FernUniversität in Hagen
Faculty of Mathematics and Computer Science

Artificial Intelligence Group

Utilizing Knowledge Graphs in Outside Knowledge-based Visual Question Answering

Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in Practical Computer Science

submitted by
Max Upravitelev

First examiner:	Prof. Dr. Matthias Thimm Artificial Intelligence Group
Advisor:	Isabelle Kuhlmann Artificial Intelligence Group
External advisor:	Dr. Christopher Krauß Fraunhofer FOKUS

Statement

Ich erkläre, dass ich die Masterarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the webpage of the artificial intelligence group.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

Berlin 19/08/2023 .....

(Place, Date) (Signature)

Zusammenfassung

Das Feld des Visual Question Answering (VQA) verbindet die Disziplinen des bild- und sprachbasierten Urteilens, indem es das Verstehen einer Szene und die Beantwortung beliebiger Fragen in Bezug auf ein bestimmtes Bild kombiniert. Die Anzahl der Fragen, die beantwortet werden können, ist durch die in einem Bild enthaltenen visuellen Informationen begrenzt. Sie kann jedoch durch die Nutzung von externem Wissen aus verschiedenen Quellen erweitert werden, zum Beispiel durch die Verwendung von Wissensgraphen (KGs), die auf öffentlich zugänglichen Wissensdatenbanken basieren. Vor kurzem wurde die Aufgabenstellung des Outside Knowledge Visual Question Answering (OK-VQA) vorgestellt, um die Forschung auf diesem Gebiet zu befördern. Mehrere aktuelle Lösungen verwenden Graph Neural Networks (GNNs) für diese Aufgabe. Eine der Herausforderungen bei der weiteren Verbesserung der Ergebnisse bei dieser Strategie ist die Interpretierbarkeit dieser Modelle. Ziel dieser Arbeit ist es zu untersuchen, ob ein aus aktuellen Forschungsansätzen ausgewähltes Basismodell gleichzeitig sowohl in der Interpretierbarkeit als auch in der Genauigkeit verbessert werden kann. Um dies zu erreichen, werden mehrere Aktualisierungen der Modellarchitektur vorgeschlagen, die sich auf einem attention-basierten Ansatz und auf Konstruktionsstrategien der Knotenmerkmale konzentrieren, die den verwendeten KG repräsentieren. Die Evaluation der vorgeschlagenen Aktualisierungen weist darauf hin, dass Interpretierbarkeit und Genauigkeit gleichzeitig verbessert werden können, wenn auch nicht in jedem Szenario. Dieses Verhalten wird in drei Fallstudien weiter betrachtet, in denen anhand ausgewählter Beispiele untersucht wird, wie attention scores zur Interpretation von Vorhersagen eines GNN-basierten Modells genutzt werden können.

Abstract

The field of Visual Question Answering (VQA) bridges the disciplines of vision- and language-based reasoning by combining scene understanding and the answering of arbitrary questions in regard to a given image. The number of questions that can be answered is limited by the visual information given in an image. However, it can be expanded by utilizing external knowledge from different sources, for example, by utilizing Knowledge Graphs (KGs) based on publicly available Knowledge Bases. Recently, the Outside Knowledge Visual Question Answering (OK-VQA) task was introduced to facilitate research in this field. Several current state-of-the-art solutions incorporate Graph Neural Networks (GNNs) for this task. One of the challenges in further improving the results of this strategy is the interpretability of these models. The goal of this theses is to investigate whether a baseline model chosen from current research approaches can be improved simultaneously in interpretability as well as in accuracy. To achieve this, several updates to the model architecture are proposed, focusing on an attention-based approach and on construction

strategies of the node features that represent the utilized KG. The evaluation of the proposed updates indicates that interpretability and accuracy can be improved simultaneously, although not in all scenarios. This behavior is further investigated in three case studies that consider selected examples to examine how attention scores can be utilized to interpret predictions from a GNN-based model.

Contents

List of Figures

1 Introduction

1.1 Motivation

In his first lecture in an introductory series on philosophical terminology [Ado73], the philosopher Theodor W. Adorno described two different approaches to the way humans are able to grasp the meaning of a concept. One approach is to define it with words: For instance, by capturing the meaning of a natural language sign related to a concept through an assessment of how this sign is generally used. In philosophy as a science, this strategy would be equivalent to capturing the meaning of a word by considering other words and the relations between them. To contrast word-based approaches, Adorno highlights another strategy outside of philosophy on how a concept can be grasped without the reliance of words: by showing things or, simply put, by pointing at things. This approach is not only independent of words, it is also more suitable in some cases, like pointing to shades of colors.

There are two Machine Learning (ML) fields that can be related to exactly these two strategies described above: Natural Language Processing (NLP) and Computer Vision (CV). Neural Network (NN)-based approaches within NLP can be summarized as a strategy to grasp a concept by analyzing clusters of similar impressions and embedding them in the vector space so that they can be related to others [Ots22]. At the same time, NN-based CV models are trained by being pointed to image data that represents these concepts, from which vector representations corresponding to specific concepts are learned.

NN-based approaches in both fields often learn from data that was created by humans for humans in the first place. Textual datasets are usually made from existing text corpora, and image datasets are often composed of human-readable image data. Therefore, there seems to be a degree of accordance between the research on NN-based reasoning processes and the way humans gain knowledge. As described above, humans can rely on different modalities that can be based on, for example, textual or visual data. Hence, it stands to reason that considering both modalities within one unified field can increase the ability of machine understanding regarding concepts in general, mimicking the human approach. One field devoted to this task is Visual Question Answering (VQA), the field of reasoning based on vision and language.

VQA aims at analyzing a scene within an image so that arbitrary questions about this scene can be answered. Its approaches often combine strategies from subfields like CV, NLP, and Knowledge Representation and Reasoning (KRR) in general. VQA can be understood as a generalization of Image Captioning (IC), which can assist blind and visually impaired people in interpreting the content of visual images by generating textual descriptions of images. While image captions can return an answer to the question “What is in the image?”, VQA provides the generalized ability to query an image for different types of questions.

The number of questions that can be answered is limited by the visual information provided in an image. It can, however, be expanded by utilizing external

knowledge. One common source for this knowledge is implicit knowledge from pre-trained language models, while another approach of external knowledge acquisition considers publicly available data, like large Knowledge Bases (KBs) for structured textual, visual and commonsense knowledge (like ConceptNet, DBPedia or VisualGenome), or unstructured sources like Wikipedia or Google Images.

Dealing with outside knowledge in VQA is a rather novel task. Recently, a first dataset (OK-VQA, [MRFM19]) was proposed to provide an evaluation benchmark for approaches to solve this specific task. Currently, the task can still be considered unsolved, since state-of-the-art approaches reach around 60 % in overall accuracy on this benchmark [1].

Most current approaches within the OK-VQA task focus on architectures based on NNs. NN-based models, such as the transformer model [VSP+17], can encode patterns correlated to meaning and return quantified values of concepts and their relations, but these feature representations and the resulting predictions are not easily interpretable for humans. This poses several challenges when utilizing these models in real-life scenarios. For example, there is the issue of compliance with legal standards, which can only be ensured if information can be given on the reasoning process behind a prediction. In addition to societal issues, there are also challenges from a technical point of view: Interpretability is needed to improve a model by error analysis, since understanding the source of wrong predictions could showcase the kinds of problem a model struggles with the most.

1.2 Problem Statement

One common approach to solving the OK-VQA task within current solutions is the utilization of end-to-end trained models based on Graph Neural Networks (GNNs). A challenge to further improve the results of this approach is the interpretability of these models. As with most NN-based strategies, GNNs often behave as black boxes, where the actual reasoning behind an answer prediction is difficult to interpret. Interpretability is, however, a desirable property, as it could highlight which particular cases the models have the most difficulty with. Nevertheless, an increase in interpretability has to take into account a possible trade-off between interpretability and accuracy, where increasing one factor could lead to a decline in the other.

1.3 Research Objectives and Strategy

The goal of VQA, in general, is the ability to query arbitrary real-world images with arbitrary questions. A step forward in this direction would be an improvement of the task proposed by the OK-VQA benchmark. Furthermore, this thesis is based on the assumption that improved interpretability can increase the results of future work by showcasing the reasons for wrong predictions. Therefore, the central research question of this thesis is:

[1] Source: https://okvqa.allenai.org/leaderboard.html, accessed on 19.08.2023

Is it viable to utilize Knowledge Graphs (KGs) in OK-VQA in a way that simultaneously improves accuracy and interpretability?

To answer the research question, this thesis follows a strategy that can be seen as a set of the following goals:

1. Build and train a chosen baseline model based on a proposed model from the literature that delivers good results and utilizes graph-structured knowledge, but is not interpretable by design.
2. Update the model architecture to enhance the interpretability of the model.
3. Propose updates for the construction of node features representing the utilized KG to improve accuracy.
4. Evaluate the baseline model and its updates to assess if the level of accuracy increases.
5. Build a demo application utilizing the model and its updates to return random examples from the OK-VQA dataset, corresponding predicted answers, and information about them.
6. Perform a qualitative analysis of specific examples to explore the improved interpretability.
7. Discuss whether the proposed updates are suitable for improving the interpretability and the accuracy simultaneously.

The model chosen for (1) is KRISP-GCN, a configuration of the KRISP model from [MCP+21]. The model was proposed in 2021 with one of the coauthors, Kenneth Marino, being also one of the coauthors of the original OK-VQA paper [MRFM19]. When it was proposed, the approach delivered a state-of-the-art result on the OK-VQA benchmark. At the time of writing this thesis, KRISP was overtaken by other approaches on the benchmark, but to this day serves as an important reference point for comparison in evaluations. It is chosen as a reference model to build the baseline model for this project because of its focus on the utilization of KGs within its architecture. The discussion of this choice is expanded within the related work section of this thesis.

The possible contributions of this thesis are based on updates to this architecture. (2) centers on the update of the architecture of the GNN-based submodule, swapping the utilized Graph Convolutional Network (GCN) with a Graph Attention NeTwork (GAT) and thus incorporating an architecture that utilizes the attention mechanism. With this first proposed update, which is called KRISP-GAT in this

thesis, the attention scores that are computed by the model can be returned and further processed. Within this thesis, they will be utilized to visualize the nodes that were rated important to find a target answer by the attention mechanism and for other interpretability-related computations within the case studies.

The other approach to updating the architecture, (3), considers the construction of the KG. When utilizing a KG in KRISP, a vector is added to every node that represents the input data that is the basis for the feature vectors utilizing this data during training. This vector can contain information from different modalities, which (in this case) can be a numerical representation of textual information, visual information, or both. Therefore, within the updates regarding (3), the construction of multimodal and modal-specific KGs is explored. The update KRISP-GAT-VF is proposed as the second update, which enhances the multi-modality of KRISP-GAT by augmenting the node features with visual features representing detected objects in an image. The third update called KRISP-GAT-MS splits these enhanced node features again into two modal-specific modalities.

The updates (2) and (3) are evaluated (4) on the OK-VQA benchmark to ensure that the level of accuracy remains at a comparable level. The evaluation focuses on ablation studies, which are performed to evaluate the actual contributions of the updates.

Within (6), the goal is to explore how the updates actually influenced the interpretability of a model. For this purpose, a demo application (5) based on a Jupyter Notebook is implemented to run the trained models in inference mode, extract the attention scores from the model, and utilize them to interpret how the final predictions were calculated. A selection of specific examples will be documented and discussed within the case studies. A qualitative approach is chosen for this part, because, to the best of my knowledge, no framework providing quantifiable metrics currently exists for the Interpretability Measurement of GNN-based model predictions (see Section 3.4 for further discussion). Due to this limitation, the goal of (6) is simply to gather qualitative examples to discuss whether and what can be interpreted based on the attention scores collected.

Finally, the results from (4) and (6) will be discussed in conclusion in (7).

1.4 Scope and Limitations

This thesis focuses on increasing the interpretability of a model by utilizing the attention mechanism within GNNs. The next step would be to investigate how the interpretation could actually be achieved, or how the attention scores could be utilized. Since this is a different topic with its own set of challenges, an investigation and systematic evaluation of possible interpretation strategies is left for future work. However, some possible strategies are explored in the case study section.

NN-based Learning (or: Deep Learning) is a highly tuneable process with many hyperparameters and possible configurations. The approach within this thesis does not follow the strategy of improving the accuracy by exploring the most perfor-

mant configuration of hyperparameters, but instead focuses on the architecture of the model. Therefore, the hyperparameters and most configurations are taken over from [MCP+21]. The same limitation of scope applies to the exploration of pre-training strategies, which are also discussed in the supplemental material of the [MCP+21] paper. While pre-training of models on other datasets can lead to significantly higher accuracy scores during evaluation, this goal is beyond the scope of this thesis.

Although being out of scope of this project due to the focus on KGs and GNNs, the transformer-based VisualBERT architecture that is utilized within a submodule of the baseline model could also be suitable for enhancing the overall interpretability due to its reliance on the self-attention mechanism.

In addition, the construction of KGs is focused on building node representations within this thesis. Another aspect of KG-construction could be utilizing other excerpts from publicly available sources, or different pruning strategies. These are, however, out of scope of this project, since the focus is on the proposed updates of the underlying architecture and not on modification of the specific content.

1.5 Thesis Outline

The beginning of the main part of the thesis is marked by a contextualization of the VQA field, to consider how it can be placed within the intersection between CV and NLP and how to distinguish it from related tasks. After this, a brief overview of the history of VQA is considered to relate the OK-VQA task to similar tasks by describing a selection of VQA datasets that led to the creation of the OK-VQA dataset, before introducing the dataset itself. The "Background" section is concluded by reviewing other approaches within the OK-VQA field, with an emphasis on approaches that inspired some updates of the baseline approach that was chosen within this thesis.

The purpose of the "Methodology I" section is to provide the theoretical foundation of the research strategy described above, especially considering the baseline model and the proposed updates. This is accomplished by introducing the approach of this thesis as a method of representation learning, which unifies approaches of representation learning within different domains that can be distinguished by the kind of input data they are processing, namely visual, textual and graph-based data.

Another common thread that connects all three sub-topics is the utilization of the attention mechanism. It is first introduced as a general method, before considering one of its most prominent applications within the transformer model, which is the architecture the VisualBERT submodule within the baseline model is built upon. Within this approach, VisualBERT is the source of internal knowledge based on textual and visual input data. Finally, strategies for utilizing different GNN-based architectures are considered, while the attention-based GAT is at the center of all three updates proposed in this thesis.

In general, the "Methodology I" section can be considered as preliminary work to

enable the presentation of the functionality of the baseline model and the proposed updates presented in the "Methodology II" section, where the overall evaluation strategy is also introduced and which consists of two parts. First, quantitative evaluations are carried out in the section "Experiments for Quantitative Evaluation", which consider different configurations of the baseline model and its updates. Second, the consideration of specific examples follows in "Experiments for Qualitative Evaluation".

Finally, the results of both evaluation strategies are discussed along with implications and pointers to possible direction within future work in the "Conclusion" section.

2 Background

2.1 Introduction to the Visual Question Answering Task

The emergence of VQA can be described from the two disciplines it unifies within one task: from the perspective of CV and from the perspective of NLP.

From the perspective of CV, the field of VQA arose from the task of Image Captioning, a task aimed at describing the content of an image in natural language. Within this task, a program takes visual content as input and produces a natural language output, thus combining the fields of CV and NLP within one task. VQA differs from Image Captioning by the amount of questions it aims to handle, since Image Captioning can be understood as always asking one single question about what kind of contents can be found in an image in summary.

Before the field of Image Captioning arose, CV and NLP were often understood as independent fields in the past, since CV is about utilizing input from camera sensors or otherwise obtained image data, while NLP deals with textual or audio data. In [WWW+22], NLP is further contrasted by the goal of teaching machines to learn how to read, as opposed to teaching machines how to see within CV. Both domains are, however, somewhat intervened due to similar methods. In fact, in recent years both domains have started to show signs of convergence due to both utilizing transformer-based models and the trend of multi-modal models in general [LS23], [ZLW+22].

The task of Image Captioning can be seen as a first unification of CV and NLP within one task. VQA can be understood as the next development of this unification that aims to answer arbitrary questions about arbitrary images, and can be seen as a function $A = VQA(I, Q)$ that has an image I and a corresponding question Q as an input and an answer A as its output.

There are several subfields of VQA, which can be specified by the three different components:

- The answers can be binary yes/no answers, free text multi-class classifications, multiply-choice classifications or aim at open-ended answer generation.
- The selection of images can indicate the necessity of domain-specific knowledge, like in Medical VQA.
- The questions can be answered by the content given in an image or by having to utilize outside knowledge, as in OK-VQA.

VQA can also be understood from the perspective of the development within the NLP field, where there is a large subfield of Question Answering (QA) based on textual input. VQA can be seen as a superset of QA tasks, where textual information alone does not suffice to answer a question and further utilization of visual information is needed. Within this process, visual information like data from images (or videos in Video Question Answering) also introduces significantly more information that might not be directly relevant for the task. At the same time, textual QA

mostly answers questions based on a given context, which are usually more narrow and tailored to finding an answer.

2.2 Related VQA Datasets

Similar to related research fields within NN-based ML, VQA is largely driven by the introduction of new datasets, created for the development of new algorithms and the evaluation of the proposed algorithms, allowing benchmarks to track progress within the fields. Therefore, the history of progress on the VQA task can be told as the history of its datasets (which was done in [WWW+22], for example, where also an overview of the domain-specific datasets for fields such as Medical VQA can be found). The focus of the following section is, however, on a selection of datasets that can be understood as especially relevant predecessors to the OK-VQA dataset to explore how various settings of this task can differ in practice.

One of the first large datasets for VQA was Visual Question Answering [AAL+15]. It aimed at popularizing the VQA task with the goal of laying the groundwork for new AI algorithms. The authors argued that the task of Image Captioning as the field that previously unified CV and NLP was not "AI-complete", unlike VQA. Thus, the need for a new task was identified, one that focuses on free-form and open-ended questions while considering both modalities. The VQA dataset consists of a subset of images from the MS COCO dataset, which is a popular image dataset for tasks like Image Captioning containing a diverse set of real-life objects and scenes. For each image in the dataset, human annotators provided a set of open-ended questions related to the image. Additionally, the VQA dataset includes ground-truth answers (mostly one worded), provided by the annotators. The answers can be either multiple-choice or free-form text. For multiple-choice questions, the dataset provides a set of candidate answers with a labeled correct answer. The dataset also provides answers to questions that could be considered plausible but are likely incorrect.

A second version of the dataset, VQAv2 [GKS+17], was later released as an update to VQA as an extended version that also addresses data balancing issues within the dataset. For example, if a question starts with "Do you see...", a strategy of always returning "yes" as an answer without even considering the image would yield 87 % [WWW+22]. VQA is available in both versions and also in different subversions that, for example, focus on synthetic image data instead of real-life data.

In the [MCP+21] paper that describes the model that was chosen as the baseline model for this thesis, the results of an approach to use a pre-training strategy by training the KRISP model on the VQAv2 dataset before training it on OK-VQA are also documented. This strategy leads to a considerable improvement in accuracy of around 6.5 %, which is intuitively explained by the similarity of the two tasks.

Another family of VQA datasets was introduced around the Visual Genome dataset [KZG+16] that was introduced for different CV tasks such as Object Recognition, Image Captioning, Scene Understanding and Visual Question Answering. The

dataset has a focus on visual relations and thus contains annotations to images that include information on visual relationships and scene graphs in addition to image annotations.

There are other datasets within this field that, for example, focus on visual reasoning (like spatial reasoning on image data) such as CLEVR (Compositional Language and Elementary Visual Reasoning [JHvdM+16]). It contains synthetic images of objects with different shapes, colors, materials, and sizes. The dataset focuses on compositional questions, such as "What color is the cube to the right of the yellow sphere?" and also comes with structured scene representations in the form of scene graphs, as well as answers to the questions. The dataset is of interest, since it represents another approach of bringing neuro-symbolic strategies to the VQA field by combining the output of neural networks with logic programming, like [EHOP22] where Answer Set Programming (ASP) is utilized to compute answers to questions based on confidences from a trained object detection NN using an ASP solver. These approaches are beyond the scope of this thesis, but could be promising research directions for future work.

All of these datasets have one thing in common: They consist of image/question pairs that can be answered by the information given in an image alone. The first dataset that focused on outside knowledge and thus paved the way for the OK-VQA dataset was FVQA (Fact-based Visual Question Answering) [WWS+17]. However, this dataset comes with a fixed set of knowledge triples as answers from a preconstructed knowledge base, while OK-VQA has simple strings as candidate answers and thus allows for the utilization of a variety of knowledge sources.

2.3 The Outside Knowledge Visual Question Answering Task

The motivation for the construction of the OK-VQA dataset, as described in the accompanying paper [MRFM19], was to move the VQA field towards models that not only recognize visual content, but also incorporate logical reasoning within their approaches, as well as knowledge about the world that cannot be answered by the visual content itself.

An example[2] from the dataset can be seen in Fig. 1. Since the answer "grapes" is neither in the image nor in the question, the given data has to be somehow connected with an outside knowledge source that, for example, contains the knowledge triplet (wine, madeFrom, grapes). The OK-VQA data set contains 14,055 open-ended questions and corresponding images like this example, which are split into different categories and over two parts of the dataset: A training dataset that contains 9,009 image/question pairs and a test dataset that serves as the dataset for the actual evaluation and contains 5,046 image/question pairs, while human annotators provided the corresponding answers. Recently, updated datasets like A-OKVQA [SKC+22] were introduced to further improve several aspects, such as the quality of the questions or the amount of knowledge that should be used to answer a question.

[2]Source: https://okvqa.allenai.org/browse.html, accessed: 27.07.2023

Question : What fruit is this beverage made of?

Answer: grapes
Answer Occurence: 5 / 5
Category: People and Everyday life

Figure 1: An example from the OK-VQA dataset.

Since the evaluation of different proposed updates to the original OK-VQA dataset is outside the scope of this thesis, their exploration is left for future work.

2.4 Related Work: OK-VQA approaches

In this section, some selected approaches to the OK-VQA task will be considered. The goal of this section is not to provide an extensive survey of all available approaches that can be found in the literature, but to set a context of how the chosen baseline model for this thesis and its updates relate to other approaches in the field. The section is divided into two parts. First, a selection of NLP-centric approaches is discussed. These approaches mostly understand the OK-VQA task as an NLP problem like Text Generation, and thus are not within the focus of this thesis. They are, however, an important part within the field and can be considered as another main cluster of approaches to contrast graph-centric strategies, which are considered in the second section. Furthermore, the two categories are not exclusive, since some approaches such as LaKo[CHC+22] utilize KGs as an external source, despite understanding OK-VQA as an NLP task.

2.4.1 Natural Language Processing-centric Approaches

The appropriately named paper "A Thousand Words Are Worth More Than a Picture" from 2022 [GPT+22] proposed the Transform-Retrieve-Generate (TRiG) model, while understanding the OK-VQA task mainly as an NLP problem. This ap-

proach translates the OK-VQA task to the task of Generative Question Answering, where an image is transformed into a verbose image caption as a first step, which is utilized to retrieve passages from Wikipedia based on scores from a semantic similarity search. Finally, an answer prediction is generated by an encoder-decoder transformer based on those inputs.

Another NLP-centric approach, LaKo (Late Knowledge-to-text Injection), was proposed in the end of 2022 [CHC+22]. This approach utilizes a KG by transforming knowledge triples from a KG into a textual format for further processing. Within this approach, the OK-VQA task is addressed as a text generation task. As a first step, captions are generated on the basis of the input image. Like QA strategies in the NLP domain, this approach is centered on providing a given context to a given question – which in this case is provided by the generated captions. Furthermore, the given question and the image captions serve as the basis for retrieving knowledge triples from a KG. After the transformation of the triples into textual format, the textual information serves as input to an encoder-decoder transformer-based submodule that returns the final prediction. According to the evaluation in [CHC+22],

LaKo outperforms KRISP with an accuracy score of 47.01 on the OK-VQA dataset. This approach is also interesting from the perspective of interpretability, since it is one of the approaches that does not incorporate a KG within a GNN and thus remains its inherent interpretability. The TRiG approach also outperforms KRISP with an accuracy score of 50.50 [GPT+22] and has also advantages in terms of interpretability, since the generated and retrieved information can be used to give clues about the reasoning process. Although NLP-centric approaches are beyond the scope of this project, a combination of these strategies with graph-centric approaches might be interesting for future work. Especially considering that the greatest advances in the OK-VQA task in the recent time frame can be attributed to the use of newer GPT-3 models in NLP-centric approaches, such as the Knowledge Augmented Transformer (KAT) [GWH+22] with an accuracy score of 54.41 or the approach from [SYWY23] with an accuracy score of 61.1.

2.4.2 Graph-centric Approaches

In the following, a selection of approaches is considered that are more related to the architecture of the selected baseline model within this thesis. Their main difference lies in different strategies for utilizing KGs within their architectures.

MUCKO

The MUCKO (Multi-Layer Cross-Modal Knowledge Reasoning) model was introduced in 2020 [ZYW+20] and was one of the first solutions proposed for the OK-VQA task, scoring 29.20 on the accuracy metric. This approach is described in

more detail than the other examples, because its utilization of modal-specific KGs can be seen as a general approach that is also used in other examples [JM23] and is also an inspiration for one of the updates to the KRISP architecture proposed in this thesis.

At the center of the MUCKO approach is the construction of three aforementioned modal-specific subgraphs that are utilized within submodules of the overall model.

The first constructed graph is a **Visual Graph**. An image that is given as part of the input is evaluated with Faster R-CNN [RHGS15] for object detection. After a set of K objects $\mathcal{O} = \{o_i\}_{i=1}^{K}$ with $K = 36$ is returned, every object in that set serves as the basis for a node in the final visual graph that contains the following information represented by features, where d is the number of dimensions in a vector:

- A visual feature vector $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ $(d_v = 2048)$
- A spatial vector $\boldsymbol{b}_i \in \mathbb{R}^{d_b}$ $(d_b = 4)$ that represents the bounding boxes of the detected objects in the image.
- a corresponding label

For the final Visual Graph $\mathcal{G}^V = \left(\mathcal{V}^V, \mathcal{E}^V\right)$ over $\mathcal{O}$, where $\mathcal{V}^V = \left\{v_i^V\right\}_{i=1}^{K}$, the nodes v_i^V are connected by the edges $e_{ij}^V \in \mathcal{E}^V$ where each edge represents the relative spatial relationship between two objects.

A **Semantic Graph** is constructed after the utilization of dense captions based on the approach from [JKFF16], where an image is transformed into D dense captions. The generated captions are z_i descriptions of local regions in the image that comprise the set of $Z = \{z_i\}_{i=1}^{D}$. The generated captions are further processed by the semantic graph parsing model [AFJG16], so for example, the string "A fire hydrant on the sidewalk" is returned as the knowledge triple (Fire hydrant, On, Sidewalk). The resulting graph is denoted as $\mathcal{G}^S = \left(\mathcal{V}^S, \mathcal{E}^S\right)$, where every node $v_i^S \in \mathcal{V}^S$ represents an object label and every edge $e_{ij}^S \in \mathcal{E}^S$ represents the relationship between the nodes v_i^S and v_j^S.

Finally, a **Fact Graph** is constructed by retrieving relevant facts from a predefined knowledge base by computing the cosine similarity between words from the question and the descriptions of the detected objects. Based on the 100 top scored facts, the Graph $\mathcal{G}^F = \left(\mathcal{V}^F, \mathcal{E}^F\right)$ is constructed, where each node v_i^F is represented by an embedding of a fact, and each edge $e_{ij}^F \in \mathcal{E}^F$ represents the relationship between the corresponding nodes.

Following the creation of the graphs, Graph Convolutional Networks are utilized on each for generating embeddings that are later combined within the final layer. This way, the last layer considers all three modal-specific subgraphs and produces a prediction based on the overall context.

MuKEA

The model MuKEA (Multimodal Knowledge Extraction and Accumulation for Knowledge-based Visual Question Answering) was proposed in 2022 and achieves an accuracy score of 42.59 on the OK-VQA benchmark. This approach is an interesting contrast to the role of KGs within its architecture: Instead of utilizing KGs from publicly available sources, a KG is learned. At the center of the approach is the goal of learning multi-modal knowledge triples to construct a KG that correlates visual objects, internal knowledge, and external knowledge in the form of (visualInformation, relation, textLabel). The resulting KG is used to directly infer an answer corresponding to a given image/question pair, which can be seen as predicting the tail (or: the textLabel) from the accumulated knowledge triples: The paper understands tasks like OK-VQA as a multimodal knowledge graph completion problem. The algorithm is trained on the VQA and the OK-VQA datasets. Besides this, the only other source for knowledge utilized in this approach is a transformer-based model. The learning-based construction of a KG from given datasets is a promising approach and could be combined with the strategies that are in the focus of this thesis in future work.

MSG-KRM

Recently, a new model was proposed that can be seen as a combination of the above approaches and achieves an accuracy score of 43.58. The MSG-KRM (Multi-Modal Semantic Graph Knowledge Reasoning Model) was proposed in May 2023 [JM23]. While the above approaches work with one (MuKEA) or two (MUCKO, KRISP) knowledge sources, MSG-KRM considers three different types of external knowledge: Structured information represented by knowledge graphs, unstructured knowledge in the form of retrieved external texts and non-symbolic (or internal) knowledge from pre-trained models while also generating Knowledge Graphs based on these sources.

Similarly to MUCKO, MSG-KRM constructs a MMKG by utilizing three separate KGs for integrating knowledge from the three sources above, introducing both symbolic and non-symbolic nodes in the process. Unlike within the MUCKO architecture, information on the type of edges is also evaluated during by the utilized GNN, which itself is based on GAT. This allows for increased interpretability of the model, which, like in MUCKO, considers the intermediate layers handling the modal-specific KGs. However, the output from these layers is passed to a further layer in both cases, which hinders the interpretability of how the final predictions came to be.

KRISP

The KRISP (Knowledge Reasoning with Implicit and Symbolic rePresentations) model was introduced in 2021 [MCP+21] and achieved an accuracy of 32.31 on the OK-VQA dataset. In the accompanying paper, another configuration is considered in which the model is also pre-trained on the VQA dataset, a strategy that raises the accuracy score further to 38.90. Due to its significance within this thesis, the details of the model are considered in the following sections. Within this section, the goal is to contextualize the KRISP approach in the general OK-VQA field to give reasons why it was chosen as a baseline model:

- KRISP considers internal and symbolic knowledge and can be considered as a representative example of approaches that follow this strategy due to its popularity and good results on the OK-VQA benchmark.
- KRISP is not interpretable by design, but the interpretability could be improved by some updates proposed in this thesis.
- Within the KRISP approach, a backbone KG is pre-constructed and then dynamically updated during training with the construction of node input data. This strategy is suitable for the exploration of possible updates in the second construction phase, which is the focus of some updates proposed in this thesis. Therefore, this architecture is well suited to evaluate the influence of its individual components and its updates within ablation studies.
- The component that handles graphs within KRISP is one of its last layers within the overall architecture, so the results generated here can be considered as the basis for the final predictions.

3 Methodology I: Representation Learning for OK-VQA

There are several possible sources for retrieving knowledge within the OK-VQA task. A common differentiation is to distinguish between implicit and explicit/symbolic knowledge [LS23].

Implicit knowledge is knowledge that can be retrieved from pre-trained NN models. It is non-symbolic, meaning that only the trained weights of a NN represent some sort of knowledge. However, these weights are not interpretable for humans out of the box, since they are available only as a very large set of numerical values.

Although neural networks suffer from an increased difficulty of interpretability, they also deliver impressive results on a variety of tasks. They are also an integral part of the baseline model of this thesis and the proposed updates:

- The NN-based object detector, Faster-RCNN [RHGS15], is utilized for various steps within the model, starting with data pre-processing. Its outputs are also utilized within one of the proposed updates.
- The NN-based Bidirectional Encoder Representations from Transformers (BERT) architecture [DCLT19] is utilized within the handling of textual data and also serves as the backbone of the VisualBERT-based [LYY+19a] submodule, which combines the processing of image and textual data within one model.
- The architectures for processing the constructed KGs are the Graph Neural Network architectures GCN [KW16] and GAT [VCC+18].

Although all of these parts are NN-based, only the VisualBERT submodule and its utilization of a pretrained BERT model can be considered as an actual source for internal knowledge within KRISP, since this part utilizes the knowledge from a pretrained transformer-based model and is also the place where the weights for returning an actual answer corresponding to an image/question pair are trained.

As discussed above, internal knowledge has one shortcoming: the difficulty of interpretability. This poses difficulties in evaluating a model with respect to the question of where it might have gotten the answers wrong and how these answers came to be. For example, because of the black-box nature of the NN-based models, it is not clear if specific information retrieved from the model is missing within the dataset it was trained on, or even wrong.

Another source of knowledge is explicit knowledge [LS23], or **symbolic knowledge** [MCP+21], which consists of unstructured or structured knowledge, such as in the form KGs. There are various large KGs publicly available that are usually utilized in OK-VQA research. Although the knowledge within a KG is interpretable by design, the utilization of KGs comes with some challenges of its own.

First, a KG has to be constructed based on excerpts from available KGs since the vast majority of the information in most publicly available KGs is not relevant to the

specific task. Second, the constructed KG is equipped with node features that can be further processed by a GNN. There are several strategies that can be considered regarding the construction of node features. And there is the issue of interpreting a GNN. Within the KRISP approach, the general structure of the input and output KG of the GNN is the same. The actual learning takes place by generating representations from the node features, which are not easily interpretable for humans.

In fact, this point can actually be seen as the saddle point of this thesis, because here is where the first proposed update to the KRISP architecture comes into play: The GNN architecture used in KRISP can be replaced with another architecture that is more suitable for interpreting the prediction results.

This section of the thesis is devoted to the exploration of the methodology of how an OK-VQA model can be built, following the KRISP approach, which also serves as the model proposed as a baseline within this thesis in a specific configuration. Furthermore, the processing of the input data is considered in regard to the processing of textual, visual, and graph data for representation learning within this approach. Finally, the preliminary work of how this baseline approach can be updated is presented, regarding the focus of enhancing the interpretability of the model by utilizing a different GNN architecture and by exploring different strategies of constructing (multimodal) KGs. Since the updates around the handling of symbolic knowledge are the main focus of the thesis, the corresponding sections are more detailed compared to their counterpart regarding the processing of implicit knowledge. However, one component of some prominent NN-based architectures will be especially considered, since it is vital for both, the handling of implicit knowledge and the handling of symbolic knowledge after one of the proposed updates to the KRISP architecture: The attention mechanism.

3.1 Preliminaries

Like with other Machine Learning fields, at its core, Machine Learning by utilizing NNs can be understood as an approach to Representation Learning. The first part of training a model is to transform input data into vectors (or: embeddings) that represent the input data within the hidden layers of a network and thus enable it to perform operations like backpropagation. The goal of NN-based Representation Learning is to generate increasingly useful representations by combining several non-linear transformations [WCPZ22].

Appropriate representations are an integral part of designing a NN architecture, as they are one of the parts that determine the performance of the overall NN-based learning algorithm. Therefore, the goal is to transform the input data in a way that represents them in a sufficient, but minimal way, with feature engineering being the field devoted to this task [WCPZ22]. Feature engineering itself is intervened with the task of data preprocessing, since the original data has to be preprocessed in a standardized way, before becoming the input data of an NN.

According to [WCPZ22], the field of deep learning-based representation learning

can be divided into the following subfields that are generally congruent to other Machine Learning approaches:

1. Supervised Learning, which requires a substantial amount of labeled data for training. The goal of the training is to learn a mapping between the input data and the corresponding labels so that the model can make predictions on unseen data.

2. Unsupervised Learning, which usually also requires a large amount of data, but is not dependent on labels. Here, the goal is to discover an underlying structure, specific patterns, and relationships within the data.

3. Transfer Learning utilizes pre-trained models as a starting point for further training the model in a new, usually related, task.

Within the approach chosen for the thesis, as well as its updates, the focus is primarily on (1). Here, the OK-VQA task is understood as a multi-class classification problem and a provided dataset with labeled data to solve it. Like in image classification tasks, a given image is classified within a set of possible answers as classes. (3) comes into play within the VisualBERT module, since a pre-trained BERT module is utilized here as a starting point. Despite it also being a pre-trained model, the utilized object detection algorithm Faster R-CNN, cannot be summarized under (3), because it is only utilized during the processing of the data, but is not actually trained within the overall model.

Different strategies of utilizing NNs within different modalities were established within the last years, while there is also a trend of utilizing unifying architectures based on transformer-models for some modalities. Before we move on to these architectures, we will first consider some preliminaries for NNs in general.

3.1.1 Neural Networks

An NN consists of basic building blocks called hidden units (or: neurons). A hidden unit can be visualized as in Fig. 2 and defined as follows:

Definition 1 (based on [Vas19]):

$$z = f\left(y\right), \text{ where } \quad y = \sum_{i=1}^{n} x_i w_i + b$$

, where

- z is the single-valued output of a hidden unit.
- f is an activation function. Activation functions introduce non-linearity in NNs to enable the approximation of non-linear functions. They are also differentiable, so gradient descent can be performed during backpropagation.

- y is the sum of all hidden unit inputs x_i (with n total inputs), weighted by the weights w_i. Additionally, another weight, the bias weight b, is added to the sum. x_i can either be the input data of the NN, or the output of another hidden unit.

This definition can also be rewritten as: $y = f(\mathbf{x} \cdot \mathbf{w} + b)$ [Vas19], where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ substitute x_i and w_i, so the dot product of the vector representations can be calculated instead of the sum (which is the operation performed in the actual calculation during training on GPUs, which are heavily optimized for parallel matrix multiplications).

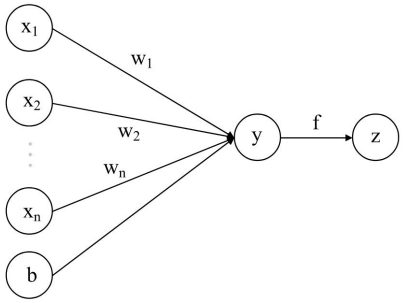

Figure 2: A visualization of a hidden unit [WWW+22]

Neural networks are usually made up of at least one layer, where a layer is the combination of multiple hidden units within one output vector [Vas19]. The classical structure of NNs is built from fully connected layers, where every hidden unit of a layer is connected to every element of the input data. With the rise of Deep Learning, many layer types were introduced, e.g., convolutional layers that are utilized in many image data-based architectures.

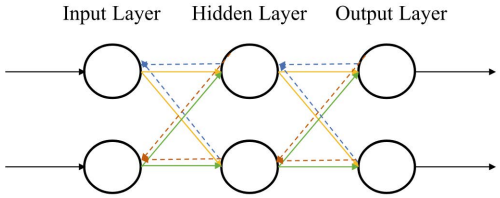

Figure 3: A simple Neural Network Architecture [WWW+22].

A simple version of an NN with three fully connected layers can be seen in Fig. 3 from [WWW+22], which illustrates the two main processes of training NNs: forward calculation and backpropagation.

During the forward calculation (green and yellow arrows), the input data is passed through each layer, resulting in the following operation for calculating the

outputs A_2 and A_3 (based on [WWW+22]):

$$\begin{aligned} Z_2 &= f(W_1 A_1 + B_2) \quad A_2 = f(Z_2) \\ Z_3 &= f(W_2 A_2 + B_3) \quad A_3 = f(Z_3) \end{aligned} \tag{1}$$

A_1 represents the input vector, while the W and B values represent the learned weights and biases.

Activation Functions

There are several activation functions available for different use cases. In the following, we will define the activation functions utilized within this thesis:

Definition 2 (based on [WWW+22]):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \text{ , with } x \in \mathbb{R}$$

Is the sigmoid function, with which every output is bounded between 0 and 1. In this way, the output can also be interpreted stochastically, since it corresponds to a probability value [Vas19]. The sigmoid function is often used in the output layer of an NN for multi-class classification problems, where each unit output represents the probability of a certain class.

Definition 3 (based on [WCPZ22]):

$$\mathrm{ReLU}(x) = max(0, x) \text{ , with } x \in \mathbb{R}$$

Is the function that represents the ReLU, the REctified Linear Unit, which maps each output to 0 if $x < 0$ and retains the output otherwise. Compared to the sigmoid function, it is computationally simpler and is therefore favored within the in-between states of a NN [Vas19]. Another activation function used in several approaches to classification problems is the softmax function:

Definition 4 (based on [Vas19]):

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}$$

, where $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ is the vector that contains the output of the final layer and thus contains the probabilities of n possible classes. The softmax function returns the probabilities as normalized values between 0 and 1.

Backpropagation

Backpropagation is the central part of how training is actually achieved in supervised NN-based learning algorithms. With backpropagation, the gradient of a given loss function (e.g., the Binary Cross Entropy loss function in KRISP [MCP+21]) is continuously updated with respect to the weights and biases of the NN, while the goal is to minimize the loss function and thus minimizing the difference between predicted output values and target values that are part of the labeled data in a dataset.

Backpropagation consists of the following steps that are repeated over and over during a training run (based on [Vas19], where also a lower level overview of the actual calculations can be found):

1. Initialize the NN weights with random values
2. Forward pass: Feed the training data into the NN
3. Loss calculation: Compute the given loss function comparing the predicted output and the target output
4. Backward pass: Compute the derivative of the given loss function with respect to the NN weights
5. Update the NN weights wrt. to calculated derivatives
6. Repeat steps 2-5

3.1.2 The Attention Mechanism

The goal of the attention mechanism is to represent the relational importance between elements by assigning them learnable score values. Within Deep Learning Approaches, this mechanism was popularized in the field of NLP field of machine translation after the application in [BCB16], but also had a great influence on other NN-based approaches, such as computer vision, due to it being a fundamental part of the transformer architecture.
The general attention framework can be defined as follows:

Definition 5 (based on [WWW+22], [Vas19]):

$$O = \text{Attention}(Q, K, V)$$

, where Q is a query, K and V are keys and values, and O is the output.

The attention mechanism returns a mapping between Q and a set of key-value pairs represented by K and V. The general steps of the mechanism are as follows (based on [WWW+22]):

1. Input Q, K and V into the $\mathrm{Attention}$ function
2. Calculate scores with $s_i = \mathrm{Score}\,(query, key_i)$, where $query$ is used to find the related key_i
3. Normalize all s_i outputs with a $softmax$ function, resulting in attention scores a_i
4. Aggregate the outputs based on their attention scores a_i and corresponding $values_i$ within a weighted sum, which is the output O of the attention mechanism

The actual $\mathrm{Attention}$ and Score functions are specified in their actual context, like within the transformer model and within the GAT model, as described in the according sections of this thesis.

3.2 Representation Learning on Image Data

3.2.1 Convolutional Neural Network-based Architectures

One of the first steps in a VQA pipeline usually involves the processing of image data. A very common approach is to utilize CNN-based models for this task. Before the current trend of using transformer-based architectures to represent images (or regions of images) by NNs, the most common architecture within supervised representation learning was the Convolutional Neural Network (CNN) [LS23]. The name-giving convolution operation was already widely utilized within image processing before the rise of NN-based architectures on tasks like Feature Extraction, Filtering, Edge Detection or Image Transformation. It can be understood as combining two functions to produce a third function, which represents their combined information.

CNNs are typically constructed by stacking four types of layers (based on [ON15]):

1. An input layer that receives the input data, e.g., a matrix of pixel values. In the case of image data, the input data can be rather large, often containing thousands or millions of pixels, often in triplicate to contain the information from three separate color channels.
2. The convolutional layer is responsible for extracting features from the input data. This is achieved by applying a set of filters (or: kernels) to the input data. Each kernel is a matrix of weights, which is convolved with the input data by an approach similar to a sliding window: While the kernel is sliding over the input data, the dot product between the kernel and the corresponding region of the input is computed, resulting in a feature map that highlights certain spatial patterns or features. The weights of the kernel are the learnable parameters that are adjusted during the training process by backpropagation.

The output of this layer is passed to an activation function like ReLU for introducing non-linearity to the network and thus enabling the learning of more complex patterns.

3. The pooling layer is responsible for downsampling the feature maps while preserving important information and applying pooling techniques.

4. The fully-connected layer is similar to the same structure in other NNs: This layer flattens the feature maps into a 1d vector and is responsible for learning representations and producing class probabilities from the activations.

There are different strategies for constructing a CNN architecture, but it usually involves combinations based on these building blocks [ON15]. Their common goal can be summarized as learning a lower dimensional representation of the input data, which usually is rather due to the amount of pixel information.

3.2.2 Object Detection Based on Representation Learning With Faster R-CNN

The embedding strategies of image data can be divided into the involvement of image level data or region-based data. Generating embeddings for a complete image can be utilized in a variety of tasks, such as image classification. Region-based data becomes important if the goal is not to compare whole images with each other but to extract specific regions from images, like bounding boxes of detected objects. Pretrained Object Detectors like Faster R-CNN (Region-based Convolutional Neural Network) [RHGS15] can be utilized for the latter [LS23].

The Faster R-CNN algorithm is an integral part of the preprocessing of image data within the KRISP approach. The region-based features of the objects are also utilized within one of the updates to the KRISP architecture proposed in this thesis. Hence, the following serves as a brief overview of the architecture behind Faster R-CNN.

Faster R-CNN has two predecessors: R-CNN [GDDM14] and Fast R-CNN [Gir15]. R-CNN was proposed in 2013 and was the first model based on a CNN architecture that introduces the idea of region proposal-based object detection. At first, region proposals are generated while identifying potential object bounding boxes as regions within the input image. Next, each of these proposed regions is fed into a CNN to extract features. The output is further fed into a set of fully-connected layers for object classification.

The original R-CNN suffered from some drawbacks regarding the training and inference time, which could mainly be attributed to treating each object detection as a separate task. To overcome these drawbacks, Fast R-CNN [Gir15] was introduced in 2015. Here, instead of processing each region proposal independently, the extraction of features is performed on the entire image only once by utilizing a CNN. Then, a region of interest pooling layer is introduced, which extracts fixed-size feature vectors based on every proposed region.

The further improved version, Faster R-CNN [RHGS15] was also introduced in 2015. It addresses another bottleneck of its predecessors: their reliance on external

region proposal algorithms. Here, a Region Proposal Network (RPN) is introduced that generates region proposals directly within the model. In the paper [RHGS15], the authors draw a parallel to the attention mechanism: The RPN tells the Fast R-CNN module where to look. The RPN is combined with the Fast R-CNN network and shares some convolutional features with it directly, and thus enables faster object detection.

3.3 Representation Learning on Textual Data

Compared to image data, representation learning on textual data is challenged by the sheer variety of forms the textual data can take. An image is usually a structure with clear dimensions, like a grid of pixels. A natural language text can be comprised of one sign or millions; it can come in different languages, etc. Hence, the preparation of the input data for NN-based architectures is not as straightforward as with image data and will receive further consideration within this section, before the actual architectures of NNs for textual inputs are considered.

3.3.1 Encoding Textual Questions

NN architectures work with fixed-size input vectors, so, for example, given a dataset comprised of textual data, a way of standardizing the data is needed. This is accomplished by generating word embeddings, which represent words or tokens as dense numerical vectors. Tokens are parts of words that were split up for further processing; their length and structure depend on the given tokenization algorithm. Word embeddings can be seen as a lookup table that maps words or tokens to their dense vector representation. These vectors are the basis for NN-based representation learning and can capture the semantic and syntactic regularities between words when subjected to a machine learning algorithm.

There are several techniques to acquire word embeddings. One of them is to use embeddings from the pre-trained GloVe model (Global Vectors for Word Representation), introduced in [PSM14]. GloVe is a popular unsupervised learning algorithm for generating word embeddings. Here, the key idea is to consider the statistics of word co-occurrence within a large textual dataset (including a complete dump of Wikipedia). The term "global" in GloVe refers to the fact that the model considers the co-occurrence of words over the complete dataset, unlike methods that only consider local text information (the immediate surroundings of where the word occurs). Thus, the method aims at embedding the meaning of a word within a vector that is not dependent on the local context: Every word in the dataset is mapped to the vector space. This way, the approach is suitable for tasks that want to utilize a pre-trained model for retrieving dense vector representations, like within KRISP.

Another word embedding technique is to utilize a pre-trained BERT model for this task, which is discussed in the corresponding section of this thesis below.

3.3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are centered around the idea of sequential processing of elements within a finite sequence $(x_1, x_2, \ldots, x_T)$, one by one [LS23]. During this process, the context based on the left side of the sequence from the current element is retained, and both the context and the current element are utilized for computation of the current element's feature representation. Although sequential processing corresponds to the way humans process a text, it suffers from the same challenge: The right-hand side of the current element that is being processed is not considered at all. The BERT architecture addresses [DCLT19] this challenge, as described in the following section.

3.3.3 The Transformer Model

Similarly to other approaches within the OK-VQA field, the main source of internal knowledge within the KRISP approach is a transformer-based model. In the following, we will consider the transformer architecture, before we move forward to BERT and VisualBERT, the models that are actually utilized in KRISP.

The transformer model was introduced in 2017 [VSP+17] and was designed primarily for sequence-to-sequence tasks such as machine translation, where a sequence of textual input is translated from one language to another.

The two key components of the architecture are the encoder and the decoder for processing and generating sequences, respectively. The encoder/decoder structure was already commonly utilized within RNN-approaches before the transformer [WWW+22] and is further improved within this approach. Both components are composed of layers of self-attention and feed-forward NNs, but differ in their roles and corresponding operations within the model.

The self-attention mechanism is a key component of the transformer model. It calculates attention scores by computing the dot product between the query vector Q, the key vector K, and the value vector V, which are derived from the input sequence. The attention scores determine the importance of different elements in the sequence, allowing the model to focus on the relevant information. Multi-head attention is used to perform multiple independent self-attention operations in parallel, and the results are concatenated and linearly transformed to obtain the final output [VSP+17].

The particular attention method was called "Scaled Dot-Product Attention" in the original paper [VSP+17] and can be calculated as follows for each head:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

, where d_k is the dimension of each key vector in K (this formula represents the formula that is used in practice when computing attention, where Q, K and V refer to the matrices that collect the corresponding vectors).

The goal here is to compute the dot products between the queries and the keys, and scale and normalize the results with the softmax function, so they become probabilities that represent the initial importance of a specific key to a specific query.

The results from the individual heads are accumulated as follows (based on [WWW+22]):

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}\,(\ \text{head}_1,\ \text{head}_2, \dots\ \text{head}_h)\, W^o \\ \text{, where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW^V\right) \end{aligned} \tag{3}$$

, where W are the weights, where the actual representation learning takes place and W^o are the weights corresponding to the output o.

The Encoder

The input data is first processed by the encoder, which takes a textual sequence as input and outputs a sequence of contextualized representations. Its role is to capture the meaning of the input sequence and encode it into an abstract representation. The encoder can be seen on the left side of the architecture in Fig. 4 consists of $N = 6$ identical layers, which themselves contain two sublayers representing:

1. A Multi-Head self-attention mechanism, where Q, K and V to calculate Eq. 2 are either based on the input data or the output of the previous encoder layer. This is where the importance of the relations between the words in a sequence is initially weighted.
2. A position-wise fully connected feed-forward network. This sublayer is responsible for the actual learning of patterns and relations within the input data.

The Decoder

The decoder generates an output sequence based on the output of the encoder. The decoder can be seen on the right side of Fig. 4 and also contains several identical layers that consist of:

1. A masked multi-head self-attention mechanism, which processes the output of the encoder and ensures that only the previous positions in the output sequence are attended by masking out all others.
2. A multi-head encoder-decoder attention mechanism, where K and V are part of the output from the encoder and Q are the queries from the decoder
3. A position-wise fully connected feed-forward network that has the same function as the corresponding sublayer in the encoder.

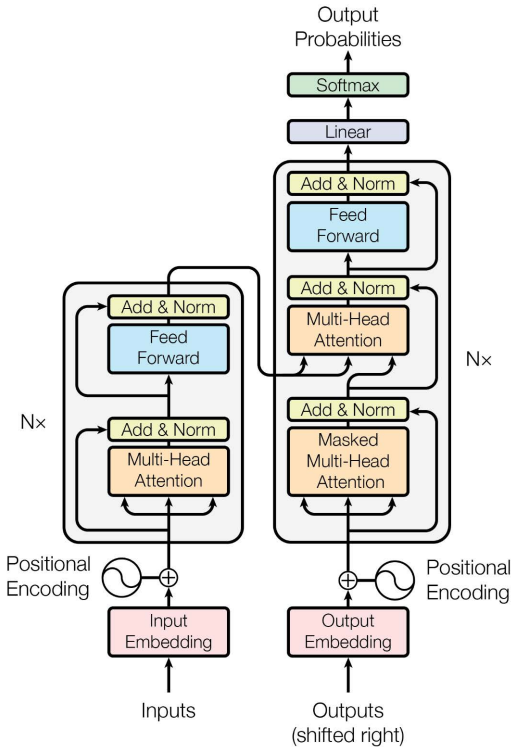

Figure 4: The Visualization of the transformer model [VSP+17].

The Transformer model also incorporates positional encodings, which provide information about the order of elements in the sequence. Positional encodings are added to the input embeddings, allowing the model to capture sequential information.

These building blocks serve as main components for different transformer-based models, which can be divided into three general approaches:

1. Encoder-only architectures that are mostly used within approaches that focus on the learning of a representation from an input sequence, like BERT [DCLT19]

2. Decoder-only architectures that focus on tasks like text generation, like the InstructGPT model [OWJ+22]

3. Encoder-Decoder architectures like the original transformer described in this section.

An overview of how different transformer architectures within current OK-VQA approaches can be found in [CHC+22].

3.3.4 The Bidirectional Encoder Representations from Transformers Model

The Bidirectional Encoder Representations from Transformers (BERT) model is one of the first Large Language Models (LLMs) that was pre-trained on a large amount of textual data with the goal of providing a simple way of fine-tuning the model with labeled data for downstream tasks such as question answering or language understanding.

Before BERT, algorithms for creating embeddings of words like GloVe [PSM14] captured the context of the sequence in one direction, e.g., by evaluating a given text from left to right. BERT considers both directions, thus contextualizing both its preceding and succeeding words. This is enabled by the transformer architecture, since its purpose is to capture the relations between all words of a given text by utilizing the self-attention mechanism, regardless of their relative positions.

The pretraining of BERT consists of unsupervised learning on a large corpus of unlabeled text with the objective of learning word prediction by, for example, masking out a word in a sentence and sequentially predicting these words based on the context from both directions [DCLT19].

Since BERT is designed for language understanding and processing and not text generation, it focuses solely on the encoder part of the transformer architecture and does not have a decoder part.

3.4 VisualBERT: Combining visual and textual representations

VisualBERT builds up on BERT [DCLT19] that was proposed within the realm of NLP. It combines BERT and outputs from object detection models such as Faster R-CNN. Within this approach, image features are extracted from detected objects and become part of the BERT inputs by being treated as unordered input tokens, so both modalities are processed jointly throughout the BERT layers. The main idea of the approach is to utilize the self-attention mechanism to align the input text and regions of the image. In this way, BERT is augmented by the ability to handle visual embeddings F. Every visual embedding $f \in F$ represents a region of the image and is computed by summing three embeddings:

- f_o is a representation of visual features within the bounding box that corresponds to the region of f.
- f_s is a segment embedding. In BERT, segment embeddings are used to indicate which specific segment of a text a token belongs to. Here, it indicates whether the embedding is an image embedding or a text embedding.
- f_p is a position embedding. In BERT, it is used to indicate the position of a token within a sentence. Here, it is utilized if alignments between words and bounding boxes are provided.

Subsequently, the visual embeddings are combined with the initial text embeddings and fed into the multi-layer transformer. This enables the model to learn

alignments between the two input modalities and to calculate their joint representation.

The original paper [DCLT19] concludes that the attention mechanism utilized in VisualBERT can capture information in a way that is interpretable for humans. This thesis focuses on handling external knowledge in the form of KGs, but it could be a promising research direction to deliver interpretable results from the implicit knowledge source as well and to compare it with the interpretation of the external source.

3.5 Representation Learning on Graph Data

A graph represents entities and the relationships between them. While the nodes of a graph correspond to the entities, the relationships are represented by the edges connecting the nodes. A graph can be defined as follows:

Definition 6 (based on [SSYR23]):

$G = (V, E)$, where G is the graph. V is the set of n nodes (or: vertices) that are connected by edges in the set E. If a node $v \in V$ and a node $u \in V$ are connected by an edge, then $e_{v,u} \in E$.

A graph can be **directed** or **undirected**. In the directed case, the edges of the graph are associated with a specific direction, which means that $e_{v,u}$ does not imply the existence of $e_{u,v}$. If the graph is undirected, no direction is associated with the edges and thus $e_{v,u} = e_{u,v}$ for all edges [SSYR23].

The **neighborhood** $\mathcal{N}(v)$ of a node $v \in V$ is the subset of all nodes in V with an edge connection to v. [SSYR23]. A $k-$**hop neighborhood** is a set of nodes, where the shortest distance from v is not greater than k (based on [WCPZ22]).

A graph is **heterogeneous**, if the graph consists of different types of nodes and/or edges. It is **homogeneous**, if it is not.

Graphs are central parts of many VQA approaches, especially those that are built with GNN-based submodules. There are different types of graphs that can be constructed and utilized for this task. For example and as introduced in the related work section, in [ZYW+20] a Visual Graph based on visual features of the image, a Fact Graph based on excerpts from Knowledge Bases, and a Semantic Graph based on the textual labels of detected objects and their attributes are constructed. The term "Semantic Graph" is often used interchangeably with the term "Scene Graph". Within this thesis, the argument from [SSYR23] will be followed to use the term "Semantic Graph", because "Scene Graph" could be understood as a generalized version of different graph types representing information about an image, so the term "Semantic Graph" is more precise.

3.5.1 Constructing Knowledge Graphs

Knowledge Graphs (KGs) are a type of structured knowledge representation, where knowledge is organized as a graph.

KGs can typically be seen as a set of knowledge triples $KG = \{(h, r, t)\}$ (based on [WMWG17]), where a knowledge triplet $(h, r, t) \in \mathcal{F}$ consists of a head entity, a relation, and a tail entity, such as (Cat, needs, Food), and $\mathcal{F}$ is a collection of facts. A common approach is to represent the entities and the relations as natural numbers corresponding to preprocessed node labels, so $h, r, t \in \mathbb{N}_0$. In this way h, r, t can correspond to indexed values in arrays representing nodes and edges.

The overall construction of KGs within the VQA task (like within the [MCP+21]) approach can be divided into two parts:

1. Constructing a KG based on excerpts from existing Knowledge Graphs during data preprocessing steps.

2. Adding one more dimension representing information about the node.

After these two steps, the constructed KG serves as the input data of a GNN-based submodule during training. The training is based on the overall structure that remains within the input as well as within the output.

The (1) part focuses on the preprocessing of data from existing KGs such as ConceptNet [SCH18]. Since these KGs are usually too large to be processed on common hardware (e.g., ConceptNet contains around 8 million nodes and around 21 million edges) and also contain a lot of information that is not relevant to questions within a specific dataset, strategies to prune the KGs are introduced. An overview of widely used and publicly available KGs can be found in [LS23] The specific pruning strategy within KRISP is discussed in the experiments section.

The (2) part is where the actual training takes place, or where the input data is processed into learnable features. There are many possible ways of constructing the input data. Within this thesis, two approaches are considered as part of the proposed updates to the baseline model: By constructing a multi-modal KG and by constructing a modal-specific KG.

Multi-Modal Knowledge Graphs

The utilization of multi-modal KGs (MMKGs) is generally on the rise within NN-based fields, not only within VQA [ZLW+22]. The term "multi-modality" in this context usually refers to a combination of the modalities text and image (although other modalities like sound could also be utilized).

In [ZLW+22], MMKGS are divided into two general approaches:

- MMKGs that have nodes in one modality (usually text), but can have information from other modalities stored within their attributes.

- MMKGs with nodes representing entities from different modalities.

The construction of MMKGs aims at adding visual information to the symbolic knowledge found in traditional KGs, for example by creating an association between a node (or a relation between nodes) and a representation of the concept that the node represents. Accordingly, the construction of an MMKG is provided by the extraction of visual entities or concepts on the one hand and on the extraction of visual relations on the other. With the goal of symbol grounding, meaning the putting into relation of symbolic concepts and their proper multi-modal representation. [ZLW+22]. While this can be achieved, for example, by retrieving images from publicly available sources, it can also be the task of a learning algorithm to learn a multi-modal representation, like [DYL+22]

Modal-specific Knowledge Graphs

One goal within this thesis is to compare the accuracy and interpretability of two different proposed KG construction methods. Therefore, the modal-specific construction of KGs is proposed. It is inspired by approaches like [ZYW+20] and [JM23] and can be understood as dividing the node features into two modalities. This results in two KGs as input data that share the same structure but whose node features are comprised with a focus of textual features and visual features, respectively. This approach is described in detail within the experiments section.

3.5.2 Introduction to Graph Neural Networks

Graph Neural Networks were already applied within the VQA task before the OK-VQA subfield emerged [WWW+22]. The necessity arose after it was found that methods based on Convolutional Neural Networks (CNNs) were not enough to model the relations within an image for the VQA task. On the other hand, the application of Graph Neural Networks (GNNs) in visual-language image understanding tasks like Image Captioning, Image Retrieval or Visual Question Answering is being progressively researched in the last couple of years [CWD+22], [SSYR23]. GNNs enable learning algorithms based on NNs to learn with non-Euclidian data. So, one of the main motivations behind this architecture is the combination of NN-based approaches with the utilization of symbolic knowledge that can be provided with a knowledge graph.

GNN-based architectures can generally be divided into [WCPZ22]:

- Spatial approaches, which consider the structure of a graph by collecting the information around each node, often representing this information with an adjacency matrix.
- Spectral approaches, which consider graph structures within the spectral domain.

In the following, we will focus on spatial approaches.

Another delimitation of the subject of the following is the type of input data with which we will deal. Recently, GNNs were applied on processing image data within several Computer Vision tasks, with promising results [WCPZ22]. Here, image data is interpreted as graph data, for example, by viewing every pixel as a node that has a fixed number of neighbors and edges, resulting in a grid. A combination of these approaches is left for future work. The following considers graph data as input that does not have a fixed structure.

The two architectures that are considered are Graph Convolutional Networks (GCNs) and Graph Attention NeTworks (GATs), which are both utilized in the baseline model and its proposed updates within this thesis.

GCNs and GATs were already utilized within VQA prior to the OK-VQA task [WWW+22] to improve the ability to evaluate the relationships detected in an image to answer a given question. GCNs are commonly used within the OK-VQA task, such as in [MCP+21] or [ZYW+20]. Some current approaches, like [JM23], explore the application of GATs within the OK-VQA task.

3.5.3 Graph Convolution Networks

The GCN architecture was introduced in 2017 [KW16] and is based on the goal of applying the CNN architecture to graphs. Since its proposal, the GCN architecture became the most popular GNN architecture within the research area and is usually considered a baseline method to work with graph data [Lab23].

In principle, image data can also be seen as a graph, where every pixel in the grid can be considered a node. One key difference between Euclidean data like image data and graph data is the number of neighbors of nodes. In image data, the number of neighbors is fixed, for example, to the number of 8 neighbors if a pixel is in the middle of an image, or 3 if a pixel is the corner pixel of an image. The pixels within this grid are fully connected to their neighbors. This is different within the graph data, where the number of neighbors can vary significantly.

This challenges the idea of applying CNNs to graph data directly. To illustrate this problem (based on [Lab23]), we can consider a basic neural network layer that represents a linear transformation $h_i = x_i W^T$, where x_i is the input vector of a node i while W represents the weight matrix. In this case, no specific consideration of the input structure is taken.

A first solution could be to update this formula as follows ([Lab23]):

$$h_i = \sum_{j \in \mathcal{N}_i} x_j W^T$$

Where N_i is the set of neighbors of a node i. This way, the number of neighboring nodes is considered and can vary.

However, the problem with this solution is that the comparison between the resulting hidden units h_i is compromised, as the difference in the number of nodes

leads to potentially very different results.

This can be improved by adding a normalization coefficient, where each embedding representing the hidden unit h_i can be divided by the degree of the node i (based on [Lab23]):

$$h_i = \frac{1}{\deg(i)} \sum_{j \in \mathcal{N}_i} x_j W^T$$

As stated in the original GCN paper, this formula still needs another update to better balance the nodes with high degrees and the low ones (based on [KW16]):

$$h_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}} x_j W^T$$

In this way, nodes with fewer neighbors are assigned higher weights to balance out the distribution.

3.5.4 Graph Attention Networks

Although the normalization coefficients $\frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}}$ in GCN reflect the structure of node surroundings by considering node degrees, they remain static values during the learning process. In GAT, these coefficients become learnable parameters, called attention scores [Lab23]:

$$h_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} x_j$$

Where α_{ij} are the attention coefficients (or: the attentions scores) between two corresponding nodes i and j that reflect the learnable importance of the relation between the nodes. The attention scores are the results of a comparison of the inputs to each other, which itself is the mechanism of self-attention. This reminds of the functionality of the Transformer model, and the authors of GAT paper [VCC+18] explicitly cite the original Transformer paper [VSP+17] as an inspiration. In fact, it can be argued that transformers can be thought of as a special case of GNNs, where the input data is fully connected (where a sentence is understood as a set, where every word is connected with every other word) – as it was done in [Jos20] or [Vel23].

Apart from the attention scores, the architecture of GAT and its functionality are very similar to GCN. Despite possible redundancies and due to the importance of GAT for this project, GAT is described step by step below, based on the original GAT paper [VCC+18] and [Lab23].

The GAT Architecture

1. Inputs and Outputs

The inputs of a GAT layer are represented by $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}, x_i \in \mathbb{R}^F$ and contain the inital input data. The outputs $\mathbf{h} = \{h_1, h_2, \ldots, h_N\}, h_i \in \mathbb{R}^{F'}$. In both cases, N corresponds to the number of nodes, while F and F' are the number of features in each node of potentially different cardinality.

2. Adding trainable parameters

The goal of every Deep Learning model is to gain expressivity, which means the ability to appropriately approximate a function. To achieve this, we need learnable parameters or at least one learnable linear transformation that is represented by a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$ and applied to every node (based on [Lab23] and (based on [Lab23]).

3. Self-attention mechanism

In the following, the self-attention operation a is performed on the nodes with $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$.

$$e_{ij} = a\left(\mathbf{W}x_i, \mathbf{W}x_j\right) \tag{4}$$

The attention score as attributed to each edge e_{ij} between the given node features h_i and h_j.

During the actual training, the attentions score a_{ij} between a node i and one of its neighbors j, both feature vectors x from each node are needed for the computation. In the self-attention operation a in GAT, the two vectors $\mathbf{W}x_i$ and $\mathbf{W}x_j$ are concatenated. Also, an additional trainable weight matrix W_{att} is applied:

$$a_{ij} = W_{att}^T \left[\mathbf{W}x_i \| \mathbf{W}x_j\right] \tag{5}$$

Where W_{att} is the weight matrix that is trained to produce the attention coefficients a_{ij}.

4. LeakyReLU activation

As with other NN-based approaches, nonlinearity is added to enable the approximation of non-linear functions.

$$e_{ij} = \text{LeakyReLU}\left(a_{ij}\right) \tag{6}$$

In GAT, the LeakyReLU function is chosen over the ReLU function. The comparison between both is visualized in Fig. 5. ReLU functions suffer from the so-called dying ReLU problem, which occurs when some hidden unit in a layer always receive negative inputs, causing them to always output 0 [Vas19]. Since this behavior is not desired within GAT, LeakyReLU was applied.

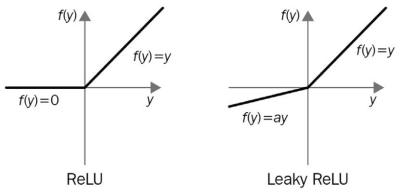

Figure 5: ReLU vs. LeakyReLU [Lab23].

5. Softmax normalization

Next, $softmax$ is applied to e_{ij} for enabling the comparison of attention scores between different nodes.

$$\alpha_{ij} = \text{softmax}\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)}$$

Which gives as the final attention scores α_{ij}.

The steps for computing α_{ij} can be summarized as follows (based on [Lab23], [VCC+18]):

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(W_{att}^{T}\left[\mathbf{W}x_i \| \mathbf{W}x_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(W_{att}^{T}\left[\mathbf{W}x_i \| \mathbf{W}x_j\right]\right)\right)} \tag{7}$$

6. GATv2 update

An updated version of GAT called GATv2 was introduced in 2021 [BAY22]. The paper identifies a problem with the way of how GAT computes attention scores. In GAT, the linear transformations are applied directly after each other. In this process, for every node h_i that queries a neighbor node h_j, the calculation of the

attention scores is not effected by the query node. GATv2 proposes a modification of the order of the operations of GAT to fix this; the overall functionality remains the same.

This is the modified order in GATv2:

$$\alpha_{ij} = \frac{\exp\left(W_{att}^{t}\,\text{LeakyReLU}\left([\mathbf{W}x_i \| \mathbf{W}x_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(W_{att}^{t}\,\text{LeakyReLU}\left([\mathbf{W}x_i \| \mathbf{W}x_j]\right)\right)} \tag{8}$$

In the original paper of GATv2 [BAY22], theoretical proofs and results of experiments on different benchmarks compare the performance of GAT and GATv2 with GATv2 consistently outperforming the original GAT. Hence, GATv2 will be utilized throughout this thesis.

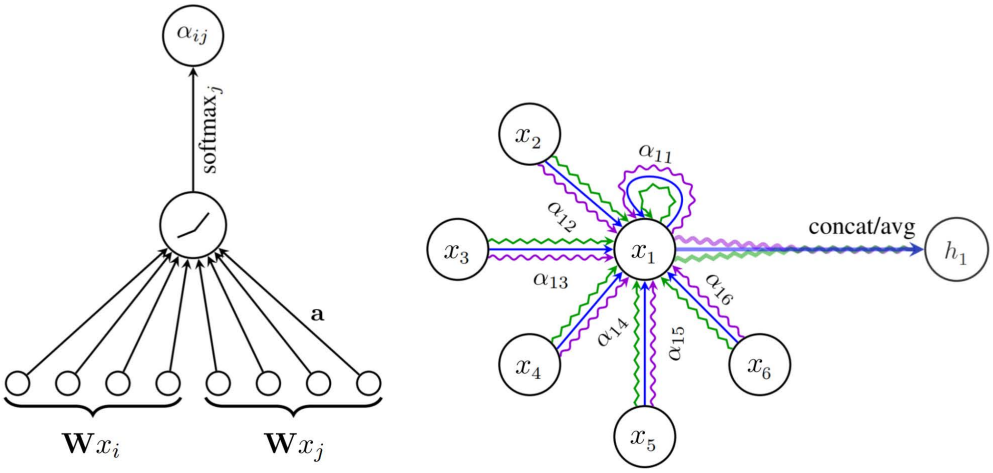

Figure 6: The attention mechanism in Graph Attention Networks (based on [VCC+18]).

7. Multi-head attention

In the GAT paper [VCC+18], two ways of integrating the result from the heads are discussed: Concatenating the resulting features or averaging them out. Concatenation is utilized within the hidden layers. This process is visualized in Fig. 6 and can be calculated as follows [Lab23]:

$$h_i = \Big\|_{k=1}^{n} h_i^k = \Big\|_{k=1}^{n} \sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k x_j \tag{9}$$

, where n is the number of attention heads and k is the index of the attention heads.

Since the last layer only has one head, concatenation is no longer sensible [VCC+18] and the final features are the averages of the results per head [Lab23]:

$$h_i = \frac{1}{n}\sum_{k=1}^{n} h_i^k = \frac{1}{n}\sum_{k=1}^{n}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k x_j \tag{10}$$

, where n is the number of attention heads and k is the index of the attention heads.

3.5.5 Interpretability of GNNs

GNNs have applications in a variety of fields, including critical applications such as drug discovery and drug repurposing, or safety and security applications. As GNNs are more and more implemented in tasks outside of research and thus have more and more impact on actual real-life decision-making, the need to understand the reasoning of GNNs surges.

Like other NN-based approaches, GNNs suffer from the difficulty of interpreting their results. Furthermore, trade-offs between the performance of a model and the increase of interpretability must be considered [DMW+23].

Currently, there is no consensus on a definition of interpretability or explainability within the literature, and both terms are sometimes used interchangeably. In fact, the search for a definition or at least the formulation of working definitions can be seen as an example of how humans can try to define non-mathematical concepts by considering how the concept is used by others, as described in the beginning of the motivation section of this thesis.

In [Lab23], the current discussions are summarized as a model that can be "interpretable" or "explainable". It is deemed "interpretable" if it is understandable to humans by design, like a decision tree (or, in our case, a knowledge graph). "Explainable", on the other hand, is attributed to models whose results can only be understood retroactively by applying explainability methods. In the case of the GNN application within the KRISP approach, the model could be understood as both: It takes a human-readable KG as an input that was augmented with more input data for each node, and returns a human-readable KG with the same structure but different node features that have to be explained retroactively.

Furthermore, in [Lab23] the differentiation between local and global explanations is discussed. While local explanations focus on the explanation of specific predictions of a model, global explanations focus on the model itself.

In [WCPZ22], "interpretation" and "explanation" are also distinguished by analyzing their usage within the literature. Here, "explanation" is summarized as a result of the collection of important features within a given prediction that can be used for a post-hoc interpretation of a result returned by a model. This corresponds to the positioning of local explanations given above. Interpretations, however, are simply referred to as a broader range of concepts that refer to models that are intrinsically interpretable.

Although the interpretability of a model is commonly referred to as a degree [WCPZ22], to the best of my knowledge, there is no current approach to actually specify this degree in numbers. While several research frameworks have been proposed for the evaluation of Graph Explainability methods (e.g., [AQLZ23]) and while these approaches propose different evaluation metrics, their goal is to quantify the difference between Graph Explainability methods, not the model predictions themselves.

For future work, it could be a fruitful research direction to establish a framework of Interpretability Measurement of GNN-based (or NN-based in general) model predictions; similar to the field of Inconsistency Measurement, where one goal is to overcome the binary view of "consistent" and "inconsistent" information by mapping inconsistency to numbers between 0 and ∞ [UTB20]. A starting point on the discussion why such a framework would have to face several challenges can be found in [CPC19]. Since a proposal of such a framework would be beyond the scope of this thesis, and since, for now, only a binary Interpretability Measure that represents whether a model prediction is interpretable, or not, would be applicable, interpretability is defined qualitatively:

Definition 7 (based on [WCPZ22]):

The Interpretability of a model is given when a human observer can interpret the cause of a decision. An interpretation is the mapping of an abstract concept into a domain that humans can understand.

Interpreting GAT

In [WCPZ22], the utilization of attention scores for interpretability is compared to post-hoc explanations of model predictions, while there is also a significant difference: The attention scores are a fundamental part of the overall model; they are not computed post-hoc for explanatory reasons. Their utilization can be seen as utilizing a byproduct for interpretability purposes, so in terms introduced above, it could be understood as a partially human-readable model, but only if some specific parts of the model are considered within additional post-hoc computations performed after the prediction.

There are several approaches of generating post-hoc explanations for GNN-based model predictions that are currently discussed in the literature ([AQLZ23], [WCPZ22]). Since this thesis focuses on interpretability by utilizing the attention-based and human-readable parts of the model, the exploration of additional explainability methods is left for future work.

It should also be noted that there is no consensus on the reliability of attention scores for the interpretability of NN-based models, in general. A short overview of pro- and contra-arguments can be found in [NZY21]. However, studies that argue

against this approach focus on NLP architectures that work with textual data.

When working with graph data, the attention scores α_{ij} calculated in Eq. 8 are computed between the feature vectors of the two corresponding nodes to indicate the importance of the edges between them. Thus, they can be visualized and interpreted, as is done in the case study section.

4 Methodology II: The Baseline Model and the Proposed Updates

4.1 The KRISP-GCN Baseline Model

4.1.1 An Overview of the KRISP Approach

The original KRISP [MCP+21] model presented a novel approach to combine two sources of external knowledge representations:

1. Implicit knowledge from the pre-trained and transformer-based language model BERT [DCLT19], which is used within the utilized VisualBERT [LYY+19b] architecture.
2. Symbolic knowledge from several publicly available knowledge bases, represented by a constructed KG, which is processed by a GNN.

Both approaches are implemented within submodules of the KRISP model and are handled sequentially. The process can be summarized as follows before being described in detail in this section:

1. A question/image pair is processed by a Faster R-CNN object detector and textual processors. This step is completed in advance of training for the entire dataset to ensure faster training.
2. In the first actual training step, the VisualBERT submodule receives a preprocessed question/image pair.
3. The output of the last layer of the VisualBERT submodule, $z^{implicit}$, is returned.
4. The prediction $y^{implicit}$ is calculated from $z^{implicit}$
5. $z^{implicit}$ is compressed from a vector of 768d to a vector of 128d and is passed to the GNN-based submodule together with the pre-processed question/image pair.
6. In the GNN-based submodule, all nodes that correspond to the question/image input data are activated based on preprocessed KG data.
7. The compressed $z^{implicit}$ vector is concatenated to every activated node. This is the first strategy to integrate internal and symbolic knowledge within the KRISP approach.
8. The updated KG is passed to a GNN-based submodule, which returns the output $z^{symbolic}$.

9. The prediction of the GNN-based submodule, $y^{symbolic}$, is calculated by multiplying $z^{implicit}$ and $z^{symbolic}$. This is the second strategy of integrating the two knowledge sources within KRISP, which is called "late fusion" in the KRISP paper [MCP+21].

10. The two vectors containing the predictions scores, $y^{implicit}$ and $y^{symbolic}$ are concatenated.

11. The final answer predictions correspond to the highest scores values within the concatenated prediction vector.

A visualization of the steps described in the summary can be seen in the visualized model overview in Fig. 7.

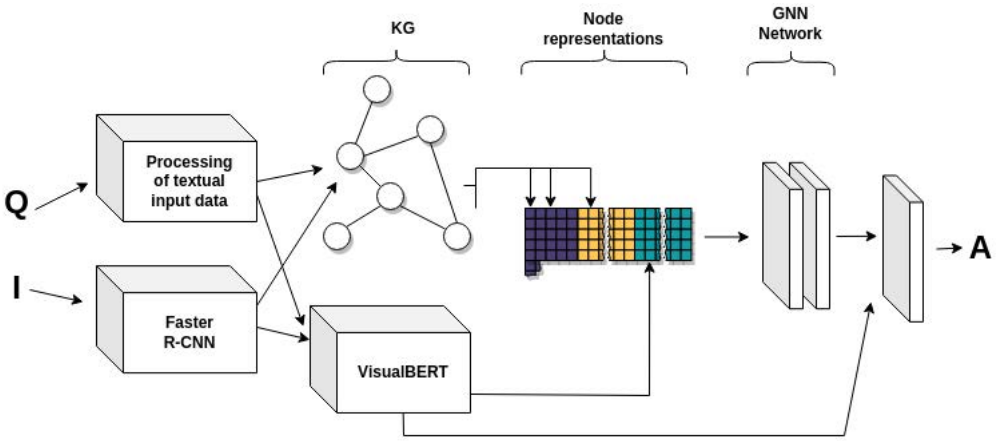

Figure 7: Overview over the general KRISP approach.

4.1.2 VisualBERT Submodule

The backbone of the implicit knowledge retrieval submodule is the VisualBERT model, which is based on a BERT model that has been pre-trained on large textual datasets.

As described in Section 3.4, VisualBERT augments the BERT architecture by incorporating visual features alongside the textual features as part of the model's input data, and thus enabling the model to work with two modalities. The visual features are gathered from utilizing an object detection model, which is a crucial part of most VQA approaches. Within the KRISP approach, the popular Faster R-CNN is utilized for detecting objects in an image. There are different object detectors available, but since exploring other approaches of object detection is out of scope of this paper, it is left for future work and Faster R-CNN will be also utilized within the following experiments as in KRISP.

For each detected object, the 2048d representation of the object is calculated within pre-processing steps and serves as part of the input data, together with features based on the bounding boxes of the detected objects which enable the embedding of visual regions associated with the detected objects, as well as the question encodings.

As in the original BERT implementation, the input question Q is tokenized utilizing the algorithm WordPiece, resulting in a sequence of $|Q|$ tokens. This sequence is embedded by utilizing pretrained BERT embeddings. Furthermore, the positional encoding of the tokens is appended to the output, resulting in the sequence $x_1^Q, \ldots, x_{|Q|}^Q$, which is the actual input of the transformer and is used for representation learning.

The output of all transformer steps is mean-pooled, resulting in the representation of the implicit knowledge z^{implicit}.

4.1.3 Graph Submodule

Constructing the Knowledge Graph

Alongside the feature representations of the detected objects, the actual prediction labels of the detected objects are also extracted from Faster R-CNN. These prediction labels are utilized within the KG pruning and also serve as a first step in transforming an image into a set of symbols.

For the collection of visual symbols, 4 image classifiers are utilized that are pretrained on the following datasets:

- ImageNet for information about objects
- Places365 for information about places
- LVIS for information on objects and parts of objects
- Visual Genome for visual relations between objects

Thus, a visual symbol within the KRISP approach can be understood as a vector with 4 elements that contains probabilities returned from 4 image classifiers that represent the probability of an object being in the given image. Furthermore, an additional classifier is added at the beginning of the vector, indicating if a concept is referenced in the question or not. This classifier is simply a binary value of 0 or 1 and is acquired during preprocessing steps.

The Knowledge Graph is constructed by extracting and processing excerpts from:

- ConceptNet [SCH18], a multilingual KG with a focus on commonsense facts.
- DBPedia [ABK+07] a multilingual KG with factual information that was extracted from Wikipedia and thus covers a variety of domains and concepts.

- VisualGenome [KZG+16] KG in the form of Semantic Graphs.
- hasPart KB [BRTC20] that focuses on part relationships between objects.

One first approach to combining these sources within one KG would be to simply combine all knowledge triples in a unifying manner. This approach would, however, be very limited, not only because of the problem of scale (it would result in a graph with millions of nodes and edges), but also because of the problem of noise (the most symbols within the KG would not have any relevance to the image/question pairs). Hence, the KRISP approach follows a pruning strategy of limiting the construction of the KG to elements that are more likely to be utilized in producing an answer prediction.

The KG is constructed within KRISP based on the following pre-processing steps:

1. Collect all symbolic concepts from the OK-VQA dataset based on the questions, the possible prediction labels from object detection algorithms (see above).
2. Include only edges in the construction of the KG that correspond to nodes that include the symbolic concepts.

The resulting KG has 7643 nodes and 35039 edges and is basically a graph consisting of the symbolic concepts extracted from the dataset and their 1-hop neighbors. The details of the selection of the excerpts and on the unification of the excerpts within one KG can be found in the supplementary material of [MCP+21]. The problem of incompleteness that influences this approach is discussed in the future work section. Other methods of constructing the KG could be explored but are left for future work since the focus of this thesis is on the latter part of the KG construction: The construction of node features.

The preprocessed KG described here was made publicly available by the authors of the KRISP paper and will be built upon within the experiments of this project. It serves as the backbone of the input of the GNN-based graph module within KRISP that is used to incorporate the knowledge from the KG.

Preparation of the GNN Input Data

Before passing the KG to a GNN, the representation of each node is built as a 433d vector with the following elements:

1. Element 0: A binary 0/1 value that indicates whether the concept appears in the question or not.
2. Elements 1-4: probabilities between 0 and 1 from each of the 4 classifiers indicating if a concept is in the image and how high the probability of its detection is.

3. Elements 5-304: A 300d GloVe representation of the concept label.
4. Elements 305-432: $z^{implicit}$ from the VisualBERT submodule, compressed to 128 elements.

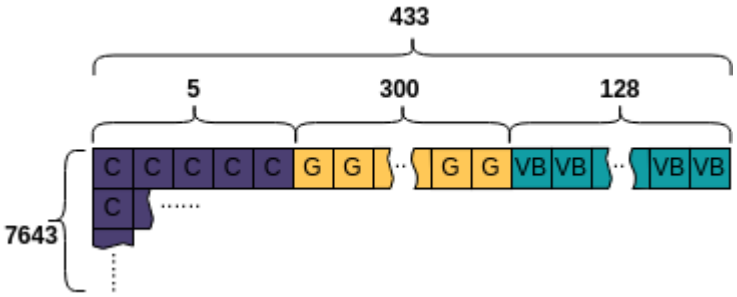

Figure 8: Node feature construction within the KRISP approach (resulting in a 7643 × 433 matrix)

During the pre-processing step, the KG representation is constructed with a vector of 305d which is added to each node, where the first five elements correspond to (1) and (2) and are zeroed out at the beginning. Additionally, the 300d GloVe representation (3) is added to every node. During the actual training, information about the nodes related to the specific question/image pair is gathered from preprocessed data, specifically information regarding (1) and (2). In this way, the nodes are "activated" to indicate the relevant nodes within the KG for further processing, while all other nodes are zeroed-out within (1) and (2) while the GloVe representation remains. Finally, (4) is concatenated as a version of $z^{implicit}$, which was compressed from 768d to 128d by passing it through a linear transformation layer and a subsequent ReLU activation for efficiency reasons. The resulting matrix is visualized in 8.

Predicting Answers

In summary, the architecture presented above describes two submodules that return the output $z^{implicit}$ from the VisualBERT submodule and the output $z^{symbolic}$ from the GNN-based submodule.

The implicit answer prediction $y^{implicit}$ is calculated in a final prediction layer by:

$$y^{implicit} = \sigma\left(W z^{implicit} + b\right) \tag{11}$$

, where σ is the sigmoid activation function. This layer is a simple linear projection of $z^{implicit}$ onto a vector of size vi, to predict an answer based on a set vocabulary of answers $V \in \mathbb{R}^{vi}$.

The amount of possible answers, vi, is determined by pre-processing steps. Within the KRISP approach, each candidate for an answer is first extracted from the OK-VQA dataset. Next, an element is added to the set of answers if it is present at least 10 times in the overall dataset. Following this method, the final set of possible answer candidates contains 2,250 elements. This also determines vg, the number of possible answer nodes in the constructed KG: If a collected answer also corresponds to a node in the KG, it is included in the graph answer set, which totals 1,746 elements.

As for the symbolic answer prediction, $z_i^{symbolic}$ represents the nodes of the input KG with the corresponding and updated node features, where each i corresponds to a node in the KG. The predictions $y_i^{symbolic}$ based on the answers $V \in \mathbb{R}^{vg}$ are calculated as follows:

$$y_i^{symbolic} = \sigma\left(\left(W^s z_i^{symbolic} + b^s\right)^T \left(W^z z^{implicit} + b^z\right)\right) \tag{12}$$

Here, the matrix representing the hidden state of $z^{symbolic}$ is multiplied by the vector representing the hidden state $z^{implicit}$, followed by a sigmoid activation, outputting normalized prediction values between 0 and 1. Finally, both prediction vectors $y^{implicit}$ and $y^{symbolic}$ are concatenated and the highest scored elements are chosen by index, which corresponds to the textual labels given in the sets of the two sets of answers. During the actual training, $y^{implicit}$ and ${}^{y_s}ymbolic$ are optimized separately by utilizing the binary cross entropy loss function.

4.1.4 Configurating the KRISP Architecture

KRISP-GCN, the model that was chosen as the baseline within this thesis, is a configuration of the general KRISP approach. There are several other configurations possible in regard to the architecture of the GNN-submodule. The architecture chosen for the final KRISP model in the original paper [MCP+21] was based on a Relational Graph Convolution Network (RGCN). Unlike GCN and GAT, the Relational Graph Convolution Network (RGCN) takes different edge types into account. Hence, RGCN can handle heterogeneous KGs as opposed to GCN and GAT, which omit the information on edge types and thus treat the constructed graph as a homogeneous graph. While a comparable counterpart to RGCN, the Relational Graph Attention Network (RGAT) exists, the GCN variant was chosen as the baseline, although the RGCN variant was the final architecture in the original paper and scored 1,73 % higher than the GCN architecture, which was also briefly evaluated in the supplemental material in the original paper.

The reason for this choice is the question of scope: One goal of this thesis is to improve a baseline model by utilizing an attention-based counterpart. So it is not necessary to choose the best performing configuration, since the same principle could be applied to a related update. As stated in the evaluation of RGAT in [BSCH19], the results indicate that RGAT also outperforms its counterpart (although marginally, as

also stated in the paper). However, this comes with higher hardware requirements, which were beyond the restraints of this thesis project. For future work, different attention-based types of GNN-architectures could be evaluated within this context, which also include other types than GAT and RGAT, such as the Heterogeneous Attention Network (HAN) architecture.

4.2 Proposed Updates to the KRISP-GCN Baseline Model

4.2.1 KRISP-GAT: Replacing GCN with GAT

The first update to the KRISP-GCN baseline model focuses on the utilized architecture within the GNN submodule: The GCN-based network is replaced with GAT. One reason for this update is the expectation that the performance of the model could improve, since GAT often performs better than GCN, as concluded in research papers like [PMS20]. The other reason for this choice is the attention mechanism within GAT which enables the model to return attention scores.

This update focuses on the GNN architecture, while the other two proposed updates focus on the construction of node representations.

4.2.2 KRISP-GAT-VF: Augmenting the MMKG With Visual Features

Within the KRISP approach, the vector representing each node's features has 5 classifier elements representing the visual content of the image, 300 elements representing the GloVe embedding of the textual label of the node, and 128 elements that represent the output of the VisualBERT module. Although the GloVe vector is the largest chunk of each node vector, the evaluations in the KRISP paper [MCP+21] indicate that its influence is rather small: Its ablation results in a drop in accuracy of only 0.36 %, as evaluated in the KRISP paper.

On the other hand, the representation of the visual content is represented by only five elements that do not contain any visual features, but probabilities regarding whether a concept is present in the image or in the question.

The Faster R-CNN object detector utilizes the region surrounding a detected object, marked by a bounding box around it. This cut-out of the image is further processed within the algorithm and is compressed to a 2048d feature vector in the process. This vector is the same vector f_o that represents a visual embedding and is utilized within the VisualBERT submodule, where it is further compressed to 768d.

As a second update and as visualized in 10, the augmentation of the input vectors of the nodes by this compressed vector is proposed within the KRISP-GAT-VF update. In this way, the initial node representations are enriched by the actual representation of a visual concept in the image.

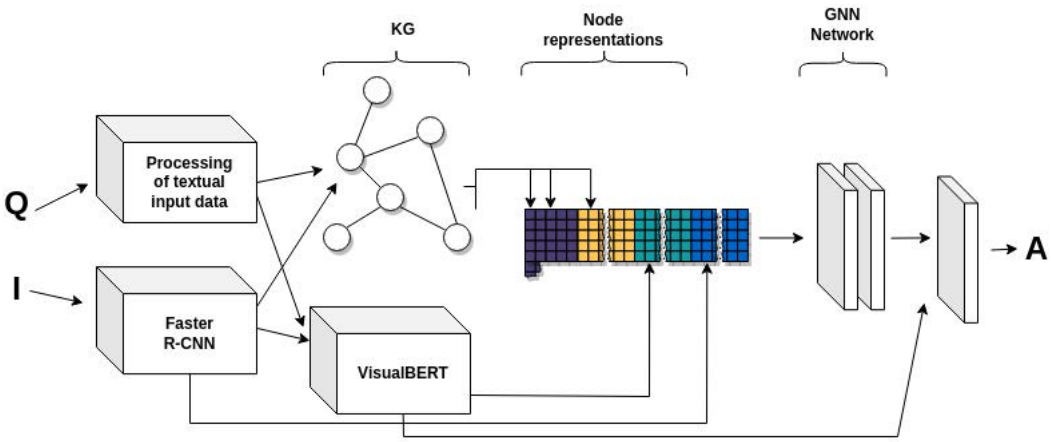

Figure 9: The KRISP-GAT-VF update: Augmenting node representations with visual features.

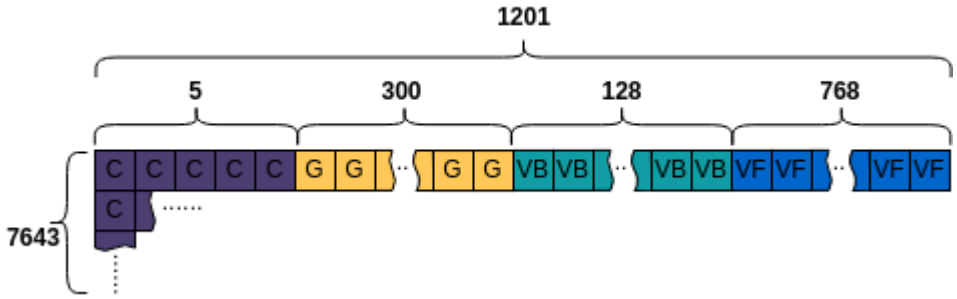

Figure 10: Node feature construction within the proposed KRISP-GAT-VF update (resulting in a 7643 × 1201 matrix)

4.2.3 KRISP-GAT-MS: Splitting the Augmented MMKG Into Two Modal-Specific KGs

Inspired by [ZYW+20] and [JM23], the augmented MMKG from the proposed KRISP-GAT-VF update is split into two modal-specific KGs. While the model architecture remains unchanged from the first update, it is augmented by an additional GNN-based submodule as seen in Fig. 11. The input data is now processed by three separate submodules:

1. The VisualBERT submodule, which remains unchanged.
2. The GNN-submodule for textual information, which constructs the same GloVe-based node features as in KRISP-GCN.
3. The GNN-submodule for visual information, which removes the GloVe features and adds the region-based visual features from the proposed KRISP-

GAT-VF update to each node

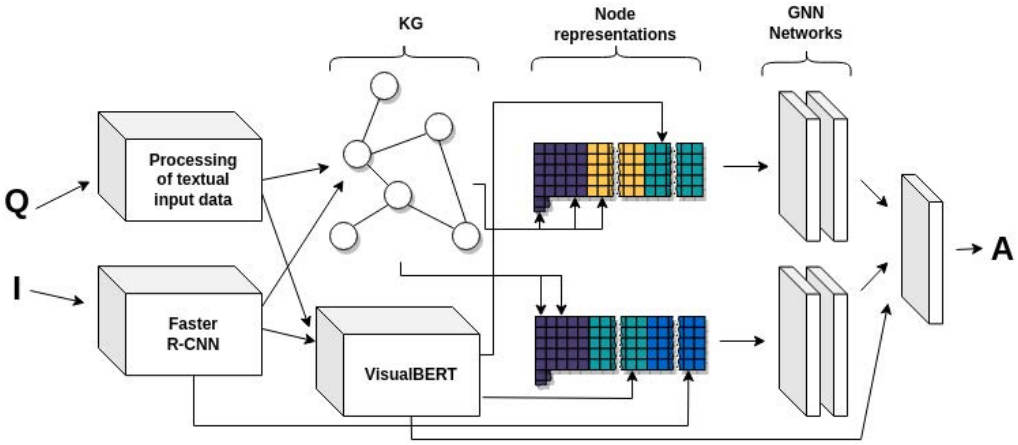

Figure 11: The KRISP-GAT-MS update: Splitting the augmented MMKG into two modal-specific KGs and handling them in separate submodules.

4.3 Evaluation Strategy

4.3.1 Quantitative Evaluation and Metrics

The general strategy for evaluating the baseline model and its updates utilizes ablation studies. Ablation studies are a common method of evaluation within the research of NN-based approaches and focus on considering the causality of a model, which usually consists of several components. Ablation studies follow the idea that if a specific component is removed or replaced, the behavior of the model changes; normally expected for the worse. After the change, the overall model can be evaluated (for example on a benchmark dataset such as OK-VQA), and the results of different ablations can be compared between each other. This strategy is also applied within the evaluation of this thesis, since the goal here is already centered on working on and replacing specific parts of a baseline model. Therefore, the baseline model and the proposed updates are evaluated after different ablations on OK-VQA by measuring the overall accuracy on this benchmark. The actual ablations are described in the "Experiments or Quantitative Evaluation" section.

Accuracy on OK-VQA

A way of calculating accuracy within the VQA task was introduced within the family of VQA datasets. It became a standard within VQA evaluations in general, and is also suggested by the OK-VQA paper [MRFM19]. In the VQA

dataset, every question/image pair is accompanied by 10 answer annotations provided by humans, while in OK-VQA only 5 are given. To make this metric applicable to OK-VQA, every annotation is used twice within the dataset, resulting in 10 possible answers per question/image pair. The metric is calculated as follows (based on [GKS+17]):

$$\text{Accuracy}(ans) = \min\left(\frac{human_{ans} \cdot ans}{3}, 1\right) \tag{13}$$

, where $human_{ans} \in \mathbf{N}_0$ is the amount of humans who provided the same answer within the annotation, as the given ans.

4.3.2 Case Study: Qualitative Evaluation of Examples

As discussed in Section 3.4, to the best of my knowledge, there is currently no model-based framework to evaluate the enhancement of the interpretability of a model in a quantitative manner. Therefore, a case study of some qualitative prediction examples is conducted within this thesis to explore how and if the interpretability of the proposed updates to the baseline model was enhanced.

One subgoal of this thesis project was to build a demo application that returns information about the calculation of a prediction of a model, such as prediction probabilities and attention scores. This application is utilized within the case studies. Additionally, the objective of the case studies is to suggest what could be done with the returned information in terms of quantitative evaluations when conducted systematically and how the returned attention scores could actually be interpreted.

5 Experiments for Quantitative Evaluation

5.1 Experimental Setup and Training Details

In the following, the hyperparameters for the experimental training runs are documented, which are partially influenced by or taken over from the original KRISP paper [MCP+21]. For example, Table 1 collects the hyperparameters for the VisualBERT submodule, which was not updated within this thesis. Therefore, the hyperparameters remain unchanged and are included here for reference.

Parameter	Value
Hidden Size	768
Visual Embedding Dim	2048
Num Hidden	12
Num Attention Heads	12
Hidden Dropout Prob	0.1
Transfer function	ReLU
BERT model name	bert-base-uncased

Table 1: Hyperparameters of VisualBERT submodule [MCP+21]

The Table 2 collects the hyperparameters for the GNN-based submodules. The hidden size of the nodes refers to the size of the output vector that represents each node after being processed by a GNN submodule, which therefore is represented by a matrix of size $b \times 7643 \times 128$, where b is the batch size and 7643 is the number of nodes in the utilized KG. The last two entries document the compression dimensions of the output of the VisualBERT module, which is compressed from 768d to 128d, and the compression of the visual features extracted by Faster R-CNN from 2048d to 128d.

Parameter	Value
Node Hidden Size	128
GNN Layers	2
GNN Type	GCN / GAT
GAT attention heads	8
Activation function	ReLU
VisualBERT input compress dim	128
Visual Features compress dim	768

Table 2: Hyperparameters for the GNN-based submodules (built upon [MCP+21])

Table 3 shows the general training parameters. A batch size of 20 was chosen to ensure better comparison, since different architecture that were utilized have also different hardware requirements. For example, GAT requires significantly more memory (one of the reasons being the utilization of 8 attention heads, so all compu-

tations take place 8 times in parallel). For this reason, the batch size was determined by the model with the highest requirements and was used for all other architectures as well. The warm-up steps refer to the incrementation of the learning over a certain number of steps. This is done to weaken the influence of the first epochs, since in the beginning of each training run the weights of the model are initialized with random values. Therefore, the first training steps usually should have a lesser influence, until actual representation patterns start to emerge.

Parameter	Value
Optimizer	Adam
Batch Size	20
Learning Rate	$5e-5$
Epsilon	$1e-8$
Warmup Steps	1500
Epochs	100

Table 3: Training Hyperparameters

The KRISP model and its implementation were published within the MMF framework by Facebook[3]. For the implementation of this thesis, some parts of the preprocessed data like the utilized KG, as well as some parts of the original code, were extracted to be built upon. The training of each configuration was conducted on machines with an NVIDIA A6000 48 GB GPU to ensure the comparison of the models. Since there are several randomized factors [Vas19] that could influence the outcome during the training of a model (such as the random batching of multiple samples per training step), each configuration was trained in three separate runs.

5.2 Presentation and Discussion of Results

5.2.1 Comparison of Accuracy on the OK-VQA Benchmark

Table 4 displays the results of the baseline evaluation and the evaluation of its updates, where the label KRISP-GAT corresponds to the results of the first update, the label KRISP-GAT-VF to the second, and KRISP-GAT-MS to the third update. For a better comparison, the two latter updates were also trained with the GCN architecture from the baseline.

The results indicate that the goal of maintaining the level of accuracy while increasing the interpretability has been reached in regard to the first update. Here, switching out the GCN architecture with an architecture that includes attention scores did lead to an increase in accuracy. However, the results also indicate, that both proposed updates on the KG construction did not improve the results. Considering the third update, it can be said that utilizing two GNN submodules with modal-specific data as was done within this thesis does not improve the results in

[3]https://github.com/facebookresearch/mmf.git, accessed: 07.08.2023

Model	Run 1	Run 2	Run 3	mean	std
KRISP-GCN	0.3095	**0.3159**	0.3126	0.3127	0.0032
KRISP-GCN-VF**	**0.3272**	0.3189	0.3214	0.3225	0.0043
KRISP-GCN-MS**	0.2950	**0.3041**	0.3003	0.2998	0.0046
KRISP-GAT*	0.3167	**0.3222**	0.3205	0.3198	0.0028
KRISP-GAT-VF*	0.3097	**0.3132**	0.3112	0.3114	0.0018
KRISP-GAT-MS*	0.3146	**0.3142**	0.3111	0.3133	0.0019

Table 4: Level of Accuracy on the OK-VQA benchmark, where * indicates updates proposed in this thesis that are also evaluated in the ablation study. ** indicates the proposed updates in combination with a GCN architecture that were included for comparison.

any case. KRISP-GAT-MS has a worse accuracy than KRISP-GAT from the first update, and the update also performs worse when combined with the GCN architecture. The results can be interpreted as showing that the update is not fruitful in general, since it follows the behavior of GCN performing worse than GAT in this regard. However, this cannot be indicated for the second update. In fact, the KRISP-GCN-VF model turns out to be the best scoring model that was built with an update proposed in this thesis, which also performs worse when utilizing the GAT architecture. This unexpected result will be further investigated within the case studies.

5.2.2 Ablation Study

The Table 5 displays the results of the ablation study. Experiments 1-10 consider different construction strategies of input data, and Experiments 11-13 consider the ablation of complete submodules.

Experiment 1 is equivalent to KRISP-GAT-VF from Table 4, while Experiment 2 is equivalent to KRISP-GAT from Table 4 and KRISP-GAT-MS without the visual graph.

When comparing the results of the ablation study, it is noticeable that omitting the VisualBERT submodule leads to a large performance drop (Experiments 12 and 13). On the other hand, the VisualBERT submodule reaches an accuracy score of 0.2836 when utilized by itself without a single GNN-based submodule (Experiment 11). Including GNN-based submodules leads to a further increase in accuracy of up to 0,0386 points, but the ablation study can be understood as an indication that with regard to performance, the most important part of the overall architecture is the VisualBERT submodule.

It can also be stated that the selection of utilized features seems to actually matter, and it can not be concluded that the accuracy simply gets better, the more elements the vector representation is composed of: Including visual features as in Experiment 1 seems to make the performance worse, whereas the selection in Experiment 2 leads

Experiment	Components	Accuracy
1	CLASS+GloVE+VBERT+VF	0.3132
2	CLASS+GloVE+VBERT	0.3222
3	CLASS+GloVE+VF	0.3090
4	CLASS+GloVe	0.3192
5	CLASS+VBERT	0.3125
6	CLASS+VF	0.2955
7	CLASS	0.3028
8	VF	0.2857
9	VBERT	0.2982
10	GloVe	0.2989
11	VBERT, w/o GNN submodules	0.2836
12	CLASS+GloVE+VF, w/o VisualBERT submodule	0.1107
13	CLASS, w/o VisualBERT submodule	0.0826

Table 5: Result of the ablation study, where **CLASS** are the 5 classification values, **GloVe** refers to the textual embedding, **VBERT** to the VisualBERT output and **VF** to the visual features augmentation.

to the best results. This is also indicated by Experiments 7, 8, 9 and 10. The shortest representation is evaluated in Experiment 7 where only a short vector with the five classifiers is utilized to represent a node. Perhaps surprisingly, this leads to the best result within the ablation studies that consider the separate parts of node features, although the other four configurations consist of (in part significantly) longer representations.

6 Experiments for Qualitative Evaluation

6.1 Case Studies: Qualitative Analysis of Examples

The focus within these case studies is on an interpretation of how the KRISP-GAT and the KRISP-GAT-VF updates, which are proposed within this thesis, differ. The KRISP-GAT-MS is not further investigated, since it did not achieve the objective of holding the accuracy while increasing the interpretability.

In the context of this thesis, a demo application was built that selects a random example from the OK-VQA dataset and returns predictions regarding the answer, as well as attention scores. These scores are utilized within the demo application as follows:

1. Collect the attention scores a_{ij} from Eq. 8 from the model during an inference step.
2. Calculate line width $line_{width}$ with $line_{width} = s \cdot a_{ij}$, where s is the scaling factor. s is merely an enhancing of visualization, since the attention scores are normalized values between 0 and 1.
3. Pass the computed $line_{width}$ values to the graph visualization routing that utilizes the library Pyvis [4].

Furthermore, the attention scores a_{ij} are used to compare path results of different models:

1. For every activated node, the shortest path between this node and the target node is computed utilizing the Dijkstra algorithm (provided within the NetworkX[5] package).
2. Since the shortest paths are the same, but the attention scores differ depending on the utilized model, they can be compared, e.g. by summing the scores along the paths.

The following case studies are performed by considering results from this demo application.

6.1.1 Case 1: Right Answer in GAT and GCN

The utilization of GAT was motivated by the ability of utilizing attention scores for the interpretability of a model. One common approach that is often mentioned in the literature is the visualization of the attention scores. As a first case study, we will consider the example that was introduced in the Section 2.3 and can be seen in Fig. 12

[4] Pyvis, Source: https://github.com/WestHealth/pyvis
[5] https://networkx.org/, accessed: 04.08.2023

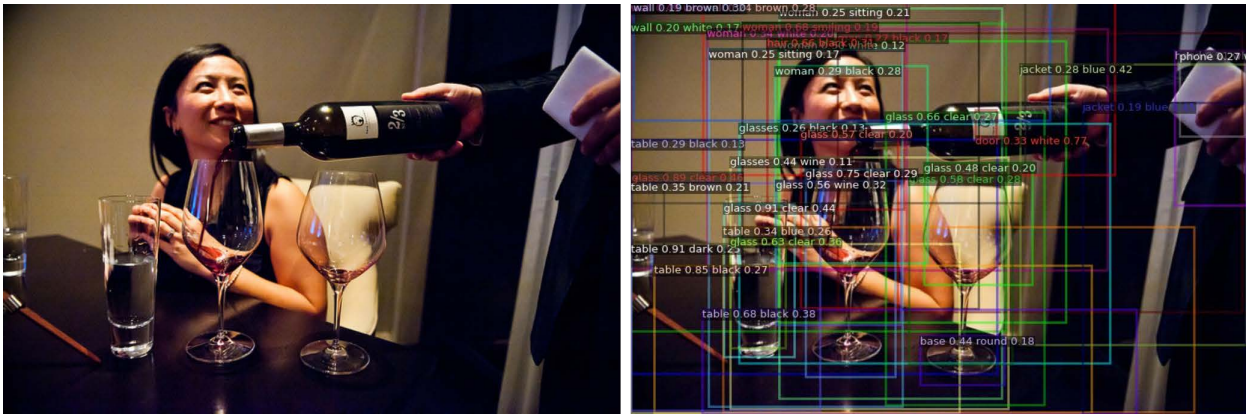

Figure 12: Question ID 5635255: What fruit is this beverage made of?

	1	2	3	4	5
Overall GCN	grape, 0.9999	grape, 0.9992	purple, 0.0049	green, 0.0034	apple, 0.0030
Overall GAT	grape, 0.9168	grape, 0.2772	wine, 0.2051	bottle, 0.1187	italy, 0.1165
VisualBERT in GCN	grape, 0.9993	wine, 0.0024	vitamin c, 0.0.0019	navy, 00.0016	broccoli,0.0010
VisualBERT in GAT	grape, 0.9168	wine, 0.0207	glass, 0.0192	cherry, 0.0169	lime, 0.0113
GNN in GCN	grape, 0.9999	purple, 0.0049	green, 0.0034	apple, 0.0030	jean, 0.0025
GNN in GAT	grape, 0.2772	wine, 0.2051	bottle, 0.1187	italy, 0.1165	orange, 0.0429
Target	grape				

Table 6: Top 5 predictions returned by the model and its VisualBERT and GNN submodules.

In Table 6, we can see that the resulting predictions for both, the GCN and the GAT architecture, correspond to the correct target answer with high confidence scores. When considering the results of GCN, the prediction values are the only information that can be interpreted with regard to the computation of the results. For example, we can see that in this particular case, the confidence of the prediction is so high that the next scored prediction is only at 0.0049. This seems to indicate a difference to the calculation within the GAT architecture, where the other probable answers within the top 5 results are all above 0.1. Also, the VisualBERT representation within the node vectors seems to have a weaker influence on GAT than GCN, since the results from the GAT submodule are more present in the overall scores.

Although the information about the final predictions can already be somewhat interpreted in regard to how it came to be, this is all the information we have in the case of GCN. With GAT, attentions scores can be considered for further interpretability. The difference can be visualized as in Fig. 13: Since we are working with a KG, we can consider the final target node and its relations to its neighbors to interpret the result. With GCN, we could speculate that the most important relation for the result follows the intuition that we are looking for a knowledge triple with "wine" as the head and "grape" as the tail, but we do not know for sure. With GAT,

we can incorporate the attentions scores into the visualization of the edges and see that, in fact, "wine" is considered the most important neighbor by the GAT module.

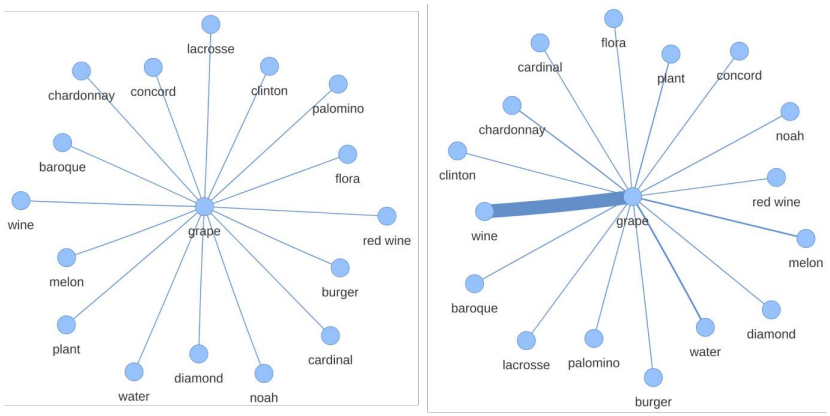

Figure 13: Visualizing GCN and GAT predictions.

Path	Attention Score
wine – grape	0.8012
water – grape	0.0496
melon – grape	0.0403
chardonnay – grape	0.0154
plant – grape	0.0147

Table 7: Attention scores for the top 5 neighbors of the target node.

While visualizing the attention scores of a node prediction can help to understand a model, this example also illustrates one shortcoming. In this case, the actual prediction can be clearly seen, while the distribution of the other scores seems to be somewhat uniform. As can be seen in Table 7, there actually is a gradation within the next values. This shortcoming can be problematic, especially in cases when the distribution is not so clear-cut towards the highest-scored value. For this reason, the next case studies will focus on utilizing the actual attention scores, instead of visualizing strategies.

6.1.2 Case 2: Right Answer in GAT, Wrong Answer in GAT-VF

In addition to visualizing the attention scores from a GAT model, there is also the possibility to work with the actual values returned by the model. This option is explored in the next case studies.

As described in the ablation study, it is somewhat surprising that, although GAT generally performs better than GCN, the performance drops if visual features are

	1	2	3	4	5
Overall GAT	microwave, 0.5168	microwave, 0.4029	dishwasher, 0.3550	oven, 0.3075	dryer, 0.2442
Overall GAT-VF	stove, 0.7556	oven, 0.5759	stove, 0.5620	oven, 0.3097	dishwasher, 0.0322
VisualBERT in GAT	microwave, 0.5168	dishwasher, 0.3550	oven, 0.3075	dryer, 0.2442	blender, 0.1609
VisualBERT in GAT-VF	oven, 0.5759	stove, 0.5620	dishwasher, 0.0322	dish, 0.0301	blender, 0.0066
GNN in GAT	microwave, 0.4029	convection, 0.2415	wireless, 0.2312	toaster, 0.1870	stove, 0.0901
GNN in GAT-VF	stove, 0.7556	oven, 0.3097	refrigerator, 0.0156	dishwasher, 0.0098	pan, 0.0084
Target	microwave				

Table 8: Top 5 predictions returned by the model and its VisualBERT and GNN submodules.

utilized as within the KRISP-GAT-VF update proposed in this thesis. This case study is an attempt to investigate whether and what attention scores can tell us about the computation of a prediction regarding the example in Fig. 14 that yields a correct result in KRISP-GAT, but a wrong one when utilizing visual features.

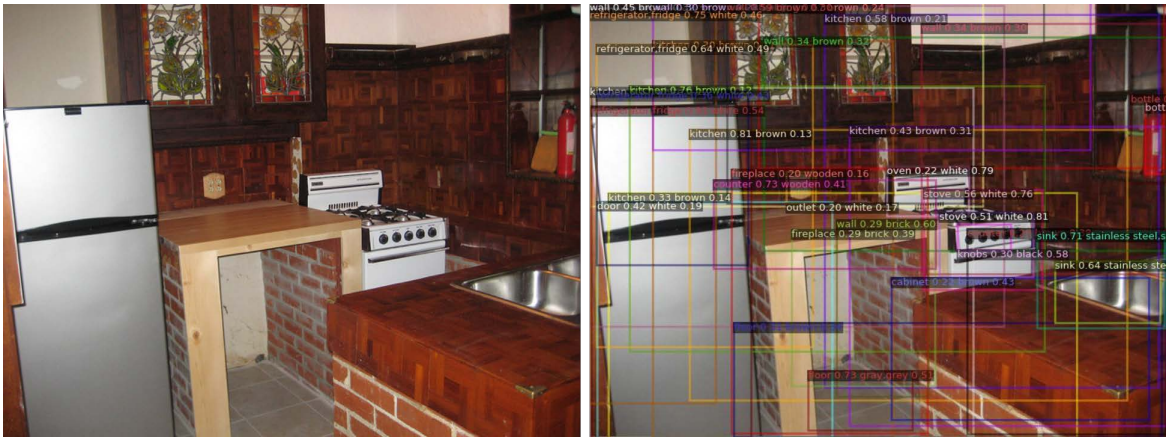

Figure 14: Question ID 4295985: What common appliance is missing from this kitchen?

Taking into account the results in Table 8, we can see that every submodule in the KRISP-GAT-VF variant yields a wrong prediction, while its counterpart consistently returns the right one. Furthermore, although "dishwasher" is also a possible target answer within the dataset, its prediction score is less than 0.05 in all cases, so it is not considered a predicted answer. This indicates that adding visual features leads to the failure of the model in terms of predicting the right answer.

As described in the introduction of this section, one way to compare the resulting attention scores could be to sum the scores over the paths between the activated nodes and the target. One first intuition could be that the sum of all of these sums should be larger in the GAT-only case. At least within this case, this intuition holds: Summing up all 33 paths (not only the top 12 from Table 9) result in 4.3256 for KRISP-GAT and 1.0339 for KRISP-GAT-VF.

GAT: Path	GAT: Sum	GAT-VF: Path	GAT-VF: Sum
knobs – stove – kitchen – microwave	0.5809	appliance – microwave	0.1532
stove – kitchen – microwave	0.5793	knobs – stove – kitchen – microwave	0.1439
wooden – house – kitchen – microwave	0.4565	stove – kitchen – microwave	0.1437
appliance – microwave	0.4355	small – mouse – kitchen – microwave	0.1153
bottle – counter – microwave	0.2993	wooden – house – kitchen – microwave	0.0741
wall – shelf – microwave	0.2684	bottle – counter – microwave	0.0726
small – mouse – kitchen – microwave	0.2232	cabinet – kitchen – microwave	0.0422
counter – microwave	0.2226	brown – eye – potato – microwave	0.0380
fire extinguisher – house – kitchen – microwave	0.2045	fire extinguisher – house – kitchen – microwave	0.0373
framed – book – kitchen – microwave	0.1579	refrigerator – kitchen – microwave	0.0359
sink – kitchen – microwave	0.0962	hinge – door – band – microwave	0.0355
stainless steel – iron – potato – microwave	0.0861	counter – microwave	0.0330

Table 9: Top 12 activated nodes out of 33 with corresponding aggregated sums of attention scores between an activated node and the target node, rounded up to four decimal places.

6.1.3 Case 3: Wrong Answers in GAT and GAT-VF

Inspired by the field of Inconsistency Measurement, the goal of the last case study is to explore the possibility of utilizing attention scores to quantify the wrongness of a prediction by investigating the example in Fig. 15.

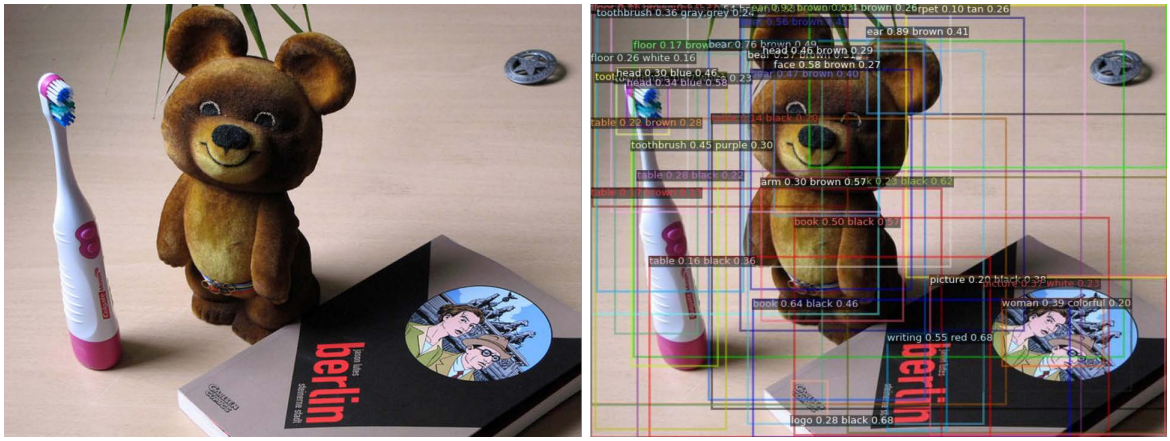

Figure 15: Question ID 1844855: Which country are these souvenirs from?

As seen in Table 10, both models that were proposed within this thesis yield wrong results. In both cases, the target "germany" is not only not predicted, it is also not present in the top 5 predictions (and actually also not in the top 15).

In both cases, the predicted result is wrong across all submodules. It is, however, notable that the top scoring predictions are countries. In fact, the predicted nodes and the target node are 1-hop neighbors of the node "country" in our KG. Even without utilizing attention scores, this opens up the strategy of counting the hops

	1	2	3	4	5
Overall GAT	usa, 0.3675	walmart, 0.2221	macintosh, 0.1849	williams, 0.1800	scotland, 0.1593
Overall GAT-VF	england, 0.8848	china, 0.8597	china, 0.67169	europe, 0.5159	england, 0.3935
VisualBERT in GAT	england, 0.105	disney, 0.0780	theodore roosevelt, 0.0764	japan, 0.0659	roosevelt, 0.0621
VisualBERT in GAT-VF	england, 0.8848	china, 0.6716	france, 0.2238	africa, 0.1817	india, 0.0525
GNN in GAT	usa, 0.3675	walmart, 0.2220	macintosh, 0.1849	williams, 0.1800	scotland, 0.1593
GNN in GAT-VF	china, 0.8598	europe, 0.5159	england, 0.3935	united kingdom, 0.3213	great britain, 0.313667
Target	germany				

Table 10: Top 5 predictions returned by the model and its VisualBERT and GNN submodules.

between the predictions and the targets based on the KG to get a quantification of how far away a prediction is from a target.

An additional strategy could be to also incorporate the attention scores in the paths between the activated nodes and the targets. In this case, the sum in KRISP-GAT results in 1.4063, while the sum in KRISP-GAT-VF results in 0.7445. This could also support the thesis that this sum can be seen as an indicator that a result is more wrong than a prediction by another model, where a higher scored value could be closer to the target prediction.

GAT: Path	GAT: Sum	GAT-VF: Path	GAT-VF: Sum
playroom – ball – street – germany	0.7595	book – city – germany	0.1565
book – city – germany	0.1288	playroom – ball – street – germany	0.1006
wooden – house – cat – germany	0.1096	bottle – grocery store – potato – germany	0.0901
ear – cat – germany	0.0814	teddy bear – nose – cat – germany	0.0846
teddy bear – nose – cat – germany	0.0502	teddy – hair – cat – germany	0.0633
country – germany	0.0476	wooden – house – cat – germany	0.0397
toothbrush – mouth – cat – germany	0.0377	toothbrush – mouth – cat – germany	0.0393
bottle – grocery store – potato – germany	0.0367	country – germany	0.0355
nose – cat – germany	0.0361	ear – cat – germany	0.0320
teddy – hair – cat – germany	0.0326	nose – cat – germany	0.0296
head – cat – germany	0.0312	brown – eye – potato – germany	0.0274
brown – eye – potato – germany	0.0296	head – cat – germany	0.0157

Table 11: Top 12 activated nodes out of 33 with corresponding aggregated sums of attention scores between an activated node and the target node, rounded up to four decimal places.

Both measures can be computed for all wrong answers on a dataset and thus could be utilized as another metric for the evaluation of a model, in addition to accuracy.

7 Conclusion

7.1 Summary and Discussion of Results

The results within the quantitative evaluation indicate that it is in fact possible to improve the accuracy and the interpretability of a model simultaneously by utilizing an attention-based GNN architecture, like in the case of replacing a GCN architecture with the GAT architecture. . The findings suggest that this is not a universal occurrence in all situations. This was especially surprising in regard to the second proposed update of this thesis, the augmentation of the constructed node features with visual features representing the detected object within an image. While this update did lead to a comparably large improvement when used together with a GCN architecture, it failed to do so when utilized together with GAT. The results also seem to indicate that splitting the utilized KG into modal-specific KGs does not improve the outcome; at least not in the way it was done within this thesis. Other strategies could be investigated and evaluated in future work. One strategy could be to not use both modal-specific submodules simultaneously, but to develop an algorithm to determine which modal-specific information might be better suited for different situations.

The second part of the quantitative evaluation focused on an ablation study. One takeaway from the study is an indicator for the general behavior of KRISP-based models: The VisualBERT submodule has the most influence on the accuracy, since without it the accuracy significantly drops when this submodule is ablated. However, while GNN-based submodules do not perform well by themselves, the results indicate that they improve an overall model when utilized alongside VisualBERT. Another takeaway from the ablation study is that it actually does seem to matter which features are utilized within node representations. Another outcome could have been that the accuracy simply gets better, if a vector representation consists of more elements and therefore has more learnable parameters.

The purpose of the case studies was to investigate how attention scores actually increase the interpretability of a model. One first approach was to visualize an answer node and its neighbors, while weighting the edges between these nodes according to the attention scores. This type of visualization seems to help to interpret how a prediction probability was calculated. When considering the same node structure with GCN, there might be an intuition of which neighbors might have been considered the most important ones for the result. With the GAT architecture, it seems to be possible to know for sure. However, the biggest takeaway from the case studies is that attention scores could make models comparable beyond prediction probabilities, and thus open up new ways of evaluating the models. This remains to be investigated systematically in future work, but some first pointers could be concluded from the case studies. The main idea here was that attention scores could be used to investigate how wrong a model is compared to a model with correct predictions; or compared with another model with wrong ones to evaluate which of the models actually could come closer to a correct result.

7.2 Future Work

The following directions towards possible future work can be derived from this thesis:

1. The dominant metric for benchmarks on the OK-VQA is accuracy. For future work that utilizes KGs, it could be considered to move away from the binary view of the wrong and right answers and toward a quantification of the wrongness of the answers, similar to the field of Inconsistency Measurement [UTB20]. This could be achieved, for example, by counting hops between the target and the predicted answer nodes, to quantify how far away an answer is. If additionally models based on the attention mechanism are utilized, the attention score across these paths could also be taken into account.

2. Establishing a framework for Interpretability Measurement of model predictions by GNN-based (or NN-based in general) architectures could help the research on the interpretability of GNN-based models. Currently, to the best of my knowledge, no such framework is available. Interpretability is already commonly referred to as a degree, so it seems fitting to actually quantify interpretability to compare it between proposed model architectures.

3. The enhanced interpretability of the baseline model was achieved by utilizing the GAT architecture within this thesis. This is only one approach to utilize an attention-based GNN architecture; other possibilities such as the RGAT or the HAN architecture also exist. For the exploration and comparison of those architectures, a framework for a quantified interpretability degree could be especially fruitful.

4. Since the first popular GNN approach proposed in [KW16], it was documented that GNNs tend to over-smooth after only two layers of processing the input data, so the node features become indistinguishable. This challenge persists to this day, but there are approaches to conquering this challenge that, for example, are currently discussed in [SSYR23]. So another pointer could be the exploration of newer GNN architectures within the OK-VQA task.

5. The interpretability approach of this thesis centers on the attention mechanism, which is also utilized in the transformer-based VisualBERT submodule that serves as the source of implicit knowledge. Some approaches to improve the interpretability based on attention within transformer-based models like [YCW+23] were proposed in the literature and could be further explored in future work. An additional challenge that would have to be addressed here is the handling of visual features instead of textual features only.

6. One direction to generally improve the accuracy of an approach that utilizes KGs is to focus on how this KG is constructed. In KRISP, the KG is built by extracting symbols from the dataset and collecting their direct neighbors. This

approach can be attributed to the necessity of pruning publicly available KGs for the purpose of cutting some irrelevant information and also to enable the processing of the resulting KGs on current GPUs. The resulting KG from the KRISP approach thus consists of 7643 nodes, 1746 of which are answer nodes. This poses a challenge, because even the VisualBERT submodule has more answers to work with, since here the answer vocabulary consists of 2250 answers that are collected from the dataset in case an answer is at least 10 times given in the overall collection. Approaches such as [JM23] address the incompleteness of knowledge sources by constructing graphs from additional sources, including language models. This direction of research could also be promising for OK-VQA approaches that especially consider interpretability, since constructed graphs based on information from language models are at least interpretable as an intermediate step.

7.3 Outlook

In general, the utilization of KGs within OK-VQA can be categorized as a promising approach, not only in regard to research on better performing models, but also in regard to models returning interpretable information on their calculation process. This opens up possibilities for further research on how these models work exactly in detail. Given a KG as a form of structured symbolic knowledge and a GNN architecture that returns attention scores, it can be accomplished to utilize both factors for interpreting the inner workings of a model.

The motivation section of this thesis began as an argument for the combination of CV-based and NLP-based approaches. While approaches that focus on transforming VQA problems to NLP-based problems deliver good results, the results of this thesis indicate that the utilization of visual features should not be completely neglected. Although the specific finding that KRISP-GCN performs better with the augmentation of visual features was more of a byproduct of this thesis, since the main focus was actually on utilizing this update together with an attention-based architecture.

Overall, this thesis underscores the promising research on interpreting attention scores in GNN-based approaches. Attention scores can be utilized not only in the interpretation of a correct prediction of a model, but perhaps also for the formulation of additional metrics besides accuracy. Especially within GNNs, this could lead to new measurements of how wrong a model actually is, since the attention scores can be compared alongside specific paths within the utilized graphs. This opens up new possibilities of comparing and interpreting the outcome of GNN based models.

References

[AAL+15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Ado73] Theodor W. Adorno. *Philosophische Terminologie. Zur Einleitung. Band 1.* Suhrkamp Verlag, Frankfurt am Main, 1973.

[AFJG16] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016.

[AQLZ23] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1):144, March 2023. Number: 1 Publisher: Nature Publishing Group.

[BAY22] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2022.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[BRTC20] Sumithra Bhakthavatsalam, Kyle Richardson, Niket Tandon, and Peter Clark. Do dogs have whiskers? a new knowledge base of haspart relations, 2020.

[BSCH19] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. Relational graph attention networks, 2019.

[CHC+22] Zhuo Chen, Yufeng Huang, Jiaoyan Chen, Yuxia Geng, Yin Fang, Jeff Pan, Ningyu Zhang, and Wen Zhang. LaKo: Knowledge-driven Visual Question Answering via Late Knowledge-to-Text Injection, November 2022. arXiv:2207.12888 [cs].

[CPC19] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8), 2019.

[CWD+22] Chaoqi Chen, Yushuang Wu, Qiyuan Dai, Hong-Yu Zhou, Mutian Xu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. A Survey on Graph Neural Networks and Graph Transformers in Computer Vision: A Task-Oriented Perspective, October 2022. arXiv:2209.13232 [cs].

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[DMW+23] Lauren Nicole DeLong, Ramon Fernández Mir, Matthew Whyte, Zonglin Ji, and Jacques D. Fleuriot. Neurosymbolic AI for Reasoning on Graph Structures: A Survey, February 2023. arXiv:2302.07200 [cs, stat].

[DYL+22] Yang Ding, Jing Yu, Bang Liu, Yue Hu, Mingxin Cui, and Qi Wu. MuKEA: Multimodal Knowledge Extraction and Accumulation for Knowledge-based Visual Question Answering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5079–5088, New Orleans, LA, USA, June 2022. IEEE.

[EHOP22] Thomas Eiter, Nelson Higuera, Johannes Oetsch, and Michael Pritz. A Neuro-Symbolic ASP Pipeline for Visual Question Answering, May 2022. arXiv:2205.07548 [cs].

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, October 2014. arXiv:1311.2524 [cs].

[Gir15] Ross Girshick. Fast R-CNN. pages 1440–1448, 2015.

[GKS+17] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[GPT+22] Feng Gao, Qing Ping, Govind Thattai, Aishwarya Reganti, Ying Nian Wu, and Prem Natarajan. A thousand words are worth more than a picture: Natural language-centric outside-knowledge visual question answering, 2022.

[GWH+22] Liangke Gui, Borui Wang, Qiuyuan Huang, Alexander Hauptmann, Yonatan Bisk, and Jianfeng Gao. KAT: A knowledge augmented transformer for vision-and-language. In *Proceedings of the 2022 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 956–968, Seattle, United States, July 2022. Association for Computational Linguistics.

[JHvdM+16] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016.

[JKFF16] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[JM23] Lei Jiang and Zuqiang Meng. Knowledge-based visual question answering using multi-modal semantic graph. *Electronics*, 12(6), 2023.

[Jos20] Chaitanya Joshi. Transformers are graph neural networks. *The Gradient*, 2020.

[KW16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[KZG+16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations, 2016.

[Lab23] Maxime Labonne. *Hands-on graph neural networks using Python: practical techniques and architectures for building powerful graph and deep learning apps with PyTorch*. Packt Publishing Ltd., Birmingham, UK, 2023. OCLC: 1377285473.

[LS23] Maria Lymperaiou and Giorgos Stamou. A survey on knowledge-enhanced multimodal learning, February 2023. arXiv:2211.12328 [cs].

[LYY+19a] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *CoRR*, abs/1908.03557, 2019.

[LYY+19b] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language, 2019.

[MCP+21] Kenneth Marino, Xinlei Chen, Devi Parikh, Abhinav Gupta, and Marcus Rohrbach. KRISP: Integrating Implicit and Symbolic Knowledge

for Open-Domain Knowledge-Based VQA. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14106–14116, Nashville, TN, USA, June 2021. IEEE.

[MRFM19] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge. pages 3195–3204, 2019.

[NZY21] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[ON15] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, December 2015. arXiv:1511.08458 [cs].

[Ots22] Jun Otsuka. *Thinking About Statistics: The Philosophical Foundations*. Routledge, New York, 1 edition, November 2022.

[OWJ+22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[PMS20] Ankit Pal, Selvakumar Murugan, and Malaikannan Sankarasubbu. Multi-label text classification using attention-based graph neural network, 03 2020.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[SCH18] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge, 2018.

[SKC+22] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-OKVQA: A Benchmark for Visual Question Answering using World Knowledge, June 2022. arXiv:2206.01718 [cs].

[SSYR23] Henry Senior, Gregory Slabaugh, Shanxin Yuan, and Luca Rossi. Graph Neural Networks in Vision-Language Image Understanding: A Survey, March 2023. arXiv:2303.03761 [cs].

[SYWY23] Zhenwei Shao, Zhou Yu, Meng Wang, and Jun Yu. Prompting large language models with answer heuristics for knowledge-based visual question answering, 2023.

[UTB20] Markus Ulbricht, Matthias Thimm, and Gerhard Brewka. Handling and measuring inconsistency in non-monotonic logics. *Artificial Intelligence*, 286, 2020.

[Vas19] I. Vasilev. *Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch*. Packt Publishing, 2019.

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[Vel23] Petar Veličković. Everything is Connected: Graph Neural Networks, January 2023. arXiv:2301.08210 [cs, stat].

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs].

[WCPZ22] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Nature Singapore, Singapore, 2022.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, December 2017. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[WWS+17] Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony Dick. Fvqa: Fact-based visual question answering, 2017.

[WWW+22] Qi Wu, Peng Wang, Xin Wang, Xiaodong He, and Wenwu Zhu. *Visual Question Answering: From Theory to Application*. Advances in Computer Vision and Pattern Recognition. Springer Nature Singapore, Singapore, 2022.

[YCW+23] Catherine Yeh, Yida Chen, Aoyu Wu, Cynthia Chen, Fernanda Viégas, and Martin Wattenberg. Attentionviz: A global view of transformer attention, 2023.

[ZLW+22] Xiangru Zhu, Zhixu Li, Xiaodan Wang, Xueyao Jiang, Penglei Sun, Xuwu Wang, Yanghua Xiao, and Nicholas Jing Yuan. Multi-Modal Knowledge Graph Construction and Application: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2022. arXiv:2202.05786 [cs].

[ZYW+20] Zihao Zhu, Jing Yu, Yujing Wang, Yajing Sun, Yue Hu, and Qi Wu. Mucko: Multi-Layer Cross-Modal Knowledge Reasoning for Fact-based Visual Question Answering, November 2020. arXiv:2006.09073 [cs].