post-hoc methods. Specifically, these methods were a proxy model explained by LIME and visualizing transition scores of the LLM. These additional methods gave supplementary information not present in the original

model, which helped provide an understanding of how the model may exploit biases in the dataset.

To explore the possibility of linguistic biases in the model, which could lead to inaccurate or unfair results, a language-only version of the model was tested, and compared its performance to the original version that used both images and language.

The key finding of this research is that larger and more complex models, like an LLM, can be explained by smaller methods added after the primary model has completed training. These additional models add no significant resources use or compute time during inference but provide valuable insights into the model. In addition, these supplementary models do not change how the larger, more complex model works. Therefore, these models can combine complex methods with layers of explanation that bring valuable insights with no cost to the accuracy of the primary model.

Chapter 5

Conclusions

5.1 Summary

This thesis has explored the realm of deep neural networks and the potential ways these can be explained. This work has focused on multimodal VQA models and how these can be explained in both the visual and linguistic domains. The VQA task has the benefit of being interactive so that a user can input an image and ask a question regarding the contents of the image. Therefore, the model that processes these requests must be a multimodal model that can comprehend both text and images and fuse these to give a correct answer.

Two distinct frameworks were introduced and discussed to provide better explanations and transparency to VQA methods. The first is named FLEX-VQA and combines a robust VQA method with the FLEX framework to give locally accurate image descriptions alongside the response to the user's question. This method uses natural language to explain locally accurate and faithful image features. An image and corresponding caption are inputted during training and used to find co-occurrences between feature maps of the image network and words in the caption. During interference, the system uses the gradient of the predicted class backpropagated in the CNN. The feature maps found in this backpropagation are labeled with the word of highest co-occurrence from the training set and used to generate a natural language description of the image. This ensures that the image description is based on relevant words associated with the predicted class and only features present in the image. This description, faithful to the underlying model combined with

a VQA task, makes an interactive system where users can ask questions and get informative answers. As this method did not have results, it was discussed in depth so that future works can use it as motivation for new explainable methods.

The second method described in this work is the Alpaca-VQA model. This is an LLM in the LLaMA family, trained to interpret images and answer questions. As language models by design are usually not multimodal, the image features were extracted using a CNN, which then was encoded into the input text. To make this model even more computationally efficient than originally, a LoRA implementation is used. This freezes the weights of the LLM and only adds a smaller update matrix during training. Using this optimization, the model can learn new tasks without forgetting what it was previously trained on. During interference, the input was put through the original weights and the new update matrix, and the merged result was given as the output. The dataset used was an extensive collection of images from the gastrointestinal tract, paired with questions and answers to each image. LLMs have no intrinsic or intuitive ways to explain how they interpret the input data or do their reasoning. As these transformer-based architectures continue to outperform other methods and therefore get implemented in new systems, it is vital to be able to interpret their decisions.

This work investigates how smaller and explainable supplementary methods can be adapted to a larger model to get a more nuanced understanding. The two methods experimented with are visualizations of transition scores and a proxy model explained by LIME. The proxy model gives insight into which parts of the input data may contribute most when predicting an answer, and the visualization of transition scores provides insight into the model's certainty when estimating a new token. Transition scores extracted during the generation of the LLMs response give insight into how well the model predicts the token's fit in the sequence. By visualizing these scores, based on the transformer self-attention, a user can get a more intuitive insight into how a sentence is generated. In the experiments, the visualizations provided intuitive insights into the way the model uses prior knowledge from pre-training while providing an assessment of the model's reliability in generating responses.

The other method used to explain the Alpaca-VQA model was a

proxy model trained to simulate the underlying LLM. As LLMs are computationally expensive to run and often challenging to interpret, a model mimicking how the model responds while being explained by LLM was developed. This model was trained on the responses that the Alpaca-VQA model had given on several image-question-pairs and used the TextExplainer method in LIME to highlight essential words in making the given prediction. Although the design of a stacked proxy model offers the possibility to obscure the inner workings of the underlying model, it proved useful when investigating how the Alpaca-VQA model used input data.

With the two explainable post-hoc methods adapted to the Alpaca-VQA model, they could together bring valuable insight into how the underlying model may use the input data. Insights gained from these models indicated that the Alpaca-VQA model seemed to have discovered a bias in the dataset and only evaluated the questions when responding, mostly ignoring the encoded image features. A language-only version of the model was tested to explore if the proxy model and transition scores had uncovered these linguistic biases found by the Alpaca-VQA model. The Alpaca-VQA model from the main experiment, trained on both images and text, was given the same test data but with the image features removed. As implied by the supplementary explanatory models, the Alpaca-VQA model utilized biases in the language. This was proven by the language-only model's higher accuracy than the one tested on images and text. As biases in datasets are common, they are sometimes hard to uncover and can lead to inaccurate or unfair results. As this finding could have been difficult to discover without these additional explanatory models, they have proven to be valuable in developing a complex system with a model that is hard to interpret at the center without sacrificing accuracy.

Through conducting experiments for this study, insights have been gained into the usefulness of post-hoc explanatory models in comprehending larger and more intricate models while maintaining the accuracy that the larger model can offer.

5.2 Main Contributions

The goal of this work has been to investigate to what extent explanatory models in different domains can provide additional insights into the underlying data, specifically in the VQA task. The experiments in this work have proven the ability of post-hoc explanatory models to provide valuable insight into the underlying model with no cost to the explained model's accuracy. The proxy model presents a useful understanding of how the LLM may work, even though it is not transparent to the underlying model but instead emulates its behavior. The visualization of transition scores provides an intuitive description of how LLMs predicts tokens in a sequence.

In this work, it has been studied how an LLM can be adapted to a new modality while preserving the knowledge from the pre-training. This fine-tuned model has then been explained by methods adapted after the training is complete. These smaller, more explanatory methods have given valuable insights and indicators of how the LLM works on the given dataset. By attaching additional explanatory methods to a fully trained model, the combined solution can leverage all of the initial benefits and accuracy while still providing users with intuitive justifications for its answers. Models that are easier for a user to understand make them more functional and effective for further development.

The key finding of this research is that larger and more complex models, like an LLM, can be explained by smaller methods added after the primary model has completed training. These additional models add no significant resources use or compute time during inference but provide valuable insights into the model. In addition, these supplementary models do not change how the larger, more complex model works. Therefore, these models can combine complex methods with layers of explanation that bring valuable insights with no cost to the accuracy of the primary model.

In the experiments carried out in this work, the additional locally accurate explanations proved valuable insights into understanding the model's predictions. Visualizations of transition scores and the proxy model explained through LIME provided valuable supplementary information that the LLM initially lacked, contributing to a better understanding of its usage and reliability.

5.3 Limitations and Future Work

As the scope of this project limits what can be explored in this work, not every aspect of the methods presented and discussed can be examined. Therefore, in this subchapter, some parts of the limitations of the techniques studied in this work are discussed, and ways to overcome these limitations and further investigate these methods in future research are suggested.

5.3.1 FLEX-VQA

The FLEX-VQA model presented in this work did not provide any results due to technical issues outside the scope of this work. However, the method offers a solution for a better understanding of the methods used in critical computer vision systems deployed today.

Some initial issues have been solved when developing this model, but some remain to be overcome. The most prominent remaining problems have been discussed in subsection 3.1.5. Most notable is to make the FLEX framework non-dependent on the machine learning framework Caffe. As this framework is no longer developed, its support for modern hardware and complimentary software dependencies is decreasing. The FLEX framework still possesses the potential of making systems dependent on CNNs more transparent to its users. Therefore, future research is encouraged to develop this method further and other variants of locally accurate post-hoc methods.

In future research, the FLEX framework should not build its dependence on Caffe for this method to be implemented efficiently and with versatility. This adaptation can be made in a multitude of ways, where the main goal would be to decouple the CNN feature extraction from FLEX. Modern machine learning frameworks have properties that easily traverse networks and list their feature maps. This allows for a more manageable implementation of FLEX that can be agnostic of the specific CNN model used.

One restriction of the current implementation of FLEX is that it is based on explaining CNNs. Most computer vision systems today rely on CNNs as the algorithms window to the outside world. Transformer-based vision systems are, however, gaining popularity and are used instead of

CNNs in many modern systems. As the current state of FLEX depends on labeling CNN feature maps, it does not currently work with other vision algorithms. The FLEX framework may still be adapted in future work to the transformer architecture, as the main requirement is that it can map specific features in the input to particular parts of the internal features. Various methods have been developed to explain transformers more transparently. Many of these rely on a GradCAM method or similar methods based on heat-maps of gradients [163, 164]. These heat-map-based methods can, in theory, be used to map specific labels to the gradients and relevance matrices throughout the computation, resulting in a locally accurate representation of these matrices. By labeling these features with co-occurring words, FLEX can be used to generate descriptive captions using the locally accurate features of the transformer.

A limitation that could be solved by adapting FLEX to a transformer is more accessible transfer learning compared to RNNs. Currently, the FLEX framework needs a dataset containing image descriptions and a single class label. The class label is used to train the CNN to predict the class, and the image description is used to train the LSTMs to output the locally accurate description. Not many datasets are available with both these features, so the transformer could be pre-trained using image descriptions to label its gradient features. FLEX can then label features in the gradients before the transformer is fine-tuned on the specific task. Although the development of this method uses FLEX in between transfer learning and does not become fully post-hoc, it does not affect the fine-tuned model. Therefore, it still contains the same benefits of allowing the finished model to train and interpret uninterrupted while providing transparency.

5.3.2 Alpaca-VQA

One constraint of the implemented Alpaca-VQA model is that it uses a CNN not pre-trained for the given task. It may not be a robust approach depending on the fact that the feature maps of the pre-trained classes from ImageNet are similar to the ones observed in separate tasks. Future work can include fine-tuning the CNN used for the specific task, making the extracted image feature more relevant. The CNN used in future works is encouraged to experiment with object detection models that can encode ROI or bounding boxes so that the LLM can learn to interpret where the

objects are in the image.

In the experiments conducted in this work, the goal has not been to train a state-of-the-art LLM, but rather explore how it can be explained. This is reflected in the relatively low accuracy for Alpaca-VQA on the given dataset. However, in future experiments, LLM could be extended to make it more beneficial to the end user. An interesting future work could include an LLM pre-trained on domain-specific data, like a medical dataset, before fine-tuning to include images. This way, the system user could ask questions about an image or simply text-only questions. Therefore, the LLM could be used as a multimodal personal assistant in a domain-specific setting.

The Alpaca-VQA model used in this work is trained to output only the predicted answer to a question to facilitate evaluation. However, this does not use the ability of LLM to construct sentences in natural language. Because the model is pre-trained on instructions-following tasks, it can still generate longer responses. Techniques such as self-instruct, also used when generating data for the Stanford Alpaca model, can make datasets with natural language answers. Future work can use these methods to create a dataset containing descriptive information in natural language based on factually correct information.

5.3.3 Explainable Methods

The experiments in this work have demonstrated how smaller, more transparent models can be adapted to explain a more complex and opaque model. However, the models described in this thesis have some limitations. One of these is that the visualization of transition scores and self-attention are unreliable sources of insight into how the model works. Some previous works have investigated how attention may not be an interpretable nor accurate method to explain methods based on attention [167, 168]. As presented in this work, these features can still bring valuable insights into how the model work and interprets the data. Therefore, a future experiment worth investigating would be a larger user study, asking end users whether attention or transition scores make a model more intuitive.

The proxy model used in this work had the limitation that it was not transparent how the Alpaca-VQA worked. It was designed to simulate the

behavior of the LLM instead of giving direct insight into how the underlying model functioned. Yet, during experimentation, it provided valuable insights that helped discover bias in the Alpaca-VQA model. As more complex models get higher accuracies at the cost of transparency, future work is needed to develop models with intrinsic explainability or methods that can be attached to a larger model post-hoc.

In summary, this thesis has explored powerful AI methods on the VQA task using CNNs and LLMs, highlighting the inherent challenges in comprehending their underlying reasoning. This work has made noteworthy progress by investigating various techniques for increased transparency. However, it is important to acknowledge that there is still much more to be done in the field of XAI as research strives towards achieving a deeper understanding of these complex systems.

Bibliography

[1] Arun Das and Paul Rad. *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey*. June 22, 2020. DOI: 10.48550/arXiv.2006.11371. arXiv: 2006.11371 [cs]. URL: http://arxiv.org/abs/2006.11371 (visited on 05/29/2023). preprint.

[2] Kamran Alipour et al. *A Study on Multimodal and Interactive Explanations for Visual Question Answering*. Mar. 1, 2020. DOI: 10.48550/arXiv.2003.00431. arXiv: 2003.00431 [cs]. URL: http://arxiv.org/abs/2003.00431 (visited on 05/25/2023). preprint.

[3] Sandareka Wickramanayake, Wynne Hsu, and Mong Li Lee. "FLEX: Faithful Linguistic Explanations for Neural Net Based Model Decisions". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (01 July 17, 2019), pp. 2539–2546. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33012539. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4100 (visited on 09/12/2022).

[4] Rohan Taori et al. *Stanford CRFM*. Alpaca: A Strong, Replicable Instruction-Following Model. URL: https://crfm.stanford.edu/2023/03/13/alpaca.html (visited on 04/07/2023).

[5] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA Model*. Tatsu's shared repositories, Apr. 7, 2023. URL: https://github.com/tatsu-lab/stanford_alpaca (visited on 04/07/2023).

[6] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. Feb. 27, 2023. DOI: 10.48550/arXiv.2302.13971. arXiv: 2302.13971 [cs]. URL: http://arxiv.org/abs/2302.13971 (visited on 03/28/2023). preprint.

[7] Ehud Reiter. "A Structured Review of the Validity of BLEU". In: *Computational Linguistics* 44.3 (Sept. 1, 2018), pp. 393–401. ISSN: 0891-2017. DOI: 10.1162/coli_a_00322. URL: https://doi.org/10.1162/coli_a_00322 (visited on 05/25/2023).

[8] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556. arXiv: 1409.1556 [cs]. URL: http://arxiv.org/abs/1409.1556 (visited on 10/17/2022). preprint.

[9] Yang Gao et al. *Compact Bilinear Pooling*. Apr. 11, 2016. DOI: 10.48550/arXiv.1511.06062. arXiv: 1511.06062 [cs]. URL: http://arxiv.org/abs/1511.06062 (visited on 11/28/2022). preprint.

[10] Yusuke Hirota, Yuta Nakashima, and Noa Garcia. "Gender and Racial Bias in Visual Question Answering Datasets". In: *2022 ACM Conference on Fairness, Accountability, and Transparency*. June 21, 2022, pp. 1280–1292. DOI: 10.1145/3531146.3533184. arXiv: 2205.08148 [cs]. URL: http://arxiv.org/abs/2205.08148 (visited on 01/17/2023).

[11] OpenAI. *GPT-4 Technical Report*. Mar. 15, 2023. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs]. URL: http://arxiv.org/abs/2303.08774 (visited on 03/16/2023). preprint.

[12] *ChatGPT*. URL: https://chat.openai.com (visited on 04/07/2023).

[13] Dario Amodei et al. *Concrete Problems in AI Safety*. July 25, 2016. DOI: 10.48550/arXiv.1606.06565. arXiv: 1606.06565 [cs]. URL: http://arxiv.org/abs/1606.06565 (visited on 05/27/2023). preprint.

[14] Richard Ngo, Lawrence Chan, and Sören Mindermann. *The Alignment Problem from a Deep Learning Perspective*. Feb. 22, 2023. DOI: 10.48550/arXiv.2209.00626. arXiv: 2209.00626 [cs]. URL: http://arxiv.org/abs/2209.00626 (visited on 05/27/2023). preprint.

[15] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 03/16/2023).

[16] Zachary C. Lipton. *The Mythos of Model Interpretability*. Mar. 6, 2017. DOI: 10.48550/arXiv.1606.03490. arXiv: 1606.03490 [cs, stat]. URL: http://arxiv.org/abs/1606.03490 (visited on 05/26/2023). preprint.

[17] *Pause Giant AI Experiments: An Open Letter*. Future of Life Institute. URL: https://futureoflife.org/open-letter/pause-giant-ai-experiments/ (visited on 05/27/2023).

[18] *REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. Apr. 21, 2021. URL: https://eur-lex.europa.eu/resource.html?uri=cellar:e0649735-a372-11eb-9585-01aa75ed71a1.0001.02/DOC_1&format=PDF (visited on 05/27/2023).

[19] OAR US EPA. *Greenhouse Gas Emissions from a Typical Passenger Vehicle*. Jan. 12, 2016. URL: https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle (visited on 05/26/2023).

[20] *Nvidia-A100-Datasheet*. URL: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet.pdf (visited on 05/27/2023).

[21] *Lavt klimagassutslipp knyttet til norsk strømforbruk i 2022 - NVE*. URL: https://www.nve.no/nytt-fra-nve/nyheter-energi/lavt-klimagassutslipp-knyttet-til-norsk-stroemforbruk-i-2022/ (visited on 05/26/2023).

[22] *Hva påvirker utslipp til luft fra veitrafikk?* ssb.no. Aug. 14, 2017. URL: https://www.ssb.no/natur-og-miljo/artikler-og-publikasjoner/hva-pavirker-utslipp-til-luft-fra-veitrafikk (visited on 05/26/2023).

[23] Chansup Byun et al. "Benchmarking Data Analysis and Machine Learning Applications on the Intel KNL Many-Core Processor". In: *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. 2017 IEEE High Performance Extreme Computing Conference (HPEC). Sept. 2017, pp. 1–6. DOI: 10.1109/HPEC.2017.8091067.

[24] Norman Jouppi et al. "Motivation for and Evaluation of the First Tensor Processing Unit". In: *IEEE Micro* 38.3 (May 2018), pp. 10–19. ISSN: 1937-4143. DOI: 10.1109/MM.2018.032271057.

[25] Anne C. Elster and Tor A. Haugdahl. "Nvidia Hopper GPU and Grace CPU Highlights". In: *Computing in Science & Engineering* 24.2 (Mar. 2022), pp. 95–100. ISSN: 1558-366X. DOI: 10.1109/MCSE.2022.3163817.

[26] David Kasperek, Michal Podpora, and Aleksandra Kawala-Sterniuk. "Comparison of the Usability of Apple M1 Processors for Various Machine Learning Tasks". In: *Sensors* 22.20 (20 Jan. 2022), p. 8005. ISSN: 1424-8220. DOI: 10.3390/s22208005. URL: https://www.mdpi.com/1424-8220/22/20/8005 (visited on 05/28/2023).

[27] John McCarthy et al. "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955". In: *AI Magazine* 27.4 (4 Dec. 15, 2006), pp. 12–12. ISSN: 2371-9621. DOI: 10.1609/aimag.v27i4.1904. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904 (visited on 03/10/2023).

[28] S.L. Andresen. "John McCarthy: Father of AI". In: *IEEE Intelligent Systems* 17.5 (Sept. 2002), pp. 84–85. ISSN: 1941-1294. DOI: 10.1109/MIS.2002.1039837.

[29] Stanford University. *Ancient Myths Reveal Early Fantasies about Artificial Life*. Stanford News. Feb. 28, 2019. URL: https://news.stanford.edu/2019/02/28/ancient-myths-reveal-early-fantasies-artificial-life/ (visited on 04/28/2023).

[30] Mary Wollstonecraft Shelley. *Frankenstein, or, The Modern Prometheus*. Knopf: New York, 1992.

[31] Karel Čapek. *R.U.R. (Rossum's Universal Robots)*. 2004th ed. London: Penguin Books. URL: https://www.gutenberg.org/files/59112/59112-h/59112-h.htm.

[32] Stanley Kubrick, director. *2001: A Space Odyssey*. scriptwriter Stanley Kubrick et al. Adventure, Sci-Fi. Feb. 28, 1969.

[33] Fritz Lang, director. *Metropolis*. scriptwriter Thea von Harbou et al. Drama, Sci-Fi. Feb. 28, 1927.

[34] George Barbastathis, Aydogan Ozcan, and Guohai Situ. *On the Use of Deep Learning for Computational Imaging*. URL: https://opg.optica.org/optica/fulltext.cfm?uri=optica-6-8-921&id=416103 (visited on 04/28/2023).

[35] Matthew B. Hoy. "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants". In: *Medical Reference Services Quarterly* 37.1 (Jan. 2, 2018), pp. 81–88. ISSN: 0276-3869. DOI: 10.1080/02763869.2018.1404391. pmid: 29327988. URL: https://doi.org/10.1080/02763869.2018.1404391 (visited on 04/28/2023).

[36] Som S. Biswas. "Role of Chat GPT in Public Health". In: *Annals of Biomedical Engineering* 51.5 (May 1, 2023), pp. 868–869. ISSN: 1573-9686. DOI: 10.1007/s10439-023-03172-7. URL: https://doi.org/10.1007/s10439-023-03172-7 (visited on 04/28/2023).

[37] Shijun Wang and Ronald M. Summers. "Machine Learning and Radiology". In: *Medical Image Analysis* 16.5 (July 1, 2012), pp. 933–951. ISSN: 1361-8415. DOI: 10.1016/j.media.2012.02.005. URL: https://www.sciencedirect.com/science/article/pii/S1361841512000333 (visited on 04/28/2023).

[38] James H. Thrall et al. "Artificial Intelligence and Machine Learning in Radiology: Opportunities, Challenges, Pitfalls, and Criteria for Success". In: *Journal of the American College of Radiology*. Data Science: Big Data Machine Learning and Artificial Intelligence 15 (3, Part B Mar. 1, 2018), pp. 504–508. ISSN: 1546-1440. DOI: 10.1016/j.jacr.2017.12.026. URL: https://www.sciencedirect.com/science/article/pii/S154614401731671X (visited on 04/28/2023).

[39] Alan M. Turing. *Systems of Logic Based on Ordinals*. 1938. URL: https://web.archive.org/web/20121023103503/https://webspace.princeton.edu/users/jedwards/Turing%20Centennial%202012/Mudd%20Archive%20files/12285_AC100_Turing_1938.pdf (visited on 04/29/2023).

[40] Alonzo Church. "An Unsolvable Problem of Elementary Number Theory". In: *American Journal of Mathematics* 58.2 (1936), pp. 345–363. ISSN: 0002-9327. DOI: 10.2307/2371045. JSTOR: 2371045. URL: https://www.jstor.org/stable/2371045 (visited on 04/29/2023).

[41] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1, 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259 (visited on 04/29/2023).

[42] Gualtiero Piccinini. "The First Computational Theory of Cognition: McCulloch and Pitts's "A Logical Calculus of the Ideas Immanent in Nervous Activity"". In: *Neurocognitive Mechanisms: Explaining Biological Cognition*. Ed. by Gualtiero Piccinini. Oxford University Press, Nov. 12, 2020. ISBN: 978-0-19-886628-2. DOI: 10.1093/oso/9780198866282.003.0006. URL: https://doi.org/10.1093/oso/9780198866282.003.0006 (visited on 04/29/2023).

[43] Luciano Floridi. "AI and Its New Winter: From Myths to Realities". In: *Philosophy & Technology* 33.1 (Mar. 1, 2020), pp. 1–3. ISSN: 2210-5441. DOI: 10.1007/s13347-020-00396-6. URL: https://doi.org/10.1007/s13347-020-00396-6 (visited on 05/05/2023).

[44] John Haugeland. *Artificial Intelligence: The Very Idea*. MIT Press, Jan. 6, 1989. 303 pp. ISBN: 978-0-262-58095-3. Google Books: AL1NEAAAQBAJ.

[45] Marta Garnelo and Murray Shanahan. "Reconciling Deep Learning with Symbolic Artificial Intelligence: Representing Objects and Relations". In: *Current Opinion in Behavioral Sciences*. Artificial Intelligence 29 (Oct. 1, 2019), pp. 17–23. ISSN: 2352-1546. DOI: 10.1016/j.cobeha.2018.12.010. URL: https://www.sciencedirect.com/science/article/pii/S2352154618301943 (visited on 04/29/2023).

[46] Leslie G Valiant. "Knowledge Infusion: In Pursuit of Robustness in Artificial Intelligence". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science* (2008).

[47] Daniel Kahneman. *Thinking, Fast and Slow*. 1st edition. New York: Farrar, Straus and Giroux, Apr. 2, 2013. 499 pp. ISBN: 978-0-374-53355-7.

[48] Sebastian Raschka. *Chapter 1: Introduction to Machine Learning and Deep Learning*. Sebastian Raschka, PhD. Apr. 5, 2020. URL: https://sebastianraschka.com/blog/2020/intro-to-dl-ch01.html (visited on 03/16/2023).

[49] Evelyn Fix and J. L. Hodges. "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties". In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247. ISSN: 0306-7734. DOI: 10.2307/1403797. JSTOR: 1403797. URL: https://www.jstor.org/stable/1403797 (visited on 10/28/2022).

[50] T. Cover and P. Hart. "Nearest Neighbor Pattern Classification". In: *IEEE Transactions on Information Theory* 13.1 (Jan. 1967), pp. 21–27. ISSN: 1557-9654. DOI: 10.1109/TIT.1967.1053964.

[51] Mikhail Belkin et al. "Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off". In: *Proceedings of the National Academy of Sciences* 116.32 (Aug. 6, 2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1903070116 (visited on 05/05/2023).

[52] Preetum Nakkiran et al. "Deep Double Descent: Where Bigger Models and More Data Hurt". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (Dec. 2021), p. 124003. ISSN: 1742-5468. DOI: 10.1088/1742-5468/ac3a74. URL: https://dx.doi.org/10.1088/1742-5468/ac3a74 (visited on 05/05/2023).

[53] Mayu Sakurada and Takehisa Yairi. "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction". In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. MLSDA'14. New York, NY, USA: Association for Computing Machinery, Dec. 2, 2014, pp. 4–11. ISBN: 978-1-4503-3159-3. DOI: 10.1145/2689746.2689747. URL: https://dl.acm.org/doi/10.1145/2689746.2689747 (visited on 05/05/2023).

[54] Steven Flores. *Variational Autoencoders Are Beautiful | Blogs*. Apr. 15, 2019. URL: https://www.compthree.com/blog/autoencoder/ (visited on 04/30/2023).

[55] Y Reddy, Viswanath Pulabaigari, and Eswara B. "Semi-Supervised Learning: A Brief Review". In: *International Journal of Engineering & Technology* 7 (Feb. 9, 2018), p. 81. DOI: 10.14419/ijet.v7i1.8.9977.

[56] David Berthelot et al. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html (visited on 05/27/2023).

[57] *Amazon Mechanical Turk Documentation*. URL: https://docs.aws.amazon.com/mturk/index.html (visited on 05/05/2023).

[58] Raphael Hoffmann et al. "Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. ACL-HLT 2011. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 541–550. URL: https://aclanthology.org/P11-1055 (visited on 05/05/2023).

[59] Zirui Wang et al. *SimVLM: Simple Visual Language Model Pretraining with Weak Supervision*. May 15, 2022. DOI: 10.48550/arXiv.2108.10904. arXiv: 2108.10904 [cs]. URL: http://arxiv.org/abs/2108.10904 (visited on 05/05/2023). preprint.

[60] Fuzhen Zhuang et al. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1 (Jan. 2021), pp. 43–76. ISSN: 1558-2256. DOI: 10.1109/JPROC.2020.3004555.

[61] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark". In: *Neurocomputing* 503 (Sept. 7, 2022), pp. 92–108. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.06.111. URL: https://www.sciencedirect.com/science/article/pii/S0925231222008426 (visited on 05/05/2023).

[62] Geoffrey Hinton. *The Forward-Forward Algorithm: Some Preliminary Investigations*. Dec. 26, 2022. DOI: 10.48550/arXiv.2212.13345. arXiv: 2212.13345 [cs]. URL: http://arxiv.org/abs/2212.13345 (visited on 03/13/2023). preprint.

[63] Martin J. Willemink et al. "Preparing Medical Imaging Data for Machine Learning". In: *Radiology* 295.1 (Apr. 2020), pp. 4–15. ISSN: 0033-8419. DOI: 10.1148/radiol.2020192224. URL: https://pubs.rsna.org/doi/full/10.1148/radiol.2020192224 (visited on 05/05/2023).

[64] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (Dec. 1, 2015), pp. 211–252. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y (visited on 03/13/2023).

[65] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09* (2009), p. 8. DOI: 10.1109/CVPR.2009.5206848. URL: http://www.image-net.org/papers/imagenet_cvpr09.bib (visited on 10/03/2022).

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386 (visited on 10/03/2022).

[67] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385 [cs]. URL: http://arxiv.org/abs/1512.03385 (visited on 02/06/2023). preprint.

[68] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 1026–1034. URL: https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html (visited on 03/13/2023).

[69] Gordon Cooper. *Software Framework Requirements For Embedded Vision*. Semiconductor Engineering. Nov. 9, 2017. URL: https://semiengineering.com/software-framework-requirements-for-embedded-vision/ (visited on 03/10/2023).

[70] Vincent Dumoulin and Francesco Visin. *A Guide to Convolution Arithmetic for Deep Learning*. Jan. 11, 2018. DOI: 10.48550/arXiv.1603.07285. arXiv: 1603.07285 [cs, stat]. URL: http://arxiv.org/abs/1603.07285 (visited on 05/01/2023). preprint.

[71] Yann LeCun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html (visited on 10/04/2022).

[72] *Understanding Convolutional Neural Networks: A Complete Guide*. Jan. 18, 2023. URL: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/ (visited on 05/01/2023).

[73] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088 (6088 Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (visited on 10/06/2022).

[74] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1, 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[75] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. May 19, 2016. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473 [cs, stat]. URL: http://arxiv.org/abs/1409.0473 (visited on 01/17/2023). preprint.

[76] Guang Li et al. "Entangled Transformer for Image Captioning". In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, pp. 8928–8937. URL: https://openaccess.thecvf.com/content_ICCV_2019/html/Li_Entangled_Transformer_for_Image_Captioning_ICCV_2019_paper.html (visited on 05/05/2023).

[77] Yann LeCun. *RNNs, GRUs, LSTMs, Attention, Seq2Seq, and Memory Networks · Deep Learning*. 2020. URL: https://atcold.github.io/pytorch-Deep-Learning/en/week06/06-2/ (visited on 05/05/2023).

[78] Andrew Jaegle et al. *Perceiver: General Perception with Iterative Attention*. June 22, 2021. DOI: 10.48550/arXiv.2103.03206. arXiv: 2103.03206 [cs, eess]. URL: http://arxiv.org/abs/2103.03206 (visited on 03/16/2023). preprint.

[79] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805 [cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 03/16/2023). preprint.

[80] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. Oct. 29, 2019. DOI: 10.48550/arXiv.1910.13461. arXiv: 1910.13461 [cs, stat]. URL: http://arxiv.org/abs/1910.13461 (visited on 03/16/2023). preprint.

[81] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: *OpenAI* (2018), p. 12. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (visited on 03/16/2023).

[82] Alec Radford et al. "Language Models Are Unsupervised Multitask Learners". In: *OpenAI blog* 1.8 (2019), p. 9. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (visited on 03/16/2023).

[83] Tom Brown et al. "Language Models Are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 03/16/2023).

[84] Jie Lei et al. *Less Is More: ClipBERT for Video-and-Language Learning via Sparse Sampling*. Feb. 11, 2021. arXiv: 2102.06183 [cs]. URL: http://arxiv.org/abs/2102.06183 (visited on 01/03/2023). preprint.

[85] Liunian Harold Li et al. *VisualBERT: A Simple and Performant Baseline for Vision and Language*. Aug. 9, 2019. DOI: 10.48550/arXiv.1908.03557. arXiv: 1908.03557 [cs]. URL: http://arxiv.org/abs/1908.03557 (visited on 03/16/2023). preprint.

[86] Weijie Su et al. *VL-BERT: Pre-training of Generic Visual-Linguistic Representations*. Feb. 17, 2020. DOI: 10.48550/arXiv.1908.08530. arXiv: 1908.08530 [cs]. URL: http://arxiv.org/abs/1908.08530 (visited on 03/16/2023). preprint.

[87] *Confusion Matrix*. scikit-learn. URL: https://scikit-learn/stable/auto_examples/model_selection/plot_confusion_matrix.html (visited on 04/16/2023).

[88] Philip Gage. *A New Algorithm for Data Compression*. 1994. URL: http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM (visited on 05/27/2023).

[89] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2016. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162 (visited on 05/27/2023).

[90] Taku Kudo and John Richardson. *SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing*. Aug. 19, 2018. DOI: 10.48550/arXiv.1808.06226. arXiv: 1808.06226 [cs]. URL: http://arxiv.org/abs/1808.06226 (visited on 05/27/2023). preprint.

[91] Amit Datta, Michael Carl Tschantz, and Anupam Datta. *Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination*. Mar. 16, 2015. DOI: 10.48550/arXiv.1408.6491. arXiv: 1408.6491 [cs]. URL: http://arxiv.org/abs/1408.6491 (visited on 05/28/2023). preprint.

[92] Eirini Ntoutsi et al. "Bias in Data-Driven Artificial Intelligence Systems—An Introductory Survey". In: *WIREs Data Mining and Knowledge Discovery* 10.3 (2020), e1356. ISSN: 1942-4795. DOI: 10.1002/widm.1356. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1356 (visited on 05/28/2023).

[93] Christoph Molnar. *Interpretable Machine Learning*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book (visited on 11/29/2022).

[94] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". Aug. 9, 2016. arXiv: 1602.04938 [cs, stat]. URL: http://arxiv.org/abs/1602.04938 (visited on 02/02/2022).

[95] Scott Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". Nov. 24, 2017. arXiv: 1705.07874 [cs, stat]. URL: http://arxiv.org/abs/1705.07874 (visited on 02/02/2022).

[96] Lloyd S. Shapley. "A Value for N-Person Games". In: *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Ed. by Alvin E. Roth. Cambridge: Cambridge University Press, 1988, pp. 31–40. ISBN: 978-0-521-36177-4. DOI: 10.1017/CBO9780511528446.003. URL: https://www.cambridge.org/core/books/shapley-value/value-for-nperson-games/1AA9D343DE7A87A97F69E999D329B57A (visited on 05/27/2023).

[97] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization". In: *International Journal of Computer Vision* 128.2 (Feb. 2020), pp. 336–359. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7. arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391 (visited on 02/02/2022).

[98] Bolei Zhou et al. "Learning Deep Features for Discriminative Localization". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, pp. 2921–2929. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/Zhou_Learning_Deep_Features_CVPR_2016_paper.html (visited on 10/06/2022).

[99] Aishwarya Agrawal et al. *VQA: Visual Question Answering*. Oct. 26, 2016. arXiv: 1505.00468 [cs]. URL: http://arxiv.org/abs/1505.00468 (visited on 10/03/2022). preprint.

[100] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. Feb. 20, 2015. DOI: 10.48550/arXiv.1405.0312. arXiv: 1405.0312 [cs]. URL: http://arxiv.org/abs/1405.0312 (visited on 01/09/2023). preprint.

[101] Donald Geman et al. "Visual Turing Test for Computer Vision Systems". In: *Proceedings of the National Academy of Sciences* 112.12 (Mar. 24, 2015), pp. 3618–3623. DOI: 10.1073/pnas.1422953112. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1422953112 (visited on 05/28/2023).

[102] Wilson L. Taylor. ""Cloze Procedure": A New Tool for Measuring Readability". In: *Journalism Quarterly* 30.4 (Sept. 1, 1953), pp. 415–433. ISSN: 0022-5533. DOI: 10.1177/107769905303000401. URL: https://doi.org/10.1177/107769905303000401 (visited on 05/27/2023).

[103] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. Mar. 29, 2022. DOI: 10.48550/arXiv.2203.15556. arXiv: 2203.15556 [cs]. URL: http://arxiv.org/abs/2203.15556 (visited on 05/03/2023). preprint.

[104] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. Oct. 5, 2022. DOI: 10.48550/arXiv.2204.02311. arXiv: 2204.02311 [cs]. URL: http://arxiv.org/abs/2204.02311 (visited on 05/03/2023). preprint.

[105] Noam Shazeer. *GLU Variants Improve Transformer*. Feb. 12, 2020. DOI: 10.48550/arXiv.2002.05202. arXiv: 2002.05202 [cs, stat]. URL: http://arxiv.org/abs/2002.05202 (visited on 05/03/2023). preprint.

[106] *OpenAI API*. URL: https://platform.openai.com (visited on 04/07/2023).

[107] Yizhong Wang et al. *Self-Instruct: Aligning Language Model with Self Generated Instructions*. Dec. 20, 2022. DOI: 10.48550/arXiv.2212.10560. arXiv: 2212.10560 [cs]. URL: http://arxiv.org/abs/2212.10560 (visited on 04/07/2023). preprint.

[108] John Kirchenbauer et al. *A Watermark for Large Language Models*. Jan. 27, 2023. DOI: 10.48550/arXiv.2301.10226. arXiv: 2301.10226 [cs]. URL: http://arxiv.org/abs/2301.10226 (visited on 04/07/2023). preprint.

[109] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, pp. 580–587. URL: https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html (visited on 10/24/2022).

[110] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html (visited on 10/24/2022).

[111] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, pp. 779–788. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html (visited on 10/24/2022).

[112] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: Proceedings of the IEEE International Conference on Computer Vision. 2017, pp. 2980–2988. URL: https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html (visited on 10/24/2022).

[113] Oriol Vinyals et al. “Show and Tell: A Neural Image Caption Generator”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, pp. 3156–3164. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Vinyals_Show_and_Tell_2015_CVPR_paper.html (visited on 10/24/2022).

[114] Andrej Karpathy and Li Fei-Fei. “Deep Visual-Semantic Alignments for Generating Image Descriptions”. In: CVPR, June 2015, p. 17.

[115] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. “DenseCap: Fully Convolutional Localization Networks for Dense Captioning”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pp. 4565–4574. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.494. URL: http://ieeexplore.ieee.org/document/7780863/ (visited on 09/19/2022).

[116] Kenneth Tran et al. “Rich Image Captioning in the Wild”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016, pp. 49–56. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016_workshops/w12/html/Tran_Rich_Image_Captioning_CVPR_2016_paper.html (visited on 10/25/2022).

[117] Derek Koehler. “Explanation, Imagination, and Confidence in Judgment”. In: *Psychological bulletin* 110 (Dec. 1, 1991), pp. 499–519. DOI: 10.1037//0033-2909.110.3.499.

[118] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. “Explaining Collaborative Filtering Recommendations | Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work”. In: *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work* (Dec. 2, 2000), pp. 241–250. DOI: 10.1145/358916.358995. URL: https://dl.acm.org/doi/abs/10.1145/358916.358995 (visited on 10/26/2022).

[119] Mary T. Dzindolet et al. “The Role of Trust in Automation Reliance”. In: *International Journal of Human-Computer Studies*. Trust and Technology 58.6 (June 1, 2003), pp. 697–718. ISSN: 1071-5819. DOI: 10.1016/S1071-5819(03)00038-7. URL: https://www.sciencedirect.com/science/article/pii/S1071581903000387 (visited on 10/26/2022).

[120] Fei Jiang et al. “Artificial Intelligence in Healthcare: Past, Present and Future”. In: *Stroke and Vascular Neurology* 2.4 (Dec. 1, 2017). ISSN: 2059-8688, 2059-8696. DOI: 10.1136/svn-2017-000101. pmid: 29507784. URL: https://svn.bmj.com/content/2/4/230 (visited on 10/31/2022).

[121] Sana Tonekaboni et al. “What Clinicians Want: Contextualizing Explainable Machine Learning for Clinical End Use”. In: *Proceedings of the 4th Machine Learning for Healthcare Conference*. Machine Learning for Healthcare Conference. PMLR, Oct. 28, 2019, pp. 359–380. URL: https://proceedings.mlr.press/v106/tonekaboni19a.html (visited on 10/31/2022).

[122] Andreas Holzinger et al. “Causability and Explainability of Artificial Intelligence in Medicine”. In: *WIREs Data Mining and Knowledge Discovery* 9.4 (2019), e1312. ISSN: 1942-4795. DOI: 10.1002/widm.1312. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1312 (visited on 10/31/2022).

[123] Abhishek Gupta et al. “Deep Learning for Object Detection and Scene Perception in Self-Driving Cars: Survey, Challenges, and Open Issues”. In: *Array* 10 (July 1, 2021), p. 100057. ISSN: 2590-0056. DOI: 10.1016/j.array.2021.100057. URL: https://www.sciencedirect.com/science/article/pii/S2590005621000059 (visited on 10/31/2022).

[124] Erico Tjoa and Cuntai Guan. “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.11 (Nov. 2021), pp. 4793–4813. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2020.3027314.

[125] Seref Sagiroglu and Duygu Sinanc. “Big Data: A Review”. In: *2013 International Conference on Collaboration Technologies and Systems (CTS)*. 2013 International Conference on Collaboration Technologies and Systems (CTS). May 2013, pp. 42–47. DOI: 10.1109/CTS.2013.6567202.

[126] Rich Caruana et al. “Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’15. New York, NY, USA: Association for Computing Machinery, Aug. 10, 2015, pp. 1721–1730. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2788613. URL: https://doi.org/10.1145/2783258.2788613 (visited on 10/25/2022).

[127] Gregory F. Cooper et al. “Predicting Dire Outcomes of Patients with Community Acquired Pneumonia”. In: *Journal of Biomedical Informatics*. Clinical Machine Learning 38.5 (Oct. 1, 2005), pp. 347–366. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2005.02.005. URL: https://www.sciencedirect.com/science/article/pii/S1532046405000225 (visited on 10/25/2022).

[128] Gregory F. Cooper et al. “An Evaluation of Machine-Learning Methods for Predicting Pneumonia Mortality”. In: *Artificial Intelligence in Medicine* 9.2 (Feb. 1, 1997), pp. 107–138. ISSN: 0933-3657.

DOI: 10.1016/S0933-3657(96)00367-3. URL: https://www.sciencedirect.com/science/article/pii/S0933365796003673 (visited on 10/25/2022).

[129] Monica Bianchini and Franco Scarselli. "On the Complexity of Neural Network Classifiers: A Comparison Between Shallow and Deep Architectures". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.8 (Aug. 2014), pp. 1553–1565. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2013.2293637.

[130] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI". In: *Information Fusion* 58 (June 1, 2020), pp. 82–115. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.012. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103 (visited on 10/25/2022).

[131] Raul Vicente Garcia et al. "The Harms of Demographic Bias in Deep Face Recognition Research". In: *2019 International Conference on Biometrics (ICB)*. 2019 International Conference on Biometrics (ICB). June 2019, pp. 1–6. DOI: 10.1109/ICB45273.2019.8987334.

[132] Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. "Deep Blue". In: *Artificial Intelligence* 134.1 (Jan. 1, 2002), pp. 57–83. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(01)00129-1. URL: https://www.sciencedirect.com/science/article/pii/S0004370201001291 (visited on 11/08/2022).

[133] David Silver et al. "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play". In: *Science* 362.6419 (Dec. 7, 2018), pp. 1140–1144. DOI: 10.1126/science.aar6404. URL: https://www.science.org/doi/full/10.1126/science.aar6404 (visited on 10/07/2022).

[134] Lee R. Dice. "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26.3 (1945), pp. 297–302. ISSN: 0012-9658. DOI: 10.2307/1932409. JSTOR: 1932409. URL: https://www.jstor.org/stable/1932409 (visited on 04/03/2023).

[135] Thorvald Sørensen. "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species and Its Application to Analyses of the Vegetation on Danish Commons". In: *Kongelige Danske Videnskabernes Selskab* 5.4 (1948), pp. 1–34. URL: https://www.royalacademy.dk/Publications/High/295_S%C3%B8rensen,%20Thorvald.pdf (visited on 04/04/2023).

[136] Jiasen Lu et al. "Neural Baby Talk". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 7219–7228. URL: https://openaccess.thecvf.com/content_cvpr_

2018/html/Lu_Neural_Baby_Talk_CVPR_2018_paper.html (visited on 09/19/2022).

[137] Yash Goyal et al. *Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering*. May 15, 2017. arXiv: 1612.00837 [cs]. URL: http://arxiv.org/abs/1612.00837 (visited on 10/03/2022). preprint.

[138] *Perona Lab - CUB-200-2011*. URL: https://www.vision.caltech.edu/datasets/cub_200_2011/ (visited on 04/17/2023).

[139] *Caffe Installation*. URL: http://caffe.berkeleyvision.org/installation.html#prerequisites (visited on 04/17/2023).

[140] *Caffe Deep Learning Framework and NVIDIA GPU Acceleration*. NVIDIA. URL: https://www.nvidia.com/en-au/data-center/gpu-accelerated-applications/caffe/ (visited on 04/17/2023).

[141] *ML nodes - Universitetet i Oslo*. URL: https://www.uio.no/tjenester/it/forskning/kompetansehuber/uio-ai-hub-node-project/it-resources/ml-nodes/index.html (visited on 04/17/2023).

[142] *Podman*. URL: https://podman.io/ (visited on 04/17/2023).

[143] *Migrate to TensorFlow 2 | TensorFlow Core*. TensorFlow. URL: https://www.tensorflow.org/guide/migrate (visited on 04/17/2023).

[144] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. Oct. 16, 2021. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs]. URL: http://arxiv.org/abs/2106.09685 (visited on 03/28/2023). preprint.

[145] Michael McCloskey and Neal J. Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: *Psychology of Learning and Motivation*. Ed. by Gordon H. Bower. Vol. 24. Academic Press, Jan. 1, 1989, pp. 109–165. DOI: 10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368 (visited on 05/10/2023).

[146] Artem Andreenko [@miolini]. *I've Sucefully Runned LLaMA 7B Model on My 4GB RAM Raspberry Pi 4. It's Super Slow about 10sec/Token. But It Looks We Can Run Powerful Cognitive Pipelines on a Cheap Hardware.* Twitter. Mar. 12, 2023. URL: https://twitter.com/miolini/status/1634982361757790209 (visited on 04/08/2023).

[147] Eric J. Wang. *Alpaca-LoRA*. Apr. 8, 2023. URL: https://github.com/tloen/alpaca-lora (visited on 04/08/2023).

[148] Hanna Borgli et al. "HyperKvasir, a Comprehensive Multi-Class Image and Video Dataset for Gastrointestinal Endoscopy". In: *Scientific Data* 7.1 (1 Aug. 28, 2020), p. 283. ISSN: 2052-4463. DOI: 10.1038/s41597-020-00622-y. URL: https://www.nature.com/articles/s41597-020-00622-y (visited on 03/17/2023).

[149] *Vgg16 — Torchvision Main Documentation*. URL: https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html (visited on 05/11/2023).

[150] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. Dec. 25, 2016. DOI: 10.48550/arXiv.1612.08242. arXiv: 1612.08242 [cs]. URL: http://arxiv.org/abs/1612.08242 (visited on 04/09/2023). preprint.

[151] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. Apr. 8, 2018. DOI: 10.48550/arXiv.1804.02767. arXiv: 1804.02767 [cs]. URL: http://arxiv.org/abs/1804.02767 (visited on 04/09/2023). preprint.

[152] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. Apr. 22, 2020. DOI: 10.48550/arXiv.2004.10934. arXiv: 2004.10934 [cs, eess]. URL: http://arxiv.org/abs/2004.10934 (visited on 04/09/2023). preprint.

[153] Glenn Jocher. *Yolov5*. URL: https://github.com/ultralytics/yolov5 (visited on 04/09/2023).

[154] Chuyi Li et al. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. Sept. 7, 2022. DOI: 10.48550/arXiv.2209.02976. arXiv: 2209.02976 [cs]. URL: http://arxiv.org/abs/2209.02976 (visited on 04/09/2023). preprint.

[155] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors*. July 6, 2022. DOI: 10.48550/arXiv.2207.02696. arXiv: 2207.02696 [cs]. URL: http://arxiv.org/abs/2207.02696 (visited on 04/09/2023). preprint.

[156] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics (visited on 04/09/2023).

[157] Zhengyuan Yang et al. *MM-REACT: Prompting ChatGPT for Multimodal Reasoning and Action*. Mar. 20, 2023. DOI: 10.48550/arXiv.2303.11381. arXiv: 2303.11381 [cs]. URL: http://arxiv.org/abs/2303.11381 (visited on 03/28/2023). preprint.

[158] Xueyan Zou et al. *Generalized Decoding for Pixel, Image, and Language*. Dec. 21, 2022. DOI: 10.48550/arXiv.2212.11270. arXiv: 2212.11270 [cs]. URL: http://arxiv.org/abs/2212.11270 (visited on 04/10/2023). preprint.

[159] *Hugging Face – The AI Community Building the Future.* URL: https://huggingface.co/ (visited on 05/11/2023).

[160] *LLaMA_tokenizer*. URL: https://huggingface.co/docs/transformers/main/model_doc/llama (visited on 04/10/2023).

[161] *Sentencepiece*. GitHub. URL: https://github.com/google/sentencepiece (visited on 04/10/2023).

[162] Hila Chefer, Shir Gur, and Lior Wolf. "Generic Attention-Model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers". In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021, pp. 397–406. URL: https://openaccess.thecvf.com/content/ICCV2021/html/Chefer_Generic_Attention-Model_Explainability_for_Interpreting_Bi-Modal_and_Encoder-Decoder_Transformers_ICCV_2021_paper.html (visited on 05/12/2023).

[163] Hila Chefer, Shir Gur, and Lior Wolf. "Transformer Interpretability Beyond Attention Visualization". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 782–791. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Chefer_Transformer_Interpretability_Beyond_Attention_Visualization_CVPR_2021_paper.html (visited on 05/12/2023).

[164] Oren Barkan et al. "Grad-SAM: Explaining Transformers via Gradient Self-Attention Maps". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. New York, NY, USA: Association for Computing Machinery, Oct. 30, 2021, pp. 2882–2887. ISBN: 978-1-4503-8446-9. DOI: 10.1145/3459637.3482126. URL: https://dl.acm.org/doi/10.1145/3459637.3482126 (visited on 05/12/2023).

[165] Moritz Böhle, Mario Fritz, and Bernt Schiele. *Holistically Explainable Vision Transformers*. Jan. 20, 2023. DOI: 10.48550/arXiv.2301.08669. arXiv: 2301.08669 [cs, stat]. URL: http://arxiv.org/abs/2301.08669 (visited on 05/12/2023). preprint.

[166] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. *Learning to Generate Reviews and Discovering Sentiment*. Apr. 6, 2017. DOI: 10.48550/arXiv.1704.01444. arXiv: 1704.01444 [cs]. URL: http://arxiv.org/abs/1704.01444 (visited on 05/12/2023). preprint.

[167] Sofia Serrano and Noah A. Smith. *Is Attention Interpretable?* June 9, 2019. DOI: 10.48550/arXiv.1906.03731. arXiv: 1906.03731 [cs]. URL: http://arxiv.org/abs/1906.03731 (visited on 05/12/2023). preprint.

[168] Sarthak Jain and Byron C. Wallace. *Attention Is Not Explanation*. May 8, 2019. DOI: 10.48550/arXiv.1902.10186. arXiv: 1902.10186 [cs]. URL: http://arxiv.org/abs/1902.10186 (visited on 05/12/2023). preprint.

[169] Samira Abnar and Willem Zuidema. *Quantifying Attention Flow in Transformers*. May 31, 2020. DOI: 10.48550/arXiv.2005.00928. arXiv: 2005.00928 [cs]. URL: http://arxiv.org/abs/2005.00928 (visited on 05/12/2023). preprint.

[170] Steven Hicks et al. *ImageCLEFmed-MEDVQA-GI-2023*. Simula, May 8, 2023. URL: https://github.com/simula/ImageCLEFmed-MEDVQA-GI-2023 (visited on 05/12/2023).

[171] Steven Hicks et al. *ImageCLEFmed MEDVQA-GI - ImageCLEF / LifeCLEF - Multimedia Retrieval in CLEF*. URL: https://www.imageclef.org/2023/medical/vqa (visited on 05/12/2023).

[172] *Sequence/Context Length of This Model? · Issue #16 · Facebookresearch/Llama*. URL: https://github.com/facebookresearch/llama/issues/16 (visited on 04/11/2023).

[173] Victor Zhong, Caiming Xiong, and Richard Socher. *Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning*. Nov. 9, 2017. DOI: 10.48550/arXiv.1709.00103. arXiv: 1709.00103 [cs]. URL: http://arxiv.org/abs/1709.00103 (visited on 05/14/2023). preprint.

[174] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: vol. 1. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. URL: http://aclweb.org/anthology/N18-1101 (visited on 05/14/2023).

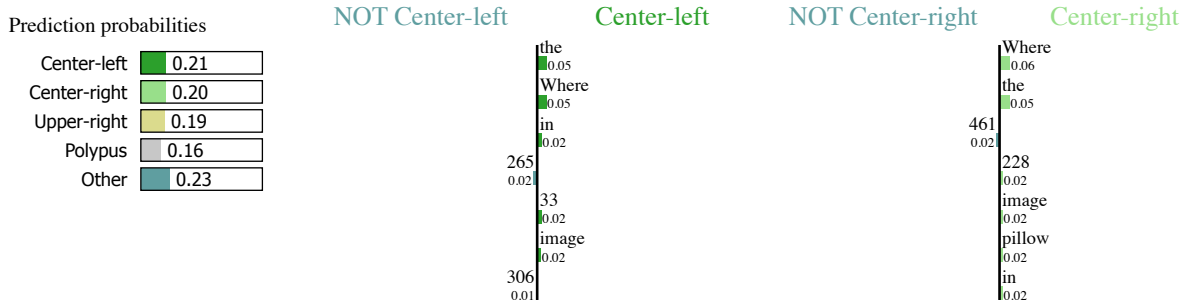
[175] *Sklearn.Linear_model.SGDClassifier — Scikit-Learn 1.2.2 Documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html (visited on 05/16/2023).

[176] *Lime Package — Lime 0.1 Documentation*. URL: https://lime-ml.readthedocs.io/en/latest/lime.html#module-lime.lime_text (visited on 05/16/2023).

Appendix A

Additional Proxy Model Explanations

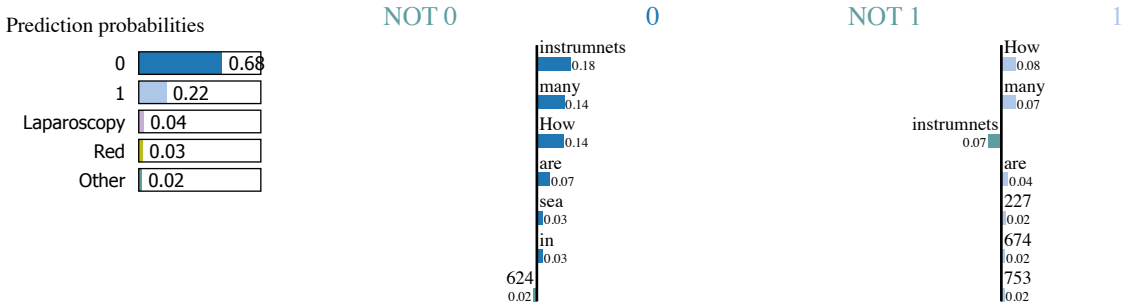
This chapter adds some additional examples of the proxy model explained by LIME. The trend of these examples is discussed in subsection 4.5.3.



Text with highlighted words

instruction: Where in the image is the abnormality?, input: [['ocarina', 18.294], ["jack-o'-lantern", 17.719], ['piggy bank', 8.768], ['mask', 4.141], ['pretzel', 2.228], ['stingray', 1.781], ['bagel', 1.596], ['potpie', 1.545], ['saltshaker', 1.543], ['Band Aid', 1.261], ['chiton', 1.214], ['pick', 1.186], ['chocolate sauce', 0.949], ['bakery', 0.942], ['electric ray', 0.908], ['triceratops', 0.877], ['pitcher', 0.875], ['bathing cap', 0.805], ['teapot', 0.756], ['spatula', 0.683], ['sunscreen', 0.672], ['toilet seat', 0.636], ['plunger', 0.61], ['dough', 0.577], ['mushroom', 0.573], ['crash helmet', 0.531], ['bathtub', 0.516], ['maraca', 0.461], ['shield', 0.444], ['bolete', 0.432], ['vase', 0.431], ['chambered nautilus', 0.425], ['jigsaw puzzle', 0.401], ['agaric', 0.383], ['football helmet', 0.378], ['sea slug', 0.376], ['cowboy hat', 0.37], ['electric guitar', 0.368], ['spindle', 0.364], ['baseball', 0.343], ['buckle', 0.33], ['goblet', 0.311], ['clog', 0.306], ['gyromitra', 0.3], ['ice cream', 0.299], ['padlock', 0.296], ['sea cucumber', 0.285], ['soap dispenser', 0.279], ['puck', 0.274], ['pomegranate', 0.273], ['breastplate', 0.267], ['tub', 0.266], ['slug', 0.265], ['anemone fish', 0.265], ['spotted salamander', 0.236], ['cuirass', 0.23], ['water jug', 0.228], ['bell pepper', 0.219], ['wooden spoon', 0.218], ['whiskey jug', 0.217], ['French loaf', 0.217], ['nematode', 0.215], ['nipple', 0.212], ['soup bowl', 0.212], ['coffee mug', 0.21], ['tray', 0.207], ['candle', 0.206], ["potter's wheel", 0.202], ['sea anemone', 0.201], ['butternut squash', 0.196], ['flatworm', 0.194], ['swimming trunks', 0.194], ['bib', 0.183], ['mixing bowl', 0.181], ['trilobite', 0.18], ['ice lolly', 0.176], ['teddy', 0.168], ['ladle', 0.162], ['pillow', 0.161], ['safety pin', 0.154], ['acorn', 0.152], ['cheeseburger', 0.15], ['conch', 0.147], ['pot', 0.147], ['wig', 0.145], ['dugong', 0.142], ['rain barrel', 0.135], ['isopod', 0.134], ['shower cap', 0.133], ['flamingo', 0.13], ['lotion', 0.129], ['eft', 0.124], ['golf ball', 0.124], ['sweatshirt', 0.123], ['Petri dish', 0.122], ['bubble', 0.12], ['sandal', 0.119], ['ballplayer', 0.115], ['jersey', 0.113], ['acorn squash', 0.113]]

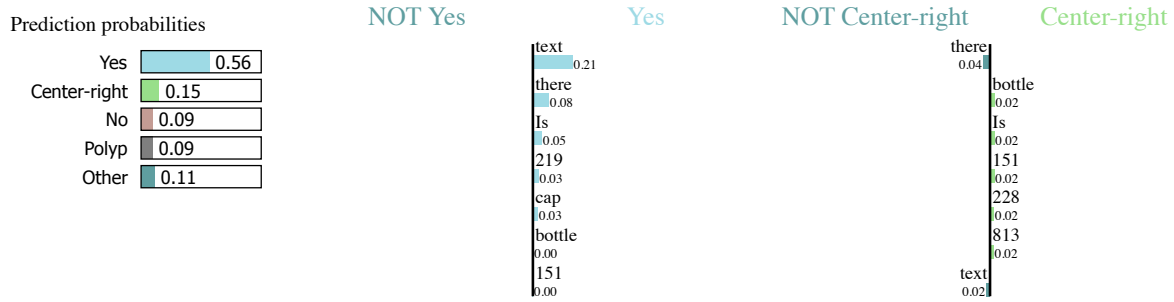
Figure A.1: The proxy model explained by LIME. The correct answer is *Upper-right*, Alpaca-VQA predicted *Center-left*, and this is the explanation of the prediction.



Text with highlighted words

instruction: How many instrumnets are in the image?, input: [['bathtub', 16.132], ['tub', 11.73], ['ice lolly', 6.817], ['dough', 6.386], ['ice cream', 4.68], ['bathing cap', 4.605], ['sunscreen', 4.135], ['frying pan', 3.87], ['spatula', 2.753], ['shower cap', 2.378], ['plunger', 2.103], ['fig', 1.956], ['electric ray', 1.811], ['tick', 1.733], ['wooden spoon', 1.686], ['chiton', 1.588], ['sea cucumber', 1.51], ['stingray', 1.449], ['Crock Pot', 1.261], ['jellyfish', 1.139], ['mixing bowl', 1.122], ['chocolate sauce', 0.916], ['conch', 0.781], ["potter's wheel", 0.674], ['hog', 0.624], ['bubble', 0.624], ['lipstick', 0.598], ['wok', 0.52], ['lotion', 0.496], ['washbasin', 0.461], ['strawberry', 0.452], ['Dungeness crab', 0.436], ['bucket', 0.421], ['face powder', 0.42], ['bagel', 0.365], ['chambered nautilus', 0.343], ['cleaver', 0.342], ['piggy bank', 0.311], ['isopod', 0.309], ['bakery', 0.309], ['axolotl', 0.273], ['puck', 0.264], ['pomegranate', 0.229], ['starfish', 0.227], ['plate', 0.227], ['Dutch oven', 0.226], ['slug', 0.212], ['bell pepper', 0.205], ['hamster', 0.198], ['mashed potato', 0.171], ['brain coral', 0.17], ['soup bowl', 0.17], ['consomme', 0.17], ['nematode', 0.166], ['sea anemone', 0.151], ['ladle', 0.15], ['hermit crab', 0.145], ['meat loaf', 0.14], ['nipple', 0.137], ['Band Aid', 0.135], ['wig', 0.129], ['snail', 0.125], ['swimming trunks', 0.119], ['orange', 0.111], ['caldron', 0.108], ['flatworm', 0.105], ['safety pin', 0.097], ['cheeseburger', 0.097], ['butcher shop', 0.095], ['sea urchin', 0.094], ['plastic bag', 0.091], ['sandbar', 0.091], ['mortar', 0.09], ['paintbrush', 0.088], ['bottlecap', 0.086], ['hotdog', 0.084], ['bikini', 0.081], ['pick', 0.081], ['candle', 0.08], ['thunder snake', 0.076], ['brassiere', 0.071], ['Petri dish', 0.068], ['Granny Smith', 0.063], ['butternut squash', 0.061], ['hip', 0.061], ['cucumber', 0.057], ['paper towel', 0.056], ['goldfish', 0.055], ['snorkel', 0.055], ['eggnog', 0.054], ['cockroach', 0.053], ['pretzel', 0.053], ['trilobite', 0.052], ['syringe', 0.051], ['hot pot', 0.051], ['crayfish', 0.05], ['nail', 0.049], ['sea slug', 0.046], ['dugong', 0.046], ['gyromitra', 0.046]]

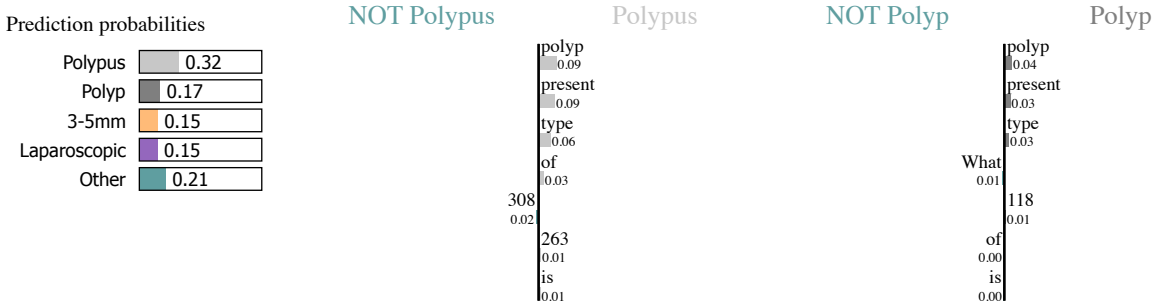
Figure A.2: The proxy model explained by LIME. The correct answer is *0*, Alpaca-VQA predicted *0*, and this is the explanation of the prediction.



Text with highlighted words

instruction: Is there text?, input: [['sunscreen', 20.371], ['swimming trunks', 9.902], ['bathing cap', 9.789], ['Band Aid', 4.735], ['lotion', 3.591], ['dugong', 3.104], ['pop bottle', 2.345], ['ice lolly', 1.656], ['book jacket', 1.465], ['snorkel', 1.369], ['iPod', 1.338], ['hog', 1.184], ['plunger', 0.954], ['pick', 0.922], ['sea cucumber', 0.897], ['sunglass', 0.859], ['bell pepper', 0.853], ['hair spray', 0.83], ['punching bag', 0.813], ['lighter', 0.808], ['great white shark', 0.793], ['bathtub', 0.723], ['tub', 0.711], ['conch', 0.708], ['comic book', 0.679], ['beer glass', 0.561], ['packet', 0.537], ['cellular telephone', 0.534], ['water bottle', 0.529], ['harmonica', 0.476], ['butcher shop', 0.455], ['ocarina', 0.451], ['pill bottle', 0.442], ['dough', 0.441], ['cash machine', 0.424], ['beer bottle', 0.412], ['oil filter', 0.398], ['sunglasses', 0.376], ['bikini', 0.355], ['hotdog', 0.347], ['jellyfish', 0.342], ['chiton', 0.318], ['Granny Smith', 0.312], ['pomegranate', 0.306], ['whistle', 0.303], ['barbershop', 0.29], ['scuba diver', 0.286], ['stingray', 0.284], ['pretzel', 0.283], ['bagel', 0.28], ['ice cream', 0.273], ['scale', 0.264], ['axolotl', 0.246], ['Crock Pot', 0.235], ['wig', 0.235], ['remote control', 0.232], ['piggy bank', 0.229], ['lipstick', 0.228], ['shower cap', 0.228], ['bubble', 0.227], ['vending machine', 0.226], ['syringe', 0.221], ['tiger shark', 0.219], ['butternut squash', 0.219], ['neck brace', 0.211], ['beaker', 0.195], ['microphone', 0.194], ['gyromitra', 0.184], ['chocolate sauce', 0.179], ['football helmet', 0.178], ["jack-o'-lantern", 0.176], ['red wine', 0.176], ['can opener', 0.171], ['oxygen mask', 0.157], ['cheeseburger', 0.154], ['scorpion', 0.151], ['cleaver', 0.151], ['coffee mug', 0.151], ['mask', 0.15], ['flamingo', 0.147], ['stopwatch', 0.147], ['fig', 0.142], ['television', 0.141], ['maillot', 0.137], ['tick', 0.136], ['perfume', 0.13], ['laptop', 0.127], ['maraca', 0.124], ['soap dispenser', 0.124], ['wine bottle', 0.123], ['wok', 0.123], ['isopod', 0.12], ['gas pump', 0.118], ['American lobster', 0.114], ['electric ray', 0.111], ['hamster', 0.11], ["potter's wheel", 0.11], ['stage', 0.11], ['bottlecap', 0.108], ['cup', 0.105]]

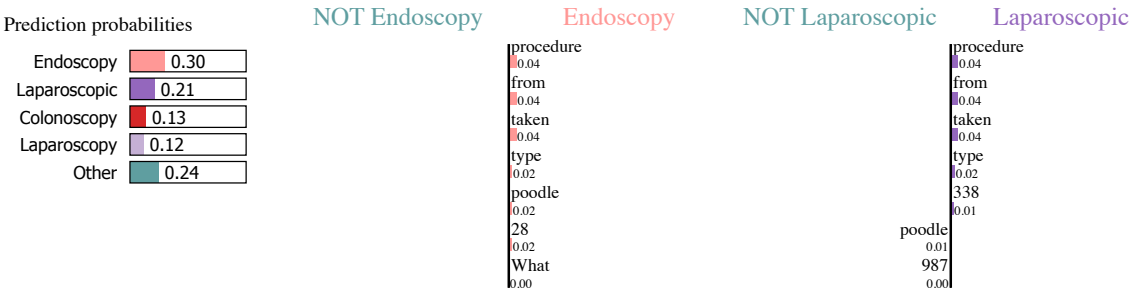
Figure A.3: The proxy model explained by LIME. The correct answer is *Yes*, Alpaca-VQA predicted *Yes*, and this is the explanation of the prediction.



Text with highlighted words

instruction: What type of polyp is present?, input: [['pretzel', 29.072], ['hotdog', 10.585], ['bagel', 5.857], ['French loaf', 4.457], ['chiton', 3.11], ['chocolate sauce', 2.724], ['cheeseburger', 1.858], ['wok', 1.721], ['dough', 1.603], ['buckeye', 1.589], ["jack-o'-lantern", 1.443], ['butternut squash', 1.319], ['mushroom', 1.25], ['bell pepper', 1.184], ['wooden spoon', 1.176], ['bolete', 1.09], ['ice lolly', 0.997], ['frying pan', 0.943], ['caldron', 0.94], ['ladle', 0.826], ['consomme', 0.725], ['hot pot', 0.679], ['soup bowl', 0.661], ['American lobster', 0.635], ['Dutch oven', 0.634], ['sea cucumber', 0.595], ['acorn squash', 0.588], ['potpie', 0.542], ['gyromitra', 0.507], ['bakery', 0.499], ['nematode', 0.49], ['Crock Pot', 0.485], ['plunger', 0.478], ['burrito', 0.468], ['mashed potato', 0.448], ['fig', 0.433], ['cucumber', 0.42], ['mortar', 0.414], ['beer glass', 0.345], ['maraca', 0.343], ['ice cream', 0.32], ['mixing bowl', 0.317], ['slug', 0.308], ['orange', 0.308], ['meat loaf', 0.303], ['agaric', 0.295], ['earthstar', 0.294], ['espresso', 0.275], ['Granny Smith', 0.269], ['rotisserie', 0.268], ['pitcher', 0.265], ['spatula', 0.263], ['ocarina', 0.261], ['candle', 0.257], ['tub', 0.251], ['pomegranate', 0.217], ['bathtub', 0.213], ['flatworm', 0.21], ['spotted salamander', 0.174], ['tray', 0.169], ['acorn', 0.156], ['conch', 0.154], ['lemon', 0.152], ['stinkhorn', 0.152], ['plate', 0.151], ["potter's wheel", 0.145], ['whiskey jug', 0.141], ['piggy bank', 0.14], ['goblet', 0.136], ['Dungeness crab', 0.133], ['packet', 0.128], ['spaghetti squash', 0.126], ['bucket', 0.124], ['cup', 0.123], ['tench', 0.121], ['steel drum', 0.118], ['guacamole', 0.118], ['banana', 0.114], ['Band Aid', 0.108], ['gong', 0.105], ['eft', 0.104], ['isopod', 0.102], ['coffee mug', 0.101], ['bubble', 0.098], ['nipple', 0.092], ['pot', 0.092], ['hen-of-the-woods', 0.09], ['teapot', 0.086], ['stingray', 0.084], ['electric ray', 0.083], ['chain', 0.079], ['hog', 0.078], ['butcher shop', 0.076], ['grocery store', 0.074], ['corn', 0.074], ['eel', 0.073], ['beer bottle', 0.073], ['common newt', 0.072], ['rock crab', 0.071], ['crayfish', 0.07]]

Figure A.4: The proxy model explained by LIME. The correct answer is *Paris iia*, Alpaca-VQA predicted *Polypus*, and this is the explanation of the prediction.



Text with highlighted words

instruction: What type of procedure is the image taken from?, input: [['hog', 12.643], ['dough', 5.328], ['flamingo', 4.063], ['dugong', 3.987], ["potter's wheel", 2.317], ['Mexican hairless', 2.312], ['cauliflower', 2.166], ['pomegranate', 1.714], ['ice cream', 1.678], ['Band Aid', 1.671], ['axolotl', 1.637], ['wig', 1.582], ['hippopotamus', 1.531], ['sunscreen', 1.204], ['butcher shop', 1.157], ['tub', 0.993], ['Granny Smith', 0.97], ['teddy', 0.922], ['mask', 0.895], ['swimming trunks', 0.846], ['brain coral', 0.837], ['Crock Pot', 0.777], ['eel', 0.773], ['wombat', 0.712], ['shower cap', 0.697], ['bathtub', 0.681], ['tick', 0.667], ['triceratops', 0.653], ['bagel', 0.633], ['hamster', 0.584], ['wool', 0.544], ['bell pepper', 0.526], ['nematode', 0.524], ['thunder snake', 0.499], ['butternut squash', 0.491], ['Chesapeake Bay retriever', 0.487], ['piggy bank', 0.486], ['paper towel', 0.484], ['scorpion', 0.457], ['miniature poodle', 0.454], ['ocarina', 0.452], ['caldron', 0.446], ['microphone', 0.401], ['tusker', 0.4], ['potpie', 0.385], ['curly-coated retriever', 0.382], ['consomme', 0.379], ['candle', 0.378], ['toy poodle', 0.37], ["jack-o'-lantern", 0.366], ['sea cucumber', 0.358], ['chiton', 0.339], ['wooden spoon', 0.338], ['lotion', 0.335], ['plunger', 0.333], ['harmonica', 0.311], ['Indian elephant', 0.302], ['orange', 0.301], ['cleaver', 0.299], ['strawberry', 0.298], ['face powder', 0.297], ['head cabbage', 0.293], ['Weimaraner', 0.29], ['platypus', 0.288], ['fig', 0.285], ['Gila monster', 0.281], ['electric ray', 0.28], ['African elephant', 0.28], ['sombrero', 0.279], ['gong', 0.277], ['stingray', 0.276], ['ski mask', 0.276], ['bucket', 0.259], ['bubble', 0.259], ['bathing cap', 0.257], ['stinkhorn', 0.255], ['sleeping bag', 0.25], ['pick', 0.236], ['sunglass', 0.235], ['toilet tissue', 0.233], ['conch', 0.226], ['red wine', 0.222], ['wok', 0.22], ['standard poodle', 0.219], ['burrito', 0.215], ['ice lolly', 0.213], ['chocolate sauce', 0.213], ['neck brace', 0.211], ['stethoscope', 0.208], ['bath towel', 0.207], ['starfish', 0.206], ['scale', 0.204], ['Dutch oven', 0.202], ['mortar', 0.2], ['mongoose', 0.197], ['isopod', 0.194], ['buckle', 0.192], ['mashed potato', 0.191], ['bolete', 0.185], ['crayfish', 0.183]]

Figure A.5: The proxy model explained by LIME. The correct answer is *Colonoscopy*, Alpaca-VQA predicted *Endoscopy*, and this is the explanation of the prediction.