High-level Understanding of Visual Content in Learning Materials through Graph Neural Networks

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Monica Laura Zündorf

geborene Haurilet

aus Temeschburg, Rumänien

Tag der mündlichen Prüfung:	19. Oktober 2020
Hauptreferent:	Prof. Dr.-Ing. habil. Rainer Stiefelhagen
Korreferent:	Prof. Dr.-Ing. habil. Jürgen Beyerer

Acknowledgements

First, I want to thank Professor Dr.-Ing. Rainer Stiefelhagen for his invaluable support and guidance throughout my PhD. I am very grateful for the wide range of opportunities that I was given at the lab: conducting research, writing proposals, designing lectures, supervising students, coordinating the practical course, and especially organizing our reading groups. I thank Professor Dr.-Ing. Jürgen Beyerer for agreeing to be a reviewer of this thesis and for giving valuable feedback.

I thank all my fellow co-workers including Adrian, Alina, Angela, Anke, Constantin, Corinna, Daniel, Kailun, Karin, Makarand, Manel, Manuel, Saquib, Sebastian, Simon, Tobias, Vanessa, and Ziad for the wonderful time at the lab. I especially want to thank Alina for all her help and for the exciting work we did together. The past few years would not have been the same without her. Special thanks also to Anke for the many discussions we had and for her help at the beginning of my PhD.

I want to thank my family and friends for their encouragement and being there for me when times were hard. Most importantly, I thank Tobias for his love and support over the past years. He always encouraged me in my research and in pursuing this academic degree. Without him, this work would not have been possible.

Abstract

Since the rise of deep learning, extraordinary progress has been made in fields revolving around *high-level reasoning and understanding* further narrowing the gap between the capabilities of humans and AI. At the heart of these fields lies the *question answering* task which aims at implementing models that answer questions based on a given knowledge base. Question answering has a wide variety of applications ranging from automatic indexing and use-cases in systems for assistive technologies to research laying the groundwork for human-like AI. Due to its importance in such applications, question answering was strongly addressed by the research community in the past several years leading to tremendous strides in both the structure of such models and in their accompanying datasets.

Popular approaches for question answering comprise neural networks which embed the textual data (question) and the given information (*e.g.*, image) into the same representation space and, then, based on their joint encoding an answer is generated. However, most of these methods handle only a *single* data type (*e.g.*, natural images) and have difficulties representing highly-structured data (*e.g.*, relations of objects in the scene). Moreover, the proposed models have a black-box structure, as one cannot infer their reasoning to generate the answer directly from the network.

In this thesis, we address these shortcomings and propose *human interpretable* approaches for the *question answering task*, where the *reasoning* of the networks can be inferred directly from the architecture. Moreover, our approaches are able to understand and reason on highly structured data from a knowledge base of *both* the textual and visual domain, which we analyze on data extracted from learning materials. We aim for networks that are capable of generalizing across multiple modality types and structures, where one would require several reasoning steps for inference. Learning materials are perfect candidates for such an analysis, as they offer rich and diverse content, different types of figures, and various interaction possibilities. Moreover, visual reasoning on learning materials has other important applications, *e.g.*, for automatically answering questions of students with visual impairments on lecture documents as well as for fast indexing of page data.

To enable the networks to reason on structured content, we leverage a *graph representation* of the given information for our neural networks. In case of images, the nodes represent object instances in the scene while the edges depict their relationships. In comparison, for text content, the nodes are sentence embeddings where the edges model their placement. We base our methods on graph neural networks, which inherently deal with highly structured input data and can give a natural representation of the knowledge base. In this thesis, we concentrate and explore two types of graph networks: a relation-centered approach based on edge pooling and a path-based method that traverses the visual input graph guided by the question.

Finally, we demonstrate the reasoning capabilities of our models in an analysis where we achieve state-of-the-art results on popular benchmarks for *visual question answering*, *multi-modal reasoning*, and *page question answering*. We evaluate different configurations of our architectures, input modalities, and data representation (*i.e.*, raw 3D visual tensors and graph representations). Our analysis shows that the models are able to provide *transparency* and *human interpretability*, *i.e.*, one can directly infer the attended knowledge and reasoning for answering each question.

Contents

1 Introduction

The ability to make effective decisions about events that occur in challenging surroundings is a key trait of human cognition. This remarkable property is a result of six million years of evolution, where humans gained the ability to make difficult decisions based on stimuli charged from the environment. We distinguish between three major steps in our thinking process: *recognition*, *reasoning*, and *action* [Parasuraman, 2000]. The *recognition* step entails the interpretation of stimuli received from the environment and their association to *semantic meaning*. While the stimuli can be visual, audio, or tactile, the semantic interpretation can range from a simple list of objects to a complex interaction between the recognized instances. In the second part of the thought process, we *reason* on the perceived semantic information and produce a possible action depending on the estimated state of the environment. The final part consists of executing the selected *action* based on the result of the reasoning process and, thus, a change in the state of the human or the surroundings occurs.

AI is a thriving field aiming at developing models that are able to *think* like humans addressing all the steps in the human thinking process: (1) recognition (*e.g.*, object recognition and classification), (2) visual and knowledge reasoning (*e.g.*, question answering), and (3) action (*e.g.*, visual navigation in robotics). In this

thesis, we focus predominantly on one of these tasks namely on *reasoning* on visual and textual input data. This task tackles the problem of selecting an action or generating a summarization either based on raw visual information or on the semantic interpretation in text form produced by the recognition step of the visual data.

A popular task in the field of visual and text-based reasoning is Question Answering (QA), which aims at answering questions based on some given input data. A key aspect of QA methods is their ability to be used in practice on a wide range of applications from robotics to assistive technologies. First, QA is a helpful tool for people with visual impairment that offers a human-friendly front-end for communication between the user and the computer system. Question answering is also used for indexing large scale data via the generated answers for each image based on a question of interest. QA offers an excellent interface for an easy communication between a user and an avatar or (house) robot. Moreover, QA is an extended example of visual dialog systems and the cornerstone of visual navigation approaches. Finally, QA is a generalization of object recognition, where different objects are recognized on request by the model.

When designing models for QA one has to decide on different properties the network should posses. In this work, we focus on: (1) high portion of *correctly* answered questions (*e.g.*, accuracy of the overall system spanning over different query types) and (2) *human interpretability*, *i.e.*, helping us to infer *why* an error occurred.

While there are several possible applications for QA, we restrict ourselves on answering questions on *learning materials*, especially focusing on graphical and textual content. Thereby, we analyze *how* to build models that possess the previously defined key-properties. We discuss the advantages and shortcomings of previous QA methods and show how to improve them for a better representation based on our data. For designing our neural architectures we are guided by these aimed properties and propose different graph neural networks that are able to *reason* on highly-structured visual and textual data. Next, we show the overall structure of this thesis and discuss the contributions of each chapter.

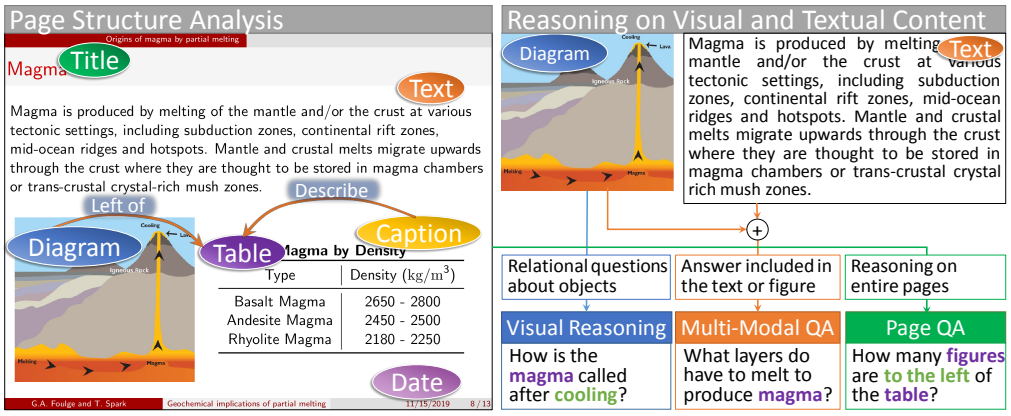


Figure 1.1: Example page of a learning material (left) with labels of the document components (*e.g.*, title, table) and their corresponding relationships (*e.g.*, describes). On the right, we show example questions for three input modality types.

1.1 Overview

In this thesis, we address the *QA task* and propose *human interpretable* approaches that are able to analyze highly structured data from *both* the textual and visual domain. Since our goal is to develop models that reason akin to humans, we employ a similar test-bed procedure through learning materials. Learning materials are good candidates for such a problem, as they offer rich and diverse content, different figure types, and various interaction possibilities (see Figure 1.1 for an example of a slide extracted from a learning material and the associated document components). Moreover, visual reasoning on learning materials has other important applications, *e.g.*, for automatically answering questions of students with visual impairments about lecture content and fast indexing of document data.

For the reasoning task across documents, we employ several techniques specialized on different types of page components. We distinguish between three groups of approaches dependent on their input data type: (1) **figure-based models**, (2) **multi-modal networks** for both text and image understanding, and (3) architectures that **reason on entire pages**. Since all these data modalities are highly structured, we propose novel machine learning methods based on the recently introduced *graph neural networks* that are able to handle relations between instances

in the visual and textual domain (*e.g.*, document components). We analyze the effectiveness of various types of graph neural networks (*i.e.*, relation-centric and path-based approaches) on our learning materials. Dependent on the data type (visual, multi-modal, or page data), we make use of different graph network types:
(1) For the **figure-based setting**, we introduce a novel path-based approach, which leverages a visual graph representation of the image (*i.e.*, the nodes parameterize the object instances in the scenes and the edges depict their relation). We demonstrate that our model is not only able to achieve *state-of-the-art results* on various popular benchmarks for figure-based QA, but also it is human-understandable as one can interpret the decisions made by the model for generating each answer.
(2) In case of **multi-modal data**, we present a new technique, that focuses on various parts of the image and text *simultaneously* dependent on the input query. Specifically, our method employs an edge-centric technique that is able to search on the *entire* structure of the text and figures by dividing the data into entities and combining them to a common edge representation. We show the strength of this model on a dataset specifically designed for textbook question answering where our model needs to understand both figures and textual content.
(3) We introduce the task of **visual reasoning on documents**, which aims at answering questions about an *entire page* covering queries about the document components and their interaction. To tackle this problem, we collect a dataset with images of pages and corresponding questions and present them in a public benchmark. On this dataset, we evaluate several models previously used for QA on natural images and show that they have difficulties on this new task. Thus, we propose a fully-convolutional architecture that is able to tackle the drawbacks of previous models by focusing on parts of the image based on the current question.

In a thorough evaluation, we show that our models achieve state-of-the-art results on popular benchmarks for *visual question answering*, *multi-modal reasoning*, and *page question answering*. Through a qualitative analysis, we show that our path-based and edge-centric models provide *transparency*, as one can directly extract the attended knowledge and reasoning for answering each question.

1.2 Contributions and outline

We analyze models for visual reasoning on different types of graphical data ranging from synthetically generated figures to images of entire pages. Thereby, we pre-process learning materials such as textbooks and slides. Then, we apply QA graph neural networks on the recognized figures, multi-modal data comprising text and images, and directly on entire raw pages.

Chapter 2: Background and Related Work. This chapter shows an overview of existing methods for document and graphical content recognition. First, we discuss several page analysis approaches ranging from simple binarization to semantic page segmentation consisting of various document component classes. Next, we present an overview of different methods for visual and textual representation, focusing on compositional embeddings via graph neural networks. Finally, we present related networks for visual and multi-modal question answering, which we grouped into five classes: ranging from global embedding schemes to graph-based reasoning.

Chapter 3: Page Analysis. For applying conventional QA approaches on learning materials, we necessitate a pre-processing step that extracts relevant regions from the page, *e.g.*, natural images, diagrams, and text blocks. We analyze different methods for fine-grained localization and recognize 25 different document component types. In contrast to semantic segmentation (the counterpart task specialized on natural images), we notice a strong overlap between distinct classes (*e.g.*, table and diagram). Thus, we allow our models to assign multiple classes to each region and define several evaluation metrics for a fair comparison between the architectures. Due to the lack of large scale datasets for segmenting learning materials, we collect and analyze two document datasets. We analyze the effectiveness of several models by providing a thorough evaluation study augmented with qualitative results.

Chapter 4: Figure Question Answering. Due to the wide range of figure types encountered in documents (*e.g.*, diagrams, natural images) and their strongly compositional nature, high-level understanding on graphical content remains a challenging

task. In this chapter, we introduce a novel architecture for high-level reasoning of several image types which represents the relational structure of the visual data through a graph. Thereby, the network traverses the visual graph using the question as a guide in search for a node that supplies information about the correct answer. We demonstrate the strength of the proposed network and compare it with state-of-the-art approaches on popular benchmarks for visual question answering.

Chapter 5: Multi-Modal Reasoning. In the multi-modal setting, since the questions are based on either the text or the visual data, it is not known a priori where the information necessary to answer the current question is located. Thus, a path-based approach would not be able to infer an answer, as the path-based network assumes that *the path is encoded in the question.* In this chapter, we propose a relation model that employs an edge refinement technique on the shared embedding of the text and the structured representation of the figure. Finally, we evaluate our proposed approach on a dataset for textbook question answering comprising school lessons including text content and instructional figures in the form of diagrams.

Chapter 6: Visual Reasoning on Pages. Previous question answering methods comprise queries about different components in the page, without considering their relationship and the overall structure of the underlying documents. Thus, we introduce the task of visual reasoning on *entire* pages, which entails answering compositional questions on *both* the graphical content and the text data. Since there are no available datasets in this task, we collect our own benchmark comprising 400k pages in the form of presentation slides, where models have to tackle a variety of complex layouts and document components. Finally, additionally, to multiple baseline and strong deep models, we propose a novel network specialized on page reasoning, which improves the performance of other methods on our dataset.

Chapter 7: Generating Semantic Graphs. The previously proposed approaches for question answering leverage graph neural networks for dealing with the high structured data content. This reasoning procedure entails both *inference from graphs* and *building the graph itself from the input.* While we focused until now on *reason-*

ing based on a graph representation (*i.e.*, we assumed the graph as given), in this chapter, we discuss *how to generate such graphs from data.* To that end, we propose a novel scheme for graph generation which addresses the following difficulties in building semantic graphs: (1) variations in graph sizes between prediction and ground truth and (2) the unordered nature of the nodes in the graph data structure. Finally, we evaluate our approach on several popular datasets for node and graph inference from both textual and visual content.

1.3 Published contributions

The contributions presented in this thesis were published at several computer vision-related venues. The first step in our system chain is analyzing pages in various document types and extracting relevant information for reasoning (Chapter 3). The proposed approaches and our collected datasets were previously published in [Haurilet et al., 2019b; Haurilet et al., 2019c; Bender*, Haurilet* et al., 2019]. In the work of [Haurilet et al., 2019a,d], we reason on diagrams, videos, and synthetic images by traversing the visual graphs in search of relevant information that could be useful for us to answer the current question (Chapter 4). Then, we discuss various types of graphical and textual data that can be useful for our models to extract. For example, textbook question answering is a multi-modal problem where one has to predict answers to questions about both text and diagrams. In [Haurilet et al., 2018], we tackle this task by representing the diagrams as a set of edges and the entire text body as a set of sentences (Chapter 5). We introduce a dataset with a novel neural architecture for answering questions directly on *entire* pages (Chapter 6) in the article published in [Haurilet et al., 2021b]. Finally, in Chapter 7, we introduce an approach for inferring semantic graphs from image and text data without using any external labels (*e.g.*, bounding box annotations) which was already presented in [Haurilet et al., 2021a]. A full list of my publications can be found in Section 8.2.

2 Background and Related Work

This thesis addresses the task of high-level understanding of figures and textual content in pages extracted from learning materials. To that end, we propose several novel machine learning techniques for page processing as well as for visual and multi-modal question answering, which were frequently tackled in the past via specialized supervised classification techniques. In this chapter, we first present a broad overview of machine learning focusing on supervised learning methods especially in the classification setting. Then, we review popular machine learning approaches based on neural networks and discuss shortly their limitations. Section 2.1 revises methods used for page analysis and networks for object localization employed on natural images, as well as compares popular datasets for pixel-wise recognition on documents. In Section 2.2, we dive into methods for visual and textual embedding based on both convolutional and recurrent networks as well as structured representations using graph neural networks. Finally, in Section 2.3, we discuss several networks previously used for question answering ranging from simple global embedding approaches to more complex graph neural networks.

Learning from data. *Supervised techniques* encompass a set of algorithms in machine learning that aim to derive relevant information from *labeled* data. More formally, based on data following a fixed but unknown distribution p_{data}, the overall aim is to select a *fitting* element from a family of functions $\mathscr{F}_\Theta = \{f_\theta | \theta \in \Theta\}$. Frequently one constraints the following: (1) each function $f \in \mathscr{F}_\Theta$ is uniquely defined by the parameter θ and (2) any function from $\mathscr{F}_\Theta$ obtains as input an instance $x \in \mathcal{X}$ (*e.g.*, an image) and generates a value $y \in \mathcal{T}$ from the target space, *i.e.*, $f_\theta : \mathcal{X} \to \mathcal{T}$. We call *learning* the process of selecting the parametrized function $f_{\theta^*} \in \mathscr{F}$ that *fits* our data. Due to our first assumption, this function selection is identical of finding a *fitting* parameter $\theta^* \in \Theta$ associated to f_{θ^*}. Thus, the learning stage comprises a minimization step of a loss function $\mathcal{L}$ between the labeled samples in $\mathcal{T}$ and the output of f_θ assessing the fitness of these functions:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim p_{\text{data}}}[\mathcal{L}(y, f_\theta(x))] \tag{2.1}$$

Neural networks. Neural networks are a specific type of family of functions, that comprise layers, which in turn consist of neurons. We distinguish between (1) an *input layer* that receives as input the given data, (2) an *output layer* that generates the prediction of the network, and (3) a *hidden layer* that established the connection between the input and output layer. The layers comprise entirely of neurons which are simple functions, such as, a linear combination of the input values. To enable the representation of more complex functions than linear ones, each neuron is associated to a so called *activation function* – usually a nonlinear function such as the Rectified Linear Unit $\text{ReLU}(x) = \max(0, x)$ with $x \in \mathbb{R}$. One also differentiates between layer types based on the connections of the neurons in other layers. These connections can be *fully-connected* when each neuron in the current layer is connected with *every* other neuron in the next layer. One can also decrease the amount of connections between the layers, *e.g.*, by inducing a location-based constraint. A popular layer type for such a connection strategy is the *convolutional layer* which comprises only edges between neurons and their location neighborhood in a fixed window (*e.g.*, a neighborhood size of 3×3).

Classification task. A set of attributes that shapes the form of our loss function is the type of the target space $\mathcal{T}$, which in turn depends on the type of problem we address. In this thesis, we only consider classification tasks, where the learning problem is restricted to nominal target values, *i.e.*, the target space is a non-ordered and finite set. To that end, the networks model a probability distribution over the set of all possible classes (*i.e.*, all values are between 0 and 1, and sum up to 1 over the class set). The function used for normalizing the output of the neural networks depends on the problem type. (1) In case of multi-label tasks, one assumes a Bernoulli distribution of the prediction, while for (2) single-label tasks only one class is predicted per instance and assumes a multinomial distribution. Thus, the sigmoid normalization is used for the multi-label classification problems, that is defined as follows: $\sigma : \mathbb{R} \rightarrow [0,1], \sigma(x) = \frac{1}{1+e^x}$. In contrast, for single-label tasks, the softmax function is frequently applied to the outputs of the networks: $\text{softmax} : \mathbb{R}^c \rightarrow [0,1], \text{softmax}(\mathbf{x})_i = \frac{e^{\mathbf{x}_i}}{\sum_j e^{\mathbf{x}_j}}, \forall i, j \in \{1, \ldots, c\}$.

Finally, we learn the weights of the network by minimizing a loss function that depicts the discrepancy of the output of the network to the ground truth annotation. We will use throughout this thesis the Cross Entropy (CE), which is a popular loss function employed in classification tasks:

$$\mathcal{L} : [0,1]^c \times \{0,1\}^c \rightarrow [0,1], \mathcal{L}(\mathbf{y}^{\text{pred}}, \mathbf{y}^{\text{gt}}) = -\sum_i \mathbf{y}_i^{\text{gt}} \log(\mathbf{y}_i^{\text{pred}}) \tag{2.2}$$

where c is the number of classes, $\mathbf{y}^{\text{pred}}$ is the output after normalization, while $\mathbf{y}^{\text{gt}}$ depicts if a specific class is present in the current sample. The reason of the popularity of the CE loss is its property that by minimizing $\mathcal{L}$ one equivalently minimizes the Kullback-Leibler divergence [Kullback, 1987] between the two distributions.

Dataset. In practice, there is only a limited amount of data that is used to approximate our function f_θ. Even though, in some cases (*e.g.*, reinforcement learning) the data can be generated dynamically, in our case, we use *fixed* data that we split into training, validation, and testing. Thus, our datasets are of the form $\mathcal{D} = \{(x, y) \mid x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$, where x is called an *instance* and y a *target* or *label*.

2.1 Page analysis

Page recognition is a popular task in computer vision that analyzes the page of a printed or digital document by breaking the visual representation of a page into relevant parts. We distinguish between the subtasks based on the type of components we want to extract from the page: (1) *binarization* splits the page into foreground and background regions, (2) *text localization* detects textual content in the page, and (3) *semantic page recognition* localizes textual, graphical, and structural components.

2.1.1 Segmentation techniques

Semantic segmentation. Semantic image segmentation is a task that aims to estimate the fine-grained location of objects in a visual scene. More precisely, models for semantic segmentation associate *each* pixel in the input image with a corresponding semantic class. Since such models generate fine-grained localizations of the object instances in the scene, this task has a wide variety of applications ranging from robotics to autonomous cars and assistive technology systems.

Fueled by the progress in computer vision and by the availability of publicly available benchmarks [Caesar et al., 2018; Cordts et al., 2016; Everingham et al., 2015; Lin et al., 2014; Neuhold et al., 2017], the task of semantic image segmentation has experienced rapid advancements in recent years [Liang-Chieh et al., 2015; Lin et al., 2016; Schwing and Urtasun, 2015; Yu and Koltun, 2015; Zhang et al., 2015]. Most of the architectures for semantic segmentation encompass an hour-glass shape comprising an *encoder* and a *decoder* network. While the encoder realizes a large receptive field by decreasing the image resolution via convolutional and pooling layers with a large stride, the decoder inflates the resulting encoding. The expansion of the output of the encoder follows on the location dimensions through deconvolutional and unpooling layers [Noh et al., 2015] until the height and width of the final maps are equal to the input image. Finally, the output of the decoder

has the shape $h \times w \times c$, where h and w stand for the height and width of the input image, and c is the number of classes. By normalizing the predicted values using softmax, one models a probability distribution for each pixel over the c classes.

Page segmentation. In comparison to semantic segmentation, page segmentation methods did not experience such strong recent advancements as segmentation methods for natural images. In page segmentation, we distinguish between bottom-up and top-down approaches, which both are based on feature engineering and often do not even require any training data (*e.g.*, by using unsupervised approaches like clustering or thresholding). Bottom-up approaches [Amin and Shiu, 2001; Drivas and Amin, 1995; Hinton et al., 2006; Lebourgeois et al., 1992] combine similar pixels of even groups of pixels (*i.e.*, superpixels) until a pre-defined condition is reached. In case of top-down methods [Ha et al., 1995a,b], the entire page is considered at the beginning, which is then split into its components based on some predefined strategy (*e.g.*, cutting in the x- and y-axis based on the distribution of dark pixels). Deep learning models have only been scarcely used for page analysis, *i.e.*, for segmenting scientific papers [Breuel, 2017; Tensmeyer and Martinez, 2017; Yang et al., 2017] and historical documents [Chen et al., 2015a, 2017]. While some networks address the lack of large scale datasets by employing shallow architectures, *i.e.*, comprising only few layers [Chen et al., 2015a, 2017], other networks compensate the lack of data by generating synthetic papers [Yang et al., 2017]. Nonetheless, page segmentation is highly related with semantic segmentation on natural images and, thus, similar hour-glass architectures can easily be transferred to page segmentation.

The work of [Chen et al., 2017] addresses the problem of segmenting historical documents of handwritten text on a very small dataset of less than 100 images. Due to the small size of the datasets, a shallow neural network is employed that consists of a convolution and two fully-connected layers. Tensmeyer and Martinez employ a Fully Convolutional Network (FCN) which was previously used for segmenting natural images on page binarization of documents comprising handwritten text [Tensmeyer and Martinez, 2017]. In contrast, the work in [Breuel, 2017] addresses the binarization problem of contemporary articles and proposes a combination of convo-

lution layers and LSTMs to capture the global structure of the page. Finally, in [Yang et al., 2017], both the visual data and the associated textual content is combined to improve the segmentation performance on a dataset for contemporary articles.

2.1.2 Datasets for page analysis

While a wide variety of large scale benchmarks for semantic segmentation on natural images were proposed in recent years, datasets for page recognition are scarce. Most of the available page datasets provide either a small number of samples [Clark and Divvala, 2016; Luong et al., 2012; Tao et al., 2014; Yang et al., 2017] which make them inadequate for deep learning approaches or comprise only a small number of classes [Clark and Divvala, 2016]. The lack of large scale datasets is a possible reason for the small number of available neural architectures for page segmentation.

RDCL [Antonacopoulos et al., 2015] is a layout recognition dataset, which captures in total 77 pages from magazines and journals used for training and testing. While this dataset comprises several text-based classes, *i.e.*, caption, credit, paragraph, page number, heading, drop capital, footer, and floating text, the number of image-based classes is low containing only decoration and images. The DSSE-200 [Yang et al., 2017] dataset considers six document classes: figures, tables, section headings, captions, lists, and paragraphs, of which it provides bounding box annotations for 200 pages. A larger dataset was introduced by [Tao et al., 2014] which covers in total 244 images of pages extracted from 35 English and Chinese e-books. Sectlabel [Luong et al., 2012] encompasses bounding box annotations on a wide variety of text-based classes for pages extracted from the PDF format of scientific articles. The CS-150 and CS-Large [Clark and Divvala, 2016] benchmarks are publicly available and comprise in total 150 and 3100 pages, respectively, with labels in form of bounding boxes localizing text, figures, tables, and the corresponding figure captions. While CS-150 only provides labels for articles that were released on only three conferences, CS-Large captures randomly selected papers from Semantic Scholar that have at least nine citations.

2.1.3 Text recognition

Optical character recognition is a popular task in computer vision aiming to translate an image into its underlying pre-defined text representation. Inspired by the success of neural networks in problems such as image captioning where one generates text from natural images, current methods for optical character recognition leverage a similar network structure. Namely, such networks have an encoder-decoder structure, where the *encoder* comprises a Convolutional Neural Network (CNN) and the *decoder*, frequently a recurrent network, generates the text sequence [Wang et al., 2012, 2019; Xie et al., 2019]. Other extensions of these architectures improve recognition results by using stacked encoder and decoder blocks [Xie et al., 2019], leveraging an alignment module between encoder and decoder [Wang et al., 2019], correcting text distortions [Zhan and Lu, 2019], employing attention mechanisms [Liao et al., 2019], or leveraging stacked attention modules [Li et al., 2019].

2.2 Modeling vision and language

2.2.1 Visual embeddings

Since the rise of deep learning, computer vision has experienced tremendous progress. A major propellant of these rapid advancements lies in the success of neural architectures on large scale object recognition datasets like ImageNet [Russakovsky et al., 2015] and Kinetics [Carreira and Zisserman, 2017]. Pre-trained networks are frequently used as a resource for strong semantic embeddings of visual data. Thereby, the features are extracted from intermediate layers and are used as input to different techniques for high-level understanding, *e.g.*, reasoning [Wu et al., 2017], image captioning [Xu et al., 2015], and zero-shot recognition [Qiao et al., 2016; Xian et al., 2017]. In case of attention-based techniques, the features are generated from the convolutional layers generating 3D tensors, while for global embedding schemes vectors from fully-connected layers are used.

2.2.2 Sentence representation

In a similar manner, as for the visual data samples, one can embed sentences into semantic representations by mapping its units, (*i.e.*, words, sub-words, or characters) into a global encoding. The words are represented by vectors, which can be simple one-hot embeddings, *i.e.*, binary vectors directly coupled with the word, or *semantic* vectors extracted, for example, from a pre-trained network [Pennington et al., 2014]. A global encoding is further obtained through recurrent networks [Conneau et al., 2017] or 1D convolution layers [Vaswani et al., 2017; Zhang et al., 2018].

2.2.3 Graph generation

Generating graphs from images. A variety of recent visual reasoning methods [Das et al., 2018a; Teney et al., 2016; Yang et al., 2019; Yao et al., 2018] operate on graph encodings of the visual scene, where the nodes portray the objects while the edges depict their relationship type [Krishna et al., 2016]. In order to generate such graphs, current methods rely on localization techniques, *e.g.*, object detectors, to recognize the nodes [Krishna et al., 2016; Lu et al., 2016a]. Object localization is a well-studied problem in computer vision addressing simple coarser detection via bounding boxes [Girshick, 2015; Liu et al., 2016; Redmon et al., 2016] to fine-grained segmentation [He et al., 2017]. There are two popular object detection techniques frequently used for coarse localization: (1) proposal-based methods and (2) single-shot detectors. Proposal-based methods comprise an initial step where proposals are generated using clustering [Uijlings et al., 2013], SVMs [Cheng et al., 2014], or neural networks [Ren et al., 2015b] that are forwarded to a second model that predicts if the bounding box contains any known objects. More popular model types for node recognition are single-shot detectors, which are trained *end-to-end* using a visual 3D tensor [Liu et al., 2016; Redmon et al., 2016; Ren et al., 2015b]. Each vector in the 3D tensors is associated to a set of object centers inferring a set of confidences being the center of an object and the corresponding possible classes. Finally, each predicted bounding box serves as a node in the sought graph.

The second step constitutes the inference of the relations between each recognized node pair. In case of *location-based* edge types, the graphs are fully-connected (*i.e.*, a label is assigned to each node pair), while for *semantic* relation detection the scene graphs are very sparse with only few annotations per node pair. Typically, relationship detection models place a bounding box over the node pair in the image, which is cropped and passed to a classification network. In case of location-based relations, a direct mapping of the found positions of the nodes is also frequently employed. Recent methods additionally refine the estimated graph through graph convolutions [Xu et al., 2017; Yang et al., 2018b], external knowledge [Gu et al., 2019a], language priors [Lu et al., 2016a], multi-task learning [Li et al., 2017], or constrain the number of object pairs by leveraging heat-maps [Newell and Deng, 2017].

Inferring graphs from text. While various machine learning methods have been introduced for inferring *text from graphs* [Koncel-Kedziorski et al., 2019; Wang et al., 2018; Yang et al., 2019; Yao et al., 2018], the research of generating *graphs from text* has been scarce so far [Cohan et al., 2020; Das et al., 2018b; Gu et al., 2019b]. Most methods operating on text data either utilize a recurrent neural network and generate a single node in every iteration step [Das et al., 2018b] or pass a fixed node set as input and only recognize the edges [Cohan et al., 2020; Gu et al., 2019b].

Graph auto-encoding. Another type of models for graph embedding, regularization via graphs, and graph generation are ones for graph auto-encoding. Methods for graph auto-encoding aim at injecting the edge matrix into the node encoding [Feng and Duarte, 2018; Gidaris and Komodakis, 2019; Pan et al., 2018; Simonovsky and Komodakis, 2018], *e.g.*, through consecutive graph convolutions [Kipf and Welling, 2017]. A decrease in memory size is established by absorbing the edges into the vertex representations and by decreasing the dimensionality of the nodes. Thus, in current methods for graph auto-encoding, the set of nodes directly impacts the variable-sized dimension of the graph embedding.

2.2.4 Representing relations

The rich structure of the scene can be captured in an effective way through *graphs*, where the object instances are represented by nodes while their relations are depicted by edges. Graph neural networks for data representation have already been utilized in a wide range of applications, such as language [Kuhlmann and Oepen, 2016], social interaction [Kok and Domingos, 2007; Sen et al., 2008; Yang et al., 2011], knowledge representation [Bouchard et al., 2015; Mahdisoltani et al., 2013; Toutanova et al., 2015], and chemistry [Radivojac et al., 2013]. The graphs can be captured directly from the CNN feature maps [Kim et al., 2017; Li et al., 2018b; Xu et al., 2017] or by combining the existing graph representations with the previously acquired knowledge base [Das et al., 2018a; Xiong et al., 2017]. While such information is passed directly as input to the neural networks, the models need to have an effective way for dealing with these data structures. There are several obstacles that make the grasp of such data difficulty: (1) the nodes in the data structure are *unordered*, (2) the adjacency tensor modeling the relations often requires a lot of memory usage, and (3) graphs can have different number of nodes.

Several previous approaches deal with these difficulties, *e.g.*, by padding the graphs to a common size or by assuming an ordering of the nodes based on their prediction confidence. We distinguish four groups of knowledge base-guided algorithms: (1) recurrent-based methods [Teney et al., 2016], approaches using graph refinement either (2) for a better node representation [Do et al., 2019; Kipf and Welling, 2017; Teney et al., 2016; Velickovic et al., 2017], or (3) for refining the edges [Santoro et al., 2017; Simonovsky and Komodakis, 2017], and (4) graph traversal approaches [Das et al., 2018a; Xiong et al., 2017]. While recurrent-based techniques assume an ordering of the nodes, node refinement methods frequently leverage graph convolution layers [Kipf and Welling, 2017] which leverage convolutions on the neighboring nodes for each of the vertices. In case of edge refinement, each edge is re-encoded to a new representation using, *e.g.*, fully-connected layers. Previous graph traversal approaches learn to pace graphs using reinforcement learning techniques. We further discuss such graph-based models in Section 2.3.5.

2.3 Visual question answering

Visual reasoning is often posed in the form of Visual Question Answering (VQA) – a task lying in the intersection of vision and language which intends to answer questions about the visual scene. Thereby, we distinguish between (1) multiple choice answers (few possible answers are paired to the current query) and (2) open-ended questions where we do not constrain to a small set of possible answers. Thus, in the open-ended setting, we aim to predict the correct answer that:

$$\hat{a} = \arg\max_{a \in \mathcal{A}} \mathbb{P}(a|X, \mathbf{q}; \theta) \tag{2.3}$$

where a is an answer from the set of all possible answers $\mathcal{A}$, X is an image, and $\mathbf{q}$ the input question encoding. Thereby, we model the probability distribution as in the previous case, *i.e.*, using a neural network. Finally, we select the answer that achieves the highest probability based on our input data.

VQA has rapidly gained popularity over the past years [Agrawal et al., 2017; Gurari et al., 2018; Krishna et al., 2016; Zhu et al., 2016], mostly being addressed through image feature maps extracted from a pre-trained CNN and subsequent question-related attention module [Yang et al., 2016; Yu et al., 2017]. In general, the ways of addressing this problem can be divided into four categories: (1) *global embedding methods* [Agrawal et al., 2017; Ma et al., 2016; Malinowski and Fritz, 2014; Malinowski et al., 2017; Ren et al., 2015a] that use a joint embedding of the global image representation and the question to produce an answer; (2) *attention-based models* that focus to parts of the image based on the question [Fukui et al., 2016; Xu and Saenko, 2016; Yang et al., 2016; Yu et al., 2017]; (3) *compositional models* [Andreas et al., 2016; Hu et al., 2017; Johnson et al., 2017b] use a modular representation of the neural networks; (4) *graph-based VQA models* [Kembhavi et al., 2016; Kim et al., 2017; Santoro et al., 2017; Teney et al., 2016], where a graph representation of the image or the question is used to produce the answer.

2.3.1 Global embedding methods

Visual and textual encoding. Global embedding methods represent the entire visual and question input through a single vector that is then fused with the input query in the prediction module which, in turn, answers the current question. These type of approaches employ popular CNNs previously used for object recognition, *e.g.*, VGG [Jabri et al., 2016; Ma et al., 2016; Ren et al., 2015a], GoogLeNet [Malinowski et al., 2017], and ResNet [Ben-Younes et al., 2017; Jabri et al., 2016], which were pre-trained on the large scale ImageNet dataset [Deng et al., 2009]. The questions are also mapped into a single global embedding using the final hidden layer of an LSTM [Malinowski et al., 2017; Ren et al., 2015a], a GRU [Perez et al., 2017], by averaging word2vec representations [Mikolov et al., 2013] of the words in the query [Jabri et al., 2016], or via 1D-convolutions [Ma et al., 2016]. Succeeding the visual and textual encoding step, the multi-modal fusion module combines the visual and textual data into a single representation. The fusion techniques for VQA can be divided by their interaction complexity into first- and second-order interaction. While first-order fusion comprises a flat interaction between input streams, second-order methods model complex interactions between each feature-pair.

First-order fusion. The method presented in [Jabri et al., 2016] tackles only the multiple choice QA task by employing a Multi-Layered Perceptron (MLP) on the concatenation of the question and image feature. In [Malinowski et al., 2017; Ren et al., 2015a], the first hidden state used to embed the input query is initialized using the visual feature vectors. While in [Malinowski et al., 2017], the LSTM generates the answers word-wise after all the words of the question were passed into the model, in [Ren et al., 2015a] an MLP generates the answer from the final hidden state of an LSTM. Ma *et al.* propose a multi-modal convolution layer that combines the words in the question with a vector representation of the image [Ma et al., 2016]. In [Perez et al., 2017], the question is combined with the 3D tensor extracted from a CNN using a Feature-wise Linear Modulation (FiLM) layer, which comprises an element-wise multiplication and addition.

Second-order fusion. Second-order pooling is a technique for capturing the interaction between different input streams by combining *each* pair of features between two input streams. Multimodal Compact Bilinear pooling (MCB) employs element-wise product in FFT space, *i.e.*, convolving both the textual and visual vectors [Fukui et al., 2016]. MUTAN [Ben-Younes et al., 2017] interpolates the global representations of the image and question via a single large 3D tensor that uses the Tucker decomposition [Tucker, 1966] strongly decreasing the number of parameters.

Prediction network. Finally, multi-modal embeddings are passed to a prediction module comprising an MLP with a classification layer. For the open-ended setting, the network selects a response from all possible answers seen during training. In contrast, for multiple choice questions, the network models a probability distribution over the given multiple choice answers, where the answer with the highest activation is selected for the final prediction.

2.3.2 Attention-based models

Global embedding schemes encode the *entire* visual data into a *single vector* encouraging the network to keep the semantic information and discard location information. Even though such properties might be useful for object recognition, VQA tackles multiple other tasks beyond conventional recognition, *e.g.*, counting task. Attention mechanisms [Xu et al., 2015] allow networks to focus on parts of the image and thus concentrate on the relevant information in the visual data.

Visual and textual encoding. While the questions are either encoded by an LSTM [Chen et al., 2015b; Yang et al., 2016] or by a 1D-CNN [Lu et al., 2016b; Shih et al., 2016; Xu and Saenko, 2016], the image embedding scheme *keeps* the location information by removing the final fully-connected layers. Vector-slices of the generated 3D tensor are then used to represent regions of the image. As in the case of the global embedding schemes, the image is encoded by a pre-trained CNN, *e.g.*, VGG [Chen et al., 2015b; Lu et al., 2016b; Shih et al., 2016; Yang et al., 2016], GoogLeNet [Xu and Saenko, 2016], and ResNet [Lu et al., 2016b]. Since attention

can be applied on the images alone or simultaneously on both image features and words in the question, we distinguish in the following between 'vanilla' attention mechanisms and networks using co-attention modules.

Simple attention mechanisms. Attention modules combine each vector sliced in the location axis of the 3D tensor with a vector representation of the question (*e.g.*, the last hidden state of an LSTM). Then, confidences of each region being relevant to answer the current question are estimated by using fully-connected layers [Yang et al., 2016], an LSTM over the words of the question [Chen et al., 2015b], or simply element-wise multiplication [Xu and Saenko, 2016]. The values in the confidence matrix are normalized using softmax on both dimensions such that the entire matrix sums to one [Kiros et al., 2015; Yang et al., 2016] or using the sigmoid function aiming to keep the values in the scale between zero and one [Chen et al., 2015b]. Then, the attention maps are used to weight the vector slices generated by the CNN, which are then summed up for a final visual vector representation.

Co-attention modules. When focusing on regions of the image, some parts of the questions are more relevant for generating the answer than others (*e.g.*, What color is the *umbrella* in the image?). To this end, co-attention mechanisms focus on parts of the question and image *simultaneously*. This paired attention mechanisms is applied multiple times enabling multi-step reasoning [Lu et al., 2016b; Yu et al., 2019]. The visual attention maps are used in a similar manner as in case of the 'vanilla' attention modules, namely, for weighting the 3D tensor slices extracted from the CNN. In case of the input query, the attention maps are only used to obtain the attention mechanisms over the images. Thus, the textual attention over the words is only used to refine the visual representations.

Prediction network. Finally, the visual vectors are combined with the question representation identically as in case of the global embedding methods by fusing them through concatenation or addition and employing a fully-connected layer.

2.3.3 Memory networks

Memory networks are a specialized type of architectures that leverage a *neural external memory* [Graves et al., 2014] that can be accessed by the network through *reading heads*, while new information is induced into memory through *write operations*. The memory is composed of vectors that depict the memory cells and, thus, the entire memory is fully described by a *single* matrix.

Visual and textual encoding. As for the attention-based techniques, the visual embeddings are extracted from a convolution layer from a pre-trained CNN, *e.g.*, ResNet in [Hudson and Manning, 2018; Ma et al., 2018] and VGG in [Li et al., 2018a; Ma et al., 2018]. In contrast, for the textual domain, a bidirectional recurrent net is used in [Li et al., 2018a] or a GRU with attention mechanism in [Xiong et al., 2016].

Memory. The memory comprises a set of cell-vectors that constitute information extracted from the image. The memory matrix is updated in multiple steps, carving the *reasoning* process of the network. The selection of the visual content to be written into memory comprises a co-attention module followed by concatenation of the question and image vector in [Ma et al., 2018], a multi-modal global encoding extracted via an attention mechanism [Hudson and Manning, 2019; Li et al., 2018a], or a recurrent model [Xiong et al., 2016]. The selected information is then leveraged to update the memory cells, which is performed by a GRU on the current memory cell and the GRU hidden state [Xiong et al., 2016], by adding the previous stored information and the hidden state of an LSTM [Ma et al., 2018], or using a global visual encoding obtained via an attention module [Hudson and Manning, 2018]. Thus, the update is conditioned on the visual information [Li et al., 2018a], on the memory state [Hudson and Manning, 2018], or on the previous memory updates [Xiong et al., 2016].

Prediction network. Finally, a fully-connected layer is employed to generate the answer using a visual encoding obtained by a read operation from the memory via weighted average pooling over the cells [Hudson and Manning, 2018; Li et al., 2018a; Ma et al., 2018; Su et al., 2018; Xiong et al., 2016].

2.3.4 Compositional models

The previously presented networks include attention mechanisms and memory matrices, where multi-step reasoning is established through stacking [Yang et al., 2016]. To counteract this and enable more interpretable multi-step reasoning, a novel type of methods was proposed that generates the structure of the visual embedding network conditioned on the input-question. This visual network comprises modules that can be attention mechanisms, element-wise operations (*e.g.*, addition), or convolutional layers. The composition of the network can either be estimated using grammar rules based on the input query [Andreas et al., 2016] or learned through the network in an end-to-end manner [Gupta et al., 2020; Hu et al., 2017].

Visual and textual encoding. As in the other models, the image is embedded using a pre-trained CNN with the final fully-connected layers discarded (*i.e.*, the output is a 3D tensor). A large difference to previous methods lies in the representation of the input query that now serves as a network structure estimate. The input query is parsed using, *e.g.*, a natural language parser [Manning et al., 2014], grammar rules [Gupta et al., 2020], or by learning end-to-end through expert-annotations [Hu et al., 2017]. To that end, the output of the query representation is the layout of which the reasoning network is composed.

Neural modules. While the weights of the modules (*e.g.*, convolutions) are learned by minimizing the loss over the question-answer pairs, the different modules are defined by a human expert and coupled to different types of words in the input query. The composition of the modules defines the structure of the reasoning network, which is estimated using the layout generated in the previous step.

Prediction network. Finally, the output of the reasoning network is either combined with the input question (*e.g.*, via concatenation) to generate the answer [Andreas et al., 2016; Gupta et al., 2020] or the raw output of the neural modules alone are leveraged for the final prediction [Hu et al., 2017].

2.3.5 Graph neural networks

While neural module networks require strong human intervention, memory networks are not human interpretable having a black-box structure. A novel type of networks were recently proposed called graph neural networks [Kipf and Welling, 2017] that operate directly on graph data structures. Due to the highly structured nature of images, one can easily represent the visual scene as graphs where the nodes represent object instances and edges depict the relationships between them. There are several graph neural networks proposed for VQA, which we group into four categories: (1) *recurrent-based networks* that traverse the graphs in a pre-defined order, (2) *node refinement methods* where in each reasoning step the nodes refine the embedding using the neighboring nodes, (3) *edge-pooling approaches* that refine the edges between the nodes in each processing step, and (4) *path-based approaches*, which traverse the graphs in search of relevant information for answering the question. Next, we will discuss each graph network type separately.

(1) Recurrent-based approaches. Recurrent Neural Networks (RNNs) are architectures employed for embedding or generating sequential data (*e.g.*, embedding words in a sentence). Since RNNs inherently deal with variable-sized dimensions, which we are confronted with when dealing with graphs (*i.e.*, variable-sized node set), they are possible candidates for graph encoding. However, since RNNs necessitate sequential data, an ordering of the nodes in the graph has to be defined. In [Kembhavi et al., 2016], an RNN is applied on the edges which are subsequently filtered through a question-based attention, while in [Kim et al., 2017] an end-to-end version is proposed, where the edges are learned inside the model. In both of these methods the ordering of the nodes is carved by the confidence of the model for each node (*i.e.*, output of the highest softmax activation). The final state is passed to the prediction module, which consists of fully-connected layers. To counteract the ordering problem, the work in [Teney et al., 2016] proposes to refine the node features using an RNN via similarity-based pooling proceeded by averaging the hidden states for each node obtaining a fixed visual encoding.

(2) Node refinement techniques. Models based on node refinement [Narasimhan et al., 2018] require a set of node encodings (*e.g.*, extracted from a CNN) and adjacency tensor (*i.e.*, comprising binary values showing the connection type between each pair of nodes). The refinement process entails several graph convolution layers that comprise convolving the node encodings with learnable weights based on the adjacency tensor. Namely, given a matrix representation $V \in \mathbb{R}^{n \times d}$ of n nodes and an adjacency matrix $A \in \{0,1\}^{n \times n}$, the graph convolution function gconv can be computed by simple matrix multiplications between the input variables:

$$\text{gconv}(V, A, W) = \sigma((A \cdot V) \cdot W) \tag{2.4}$$

where the function σ is an activation function such as ReLU and $W \in \mathbb{R}^{d \times h}$ is a learnable weight matrix that maps the input node vectors of size d to h-dimensional vectors. Thus, while the adjacency matrix the features of all neighboring features for each node separately (via $A \cdot V$), the weight matrix maps these combined features to a new embedding space (via the multiplication with W). For VQA, this operation is applied several times on the input node features and, then, a global representation of the graph is obtained through global average pooling on the variable dimension of the node set. This global graph representation is then combined with the question (*e.g.*, via concatenation or sum) to infer the final answer using a prediction layer.

(3) Relational-embedding methods. Another type of networks [Battaglia et al., 2016; Santoro et al., 2017] are edge-centric approaches, which represent the graphs as an unordered set of edges each of which is framed as a vector. Therein, the edges can be semantic predicates between the objects (*e.g.*, `holding`), or node pairs (*e.g.*, concatenating the two vectors representing the two nodes). The variable-sized dimension caused by the change of size of the edge set is discarded through weighted average between the edges obtaining a fixed visual vector representation. As in the previous cases, the global graph vector representation is passed through fully-connected layers which output a probability distribution over the answer set.

(4) Path-based approaches. While node and edge processing methods *refine* the graph input based on the question, in path-based approaches the graphs remain *fixed* during the entire inference. Even though there are no previous path-based methods for QA on images, the recent works of Xiong *et al.* [Xiong et al., 2017] and Das *et al.* [Das et al., 2018a] address the question answering task on knowledge graphs by generating paths. The authors represent text-based knowledge as a graph, where the nodes determine knowledge terms (*e.g.*, `Germany`) and the edges show their relationship (*e.g.*, `capital-of`). To answer an input query, the models generate *discrete* paths (*i.e.*, a *single* node is traversed at a time). Due to the discrete nature of the paths, the networks are not differentiable and, thus, conventional back-propagation cannot be used for training. Thus, the training procedure follows the REINFORCE [Williams, 1992] algorithm loosening the discretization of the node assignment. Moreover, these models entail that at each iteration step a single node is visited (due to the discrete nature of the approach). This property makes it impossible for the networks to .

3 Page Analysis

In this thesis, we aim to analyze models for *visual and multi-modal reasoning on learning materials*, where various types of figures (*e.g.*, diagrams, natural images) and modalities (images and text) come into play. However, in previous years, conventional question answering models mostly focused on *single* figure and modality types, contrary to the rich data variety incorporated in learning materials. To that end, we follow two different directions for high-level reasoning on entire pages: (1) localizing different components in the page and enabling the use of off-the-shelf QA techniques dependent on the data type and (2) applying QA models trained end-to-end on image representations of pages. For the first direction, we first need to *localize* the different components in the page and *classify* them to their respective type (*e.g.*, natural image, diagram). In the first setting, the additional page pre-processing step enables the use of conventional visual and text-based question answering *directly* on the detected page components (Figure 3.1). In this chapter, we address the problem of page analysis and fine-grained page-component localization as a pre-processing step for our question answering task. Therein, we focus in analyzing two learning material types: (1) digital presentation slides and (2) photos of slides cap-

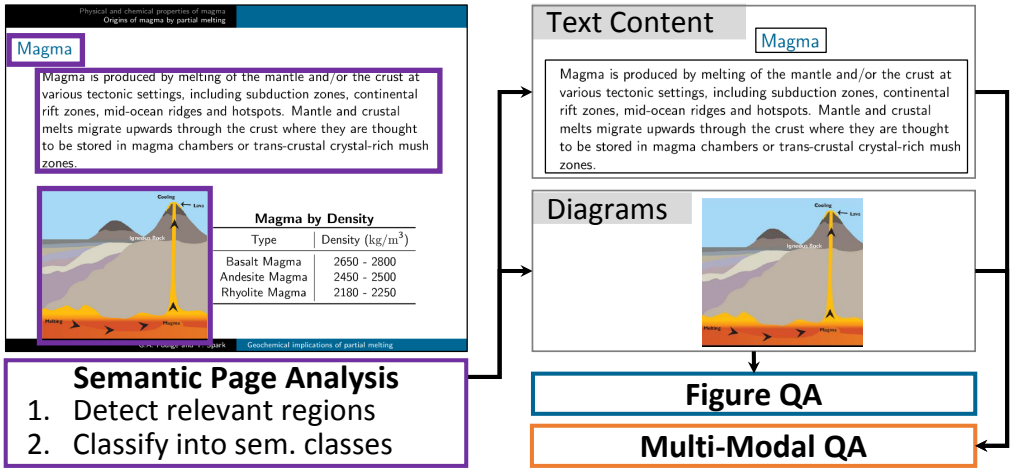


Figure 3.1: The pipeline for the first direction that we take for reasoning on learning materials: (1) localizing page components and (2) applying different conventional QA techniques based on the recognized component types.

tured during lectures (*i.e.*, captured in the wild). In comparison to other document types like textbooks, magazines, and scientific papers, presentation slides comprise a high variety of layouts as well as capture a wide range of different figure types.

Contributions. In this chapter, we address the component localization problem in documents and propose a novel task of page segmentation of *presentation slides*. Since page analysis of digital slides and slides in the wild was not approached before, we collect our own datasets and analyze the data by various methods proving their unique properties, *e.g.*, overlap between different classes and location variance of several document components. We tackle this task and analyze several neural architectures previously used for semantic segmentation on natural images on our collected datasets. Additionally to several deep networks, we evaluate multiple baseline approaches and show that the neural networks are able to improve their results by a considerable margin. Finally, to compare the models on our page segmentation problem, we introduce novel evaluation metrics specialized for measuring the discrepancy between pixel-wise, overlapping regions.

Publications. This chapter is based on the works published in [Haurilet et al., 2019b] and [Haurilet et al., 2019c].

3.1 Slides in digital format

Presentation slides are one of the most prominent document formats to share information and ideas for educational and business purposes. For example, the slide sharing service SlideShare[1] alone states to have more than 400 thousand slides uploaded monthly with an estimated 80 million unique visitor per month[2]. Thus, in order to tap this massive knowledge base it is crucial to enable *automatic analysis* of such documents. Automatic document understanding and retrieval enables millions of users to have fast and convenient access to the sought information. Moreover, slides are used massively in many educational institutes, hence, it is quite important to enable students with visual impairment to have a convenient and reliable access to this knowledge source. However, the research of automatic slide analysis is scarce as no publicly available datasets exist to enable the vision research community to tackle this problem. The availability of such a dataset is vital for developing novel and accurate approaches for slides automatic understanding and visual analysis. Additionally, a large scale and diverse dataset is crucial to benchmark and enable modern machine learning technique like deep learning.

Page analysis aims to extract the semantic components of a document page represented in digital format, *i.e.*, images. These components are usually related to the layout (*e.g.*, headers and footnotes) or to the individual page contents (*e.g.*, tables and diagrams). In this thesis, we approach the page segmentation task as a *pixel-wise* classification task where each pixel in the page is classified into a subset of predefined categories. Page segmentation is quite relevant to the popular image segmentation task [Everingham et al., 2015] where natural images from indoor or outdoor environment are segmented into one of the defined object categories like person, bike, or building. Nonetheless, there are some key differences between these two segmentation problems. For example, unlike image segmentation where we usually deal with relatively large scale objects like car, road, sky, in page segmentation, we need to handle a fine-grained set of classes that usually occupy only a tiny

[1] http://slideshare.net
[2] https://blog.linkedin.com/2015/08/25/

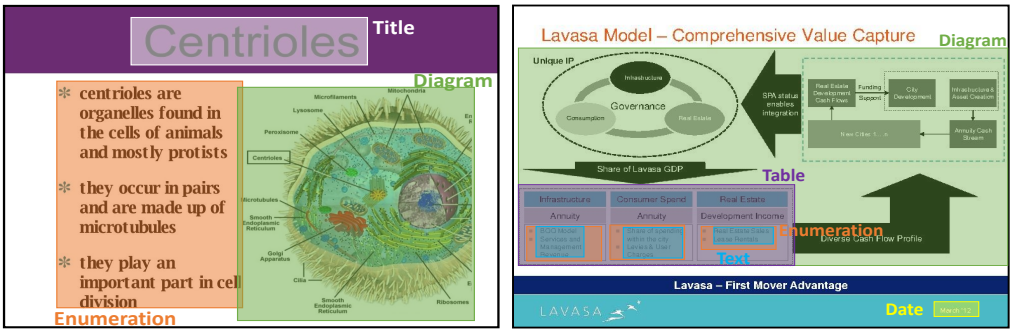


Figure 3.2: Examples pages from our proposed dataset. We associate each page with document component labels spanned over three categories: textual, structural, and graphical elements. Since distinct semantic classes are overlapping (right), we allow each pixel to be mapped to multiple classes.

spatial area of the image like footnote, page number, or legend (Figure 3.2 right). Furthermore, the semantic categories of components in natural images is *location invariant* (*e.g.*, a car is a car regardless whether it appears on the top, left, or to the right part of the image). However, in page segmentation, components represent different semantics depending on their size and spatial location in the page; for example, *Centrioles* in Figure 3.2 (left) appears as both a `title` and a `bullet-point`. Additionally, there is a high overlap of labels in page segmentation as a pixel can belong to *multiple categories at the same time*, *e.g.*, in Figure 3.2 (right) we have pixels that are part of four different classes: `text`, `bullet-point`, `table`, and `diagram`.

To that end, we collect and publicly release a dataset for **S**lide-**Pa**ge **Se**gmentation (SPaSe), which augments the publicly available SlideShare-1M dataset [Araujo et al., 2015] for slide retrieval with segmentation labels. Our dataset comprises 2000 labeled slide images with fine-grained annotations of 25 classes (*e.g.*, `title`, `drawing`, and `table`). The dataset has a high intra-class variance where, *e.g.*, `plots` and `text` can be both produced digitally and handwritten (Figure 3.2). Additionally, the collected slides are multilingual where in addition to English there are languages like Spanish, French, and Romanian. This creates an interesting and challenging benchmark for tasks like text segmentation. While it is common in image segmentation to have a *single* annotation associated to each pixel (*e.g.*, [Cordts et al., 2016; Evering-

ham et al., 2015; Lin et al., 2014]), this is inadequate for slide segmentation due the multi-facet nature of the slide components. Thus, we provide multi-label annotations of individual pixels to capture the overlapping semantic representations of objects. Furthermore, we define novel evaluation metrics to quantify the performance of different segmentation approaches in our multi-label setup. In a thorough evaluation on SPaSe, we show that our large scale dataset enables deep learning methods to be trained from scratch without the need of additional data sources. Moreover, we analyze the correlation of the defined categories and locations and demonstrate the impact of spatial information on segmentation performance. In summary, we show the following unique properties of the collected dataset: (1) the visual data is balanced in the semantic classes, (2) a subset of classes are frequently co-occurring, (3) document components in pages are location-variant, and (4) a high number of pixels are associated to more than one label. Finally, we adapt and evaluate several popular approaches for natural image recognition on our dataset and show that positional embeddings can improve the performance of models for page segmentation.

3.1.1 Data collection and annotation

Next, we describe the collection and annotation process of the SPaSe dataset for page segmentation of digital slides.

Slide selection. SlideShare-1M is a publicly available dataset of one million presentation slides scraped automatically from the website SlideShare and leveraged for tackling the novel task of slide retrieval. However, this dataset contains very similar or duplicate slides, pages containing only single words, and images with very low resolution. To retain a high quality of the page images in SPaSe, we apply the following slide selection process: (1) we choose at most one slide from each presentation available in the SlideShare-1M dataset, (2) we discard presentations with low quality images, and (3) we select slides with a high variety of document component types.

Category definition. We then define the set of categories used to annotate the pixels of the 2000 selected slides. For that purpose, we collaborate with an academic

institute which offers aid services for students with visually impairment. One of their main activities is to provide detailed description of lecture materials like papers, exams, and presentation slides. This is usually achieved through manual effort of tens of trained assistances. Theses assistances will go through the large amount of slides and supply structured descriptions that are tailored to the needs of the visually impaired in regards of the slides structure and content. These descriptions represent an excellent data source to identify the most frequent and important object types that are usually encountered in the slides. Based on this data source and discussions with the experts, we identified 25 relevant categories. In addition to the `background` class, these categories can be split into three main groups (see Figure 3.3): (1) 14 text classes (*e.g.*, `title`, `code`, and `footnote`); (2) 6 figure classes (*e.g.*, `plot`, `map`, and `logo`); (3) 4 structural classes (*e.g.*, `table` and `enumeration`);

Annotation process. We provide fine-grained annotations at *pixel-level* for 25 classes. This level of granularity is necessary for this type of data as many of these semantic classes represent *fine* entities like `page number` or `date`. Another option are bounding box annotations, however, they are not adequate as many of the background pixels will be wrongly annotated with the foreground class which will significantly increase the noise in the training data. Additionally, we notice that there is a high region overlap between the categories, *e.g.*, some tables contain enumerations inside the cells, while a subset of `diagrams` contain `text`, `drawing`, or even `tables` in its nodes (Figure 3.2 left).

Hence, we use multi-label annotations where a pixel may belong to multiple overlapping categories at the same time. To that end, we develop an annotation tool based on the one used in [Caesar et al., 2018]. Each slide image is first divided into fine superpixels using Simple Linear Iterative Clustering (SLIC) [Achanta et al., 2012]. SLIC leverages both visual and spatial information of pixels in a weighted distance measure that controls the size and compactness of the superpixels. Since we have very fine structures in our data, we set the superpixels extracted by SLIC to be

Number of overlapping classes	1	2	3	4	5
Number of pixels in overlap	130M	65M	3.6M	180k	2k

Table 3.1: Number of pixels with an associated per-class label overlap from one to five (*e.g.*, 65M pixels have two associated classes)

relatively small (13000 superpixels per image). Finally, the output of SLIC is used by the annotators to classify each of the superpixels into the 25 classes. With this strategy the annotators can label an entire region comprising multiple pixels at a time.

To show the performance of this annotation method, we annotate 100 images by three different annotators in a similar manner as [Caesar et al., 2018; Shen et al., 2017]. We obtain a mean pairwise label agreement of 77% over the annotators, similar to COCO-Stuff [Caesar et al., 2018] with an agreement of 74% and ADE-20K [Shen et al., 2017] with 67%.

3.1.2 Per-pixel class overlap

Another aspect of our dataset is the amount of class overlap per pixel, as this influences some of our metrics where a prediction is considered correct if and only if all classes of the pixels are classified correctly. Moreover, the high amount of class overlap per pixel justifies our original motivation for enabling multi-label per-pixel classification. Finally, strong overlapping classes indicates the overall difficulty of this dataset portraying a major challenge for our neural networks.

Table 3.1 depicts the number of pixels in the entire dataset for different amount of per-pixel overlap. As we see, 130M pixels (excluding the background class) have no overlap and, thus, in these cases one-class segmentation would have sufficed. However, in around 70M pixels this would have failed, as we have an overlap of at least two categories. By far the highest multi-class labels is with an overlap of two, where we have 65M pixels, around a third of non-empty pixels in our dataset. This is mostly caused by the `text` class that has very frequently an overlap with structure classes, *e.g.*, enumerations and tables. The maximum overlap that SPaSe has is five which we have in around 2k pixels.

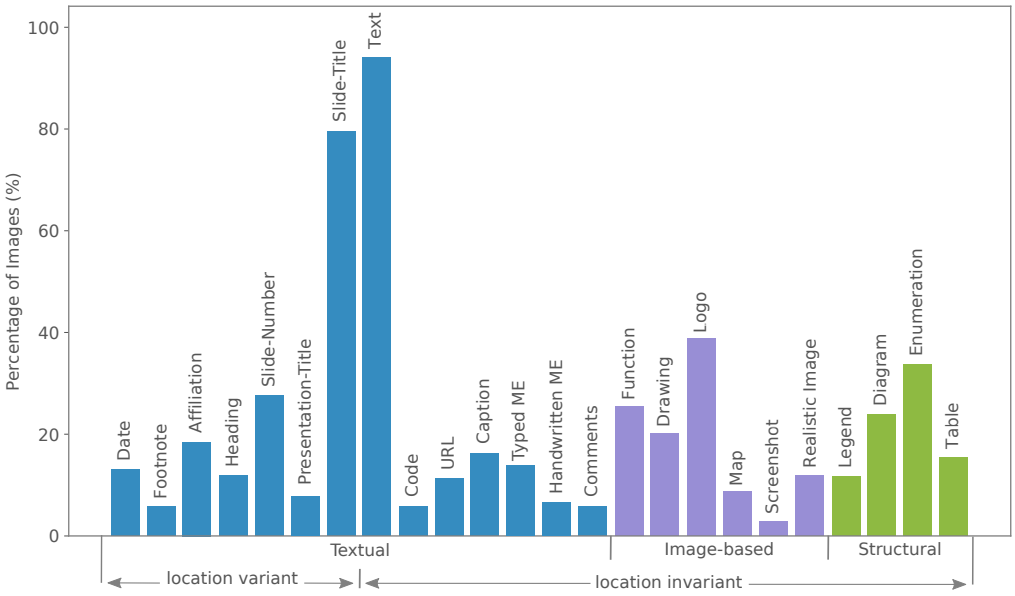


Figure 3.3: Overview of the class distribution in our dataset. In this figure, we split the classes by their main categories, *i.e.*, textual (blue), image-based (purple), and structural (green) element types.

3.1.3 Per-class image distribution

In Figure 3.3, we visualize the distribution of the classes over the page images in our dataset. Most of the classes appear in 10% to 30% of the slides showing a relatively balanced distribution. Two of the most prominent classes in more than 75% of the data are `text` and `slide-title`. This is not surprising as several of our classes are simultaneously text-classes, *e.g.*, `date`, `heading`, `slide-number`. Among the least frequent classes (in less than 6% of the data) are `footnote`, `code`, and `screenshot`. In case of the structural classes, the most represented class is the `enumeration` category, while the `logo` class is the most frequent in the image-based main category. Additionally, in around 12.3% of the data we encounter handwritten text (*i.e.*, `comments` and `handwritten mathematical expression`). This is, for example, one of the unique properties of slides in comparison to other document formats like papers or magazines where text is always typed.

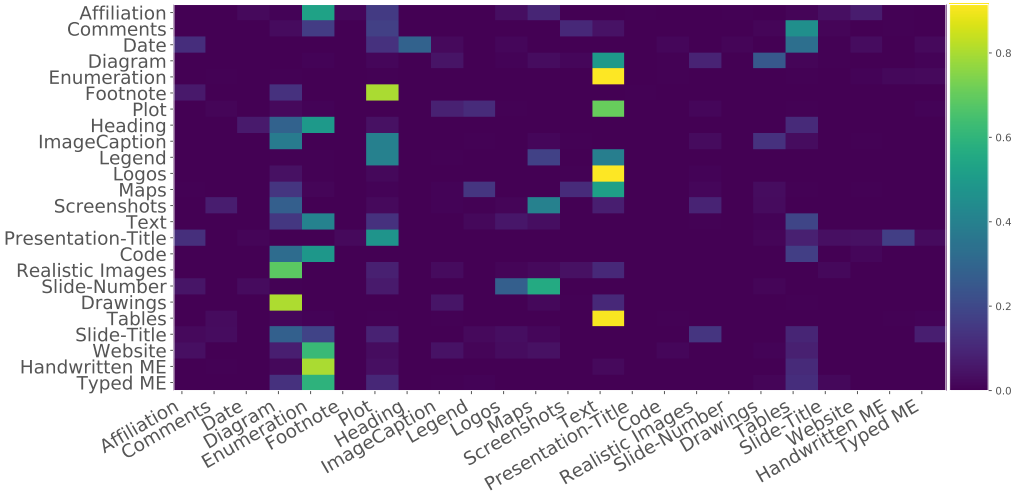
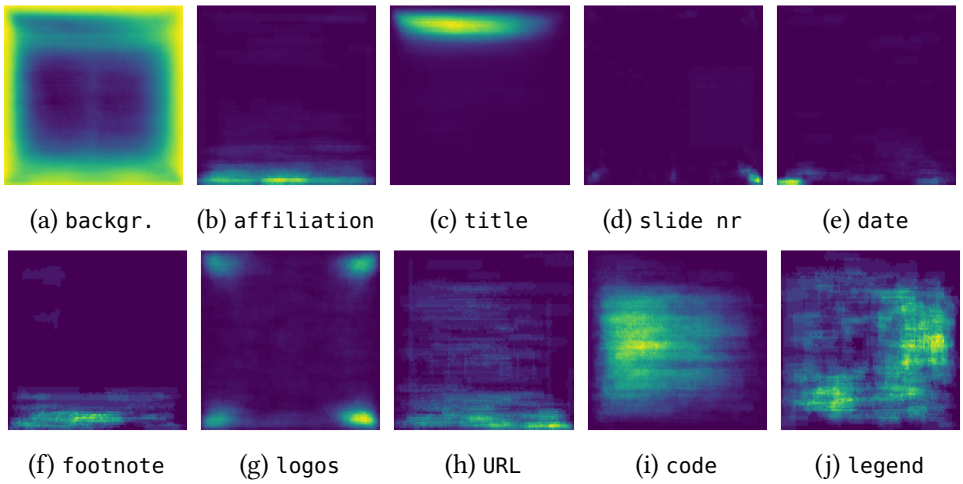


Figure 3.4: Pixel frequency of the pair-wise class overlap normalized over each semantic class (*i.e.*, x-axis).

3.1.4 Overlapping regions

Next, we analyze the frequency of the pairwise occurrences of different classes in our dataset (Figure 3.4). Note that in this figure, we only show a subset of the classes that have a strong overlap among each other.

Not surprisingly, we have a strong overlap with the `text` class, since especially structure classes like `diagrams` and `tables` contain text. Another category with a strong class-wise intersection is the `enumeration` component type, which has an overlap with text-based entities, *e.g.*, `URL` and `code`. Interestingly, we have more `legends` assigned to `plots` than to `maps`. We also notice that `handwritten text` (comments) are often written on `plots` and `tables` to provide additional information for these complex illustrations. Furthermore, the dataset encompasses many `enumerations` which comprise `typed mathematical expressions` that are usually included in math homework slides, *i.e.*, a list of exercises containing some formulas. Natural images and drawings usually do not have an overlap, but drawings are more likely to appear within a diagram than in natural images.



(a) backgr. (b) affiliation (c) title (d) slide nr (e) date

(f) footnote (g) logos (h) URL (i) code (j) legend

Figure 3.5: Pixel-wise heat maps showing the location of different entities in the page (lighter indicates more frequent).

3.1.5 Location heat maps

In Figure 3.5, we show the occurrence frequency in the page of ten of our classes. We see that classes like `title`, `slide number`, and `logo` have a strong position prior. Especially, we have text that changes its semantic meaning dependent on its *size* and *position* like `title`, `slide number`, and `footnote`. For example, numbers located in the page body are either `mathematical expressions` or just belong to the text body, while stand-alone text at the corners of the page is usually the `page number`. The same we have for slide `title` at the top and `footnote` at the bottom of the image. In the case of the `title` class, we also notice the larger font that is typically used and, thus, simply classifying the text at the top as `title` is not sufficient. The font size of the surrounding text is therefore also important to be able to classify slide titles. Interestingly, we see that the `legend` component are mostly positioned in the right and bottom side. Also, we are able to recognize `programming code` by the short line breaks that we see in most programming languages. Not surprisingly the background is mostly located at the borders of the page as we see in Figure 3.5a. Due to the positioning and the high frequency of the slide titles, the background appears slightly less at the top than at the bottom of the page.

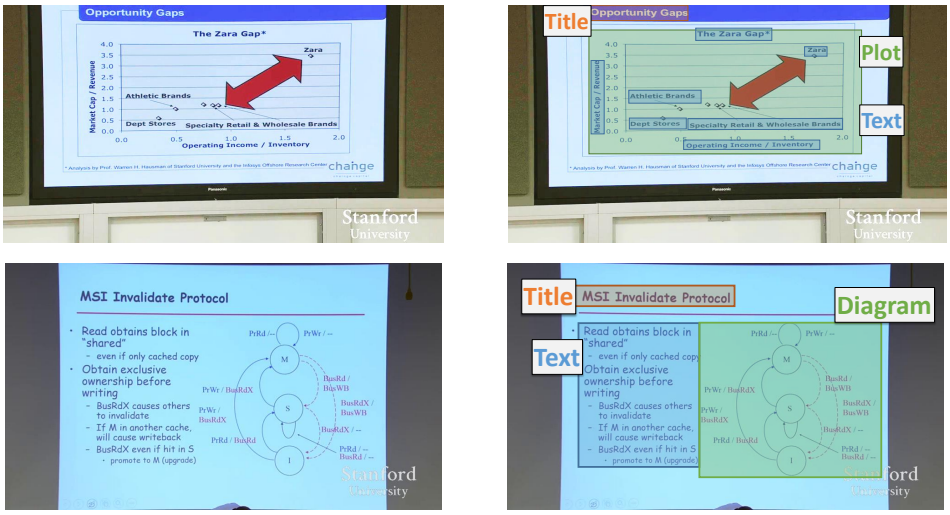


Figure 3.6: Example of slides captured in the wild (left) with the corresponding class annotations (right).

3.2 Slides captured during lectures

Presentation slides are highly useful documents for introducing a topic to an audience in an intuitive way. The importance of slides has grown rapidly from being a supplement to lecturers' speeches and printed papers to being the main pathway for dissemination of knowledge, *e.g.*, in university classes. A downside of this medium is the difficult automated analysis and information extraction, as slides are more visual and vary greatly in their structure, layout, and relations of the entities, which can become very complex. While understanding and automatically converting the knowledge in a digital form is more difficult for slides than for conventional textbooks, solving this problem would make information more accessible, having a substantial impact for people struggling with visual impairment or blindness.

As mentioned previously, page segmentation is an essential prerequisite step for most of document understanding tasks, which identifies distinct portions of the document and assigns them a semantic meaning. Problems in this task vary greatly in terms of the segment types: ranging from simple binarization which classifies a pixel to page or non-page; to *semantic* page segmentation with a high number of diverse content classes, such as `heading`, `title`, `URL`, and `table`.

While multiple works targeted the problem of page segmentation in contemporary documents in the past, the research of slide analysis has been sparse so far. Compared to previously published methods which focus on scientific articles [Clark and Divvala, 2016; Yang et al., 2017], magazines [Clausner et al., 2017], e-books [Tao et al., 2014] *etc.*, page segmentation of presentation slides captured in the wild encounters a far higher complexity of page structure and various possible visual classes (*e.g.*, drawings and handwritten text).

In the previous sections, we introduced the task of page segmentation on *digital* presentation slides, which were automatically extracted from PDFs. However, slide documents are not always offered to the viewers beforehand. Besides, the listener might not know, which of the slides is currently presented making such presentations very hard to follow for visually impaired people. It is unclear, how such methods perform in an uncontrolled setting, where the slide image does not originate from a pre-processed document, but, for example, from a display photo taken during a lecture. A model for segmenting such "slides in the wild" should be able to additionally handle problems related to natural images, such as noise, varying illumination conditions, and different rotation and scaling (see Figure 3.6).

Next, we tackle the problem of slide segmentation in the wild, where the slide does not originate from a digital document but is contained in a part of a *natural* image. Our overall application goal is a model, which allows a person to take a live photo of a slide, *e.g.*, from a display during a lecture, and then provides the associated semantic information of the captured page.

The *WiSe* dataset for slide ***Se***gmentation in the ***Wi***ld covers pixel-wise annotations for 25 different classes on 1300 pages captured during lectures. Since a pixel can belong to multiple independent classes (such an example is shown in Figure 3.6 with an overlap between `text` and `plot`), we allow overlapping annotations, *i.e.*, a pixel can be associated to multiple labels as in the case of SPaSe.

Finally, we employ several neural architectures on our proposed WiSe benchmark which were originally used for semantic segmentation and report promising results for unconstrained slide segmentation (Section 3.6).

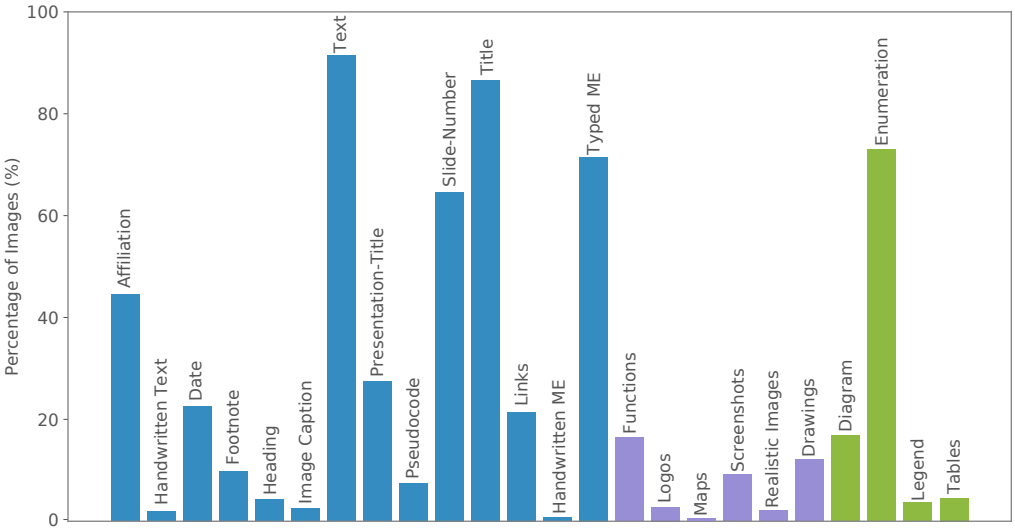


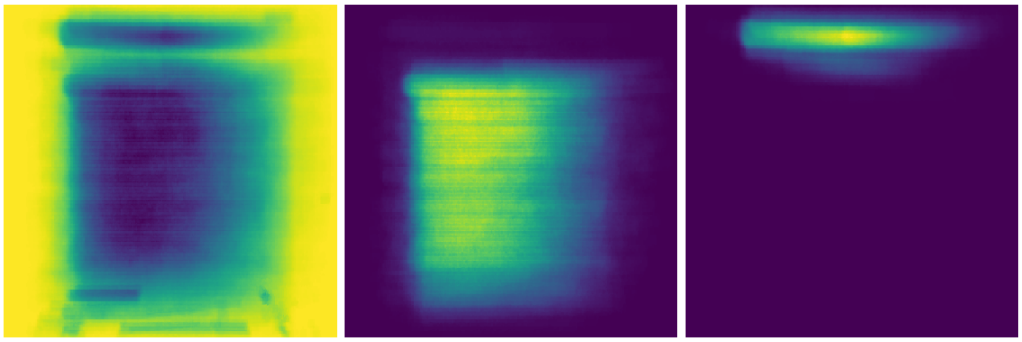
Figure 3.7: Overview of the class distribution in the WiSe dataset. We group the classes into three categories: text-based, graphical, and structural regions.

3.2.1 Annotation protocol

In total, we annotated 1300 slides which we split into 300 images for testing, 100 for validation, and the remaining 900 slides are kept for training. To help reduce redundancy and, thus, possible over-fitting of our models, we made sure that there are no identical slides in the dataset. We extracted the slides in our dataset from the publicly available Class-X lecture dataset provided by Araujo *et al.* [Araujo et al., 2015], which we manually label using pixel-level annotations for our 25 previously selected classes. As stated previously, fine-grained annotations are especially important in case of slides, as we have graphics and structures that often do not have a quadratic shape (see examples in Figure 3.6). We used the same annotation tool as for labeling SPaSe and generate a high number of superpixels.

3.2.2 Analysis of the semantic classes

In Figure 3.7, we show the distribution of the different classes in WiSe and group them into textual, graphical, and structure-based regions. As we see, the `text` label is by far the most prominent class with more than 90% of the available slides contain



(a) background (b) enumeration (c) title

Figure 3.8: Distribution of three of our classes on the page. As for SPaSe, some classes are strongly location variant.

some type of text. The text label is followed by more specific text classes like title, slide number, and printed mathematical expressions which occur in over 60% of WiSe. Finally, classes like maps and handwritten mathematical expressions are not very common, but still are present in some slides.

3.2.3 Location heat maps

In comparison to semantic segmentation on natural images where the class of an object does not change dependent on the location in the image, in case of pages, some classes are strongly location variant. While in SPaSe the slides were completely centered in the page, in slides captured during lectures, the slides suffer distortions, rotations, tilts, and translations (Figure 3.6).

Even though photos of slides captured using a camera have such kind of strong transformations, we still see that the slides are mostly centered in the image, as some components are clustered in some regions in the figure (see Figure 3.8). For instance, large text on top of the page is very frequently the title of the slide, while document components like diagrams and enumerations are located in the center of the page, whereas footnotes are mostly at the bottom. In comparison, the background class can be seen in the first image showing that most content is localized in the center with a small interruption due to the title class at the top of the page.

Dataset	#Pages	#Txt	#Img	Wild	Pixw.	Overl.
Magazines						
RDCL17 [Clausner et al., 2017]	77	10	2	✗	✓	✗
E-Books						
CM [Tao et al., 2014]	244	12	3	✗	✓	✗
Papers						
CS-150 [Clark and Divvala, 2016]	150	2	2	✗	✗	✗
DSSE-200 [Yang et al., 2017]	200	2	3	✗	✗	✓
SectLabel [Luong et al., 2012]	347	20	3	✗	✗	✓
CS-Large [Clark and Divvala, 2016]	3100	2	2	✗	✗	✗
Street-View						
SVT [Wang and Belongie, 2010]	350	1	0	✓	✗	✓
Presentation Slides						
SPaSe (Ours)	2000	14	10	✗	✓	✓
WiSe (Ours)	1335	14	10	✓	✓	✓

Table 3.2: Overview of datasets for page segmentation grouped by document type. We show the number of text, graphical, and structural classes for each dataset and if the images were captured by a camera (*i.e.*, are captured in the 'wild'). We point out which datasets contain pixel-wise segmentation and have overlapping annotations (*i.e.*, enabling the possibility of a pixel belonging to multiple classes).

3.3 Comparison between datasets

In Table 3.2, we compare our datasets on digital slides and slides captured in the 'wild' with other publicly available datasets used for page segmentation. We show the document type covered in each dataset with the corresponding number of pages and the number of classes for the text, images, and structural elements (*e.g.*, `tables`, `lists`). While CS-Large has the largest number of pages, it also has the lowest number of classes. It provides annotations for only four classes: figures, tables, image captions, and text. In comparison, our datasets have the second highest number of pages, while providing pixel-wise annotations of more than 20 classes. Moreover, a key feature of our dataset in comparison to the others is that we provide overlapping class annotations of pages. The closest dataset in this regard is the semantic segmentation dataset ADE-20k [Shen et al., 2017], which allows an

overlap between maximum of two subsets of classes (objects and parts of objects). In comparison to them, we allow any possible combination of classes, especially since multiple combination (*i.e.*, more than two) of region classes can occur.

3.4 Methods for page segmentation

Baselines. First, we evaluate two baselines on the slide segmentation task: (1) *Uniform*, which picks each class with equal probability and (2) *Background*, which chooses the most frequent category as the output class, *i.e.*, the `background` class.

FCN-8s [Long et al., 2015]. The Fully Convolutional Network (FCN) is a neural network for semantic segmentation that comprises an encoder of several convolutional layers [Simonyan and Zisserman, 2015] pre-trained on ImageNet [Deng et al., 2009] (*i.e.*, the features are continuously down-sampled by maxpooling layers), and a decoder with multiple upscale layers for resizing the features to the initial image size.

FRRN [Pohlen et al., 2017]. The Fully Resolutional ResNet (FRRN) is a network that leverages two processing streams to capture both the semantic information in the image and fine-grained contours. While the first stream processes the input image with progressively larger receptive fields, the second stream leverages residual connections to keep the feature maps at a high resolution.

DeepLab [Chen et al., 2018]. DeepLab contains multiple dilated convolution layers [Holschneider et al., 1990] to enlarge the receptive field, but at the same time keep the feature dimensions. The pyramid pooling then extracts features at multiple scales and, thus, captures small objects and image context.

Learning setup. For the multi-label page segmentation, we require for our models to infer *multiple* classes per pixel. Thus, we replace the softmax output layer which was used throughout the previously presented neural networks with a sigmoid activation function and train these models using binary cross entropy loss. We train these neural networks for a total of 50 epochs using the same optimizers as they

were originally used for image segmentation in [Chen et al., 2018; Long et al., 2015; Pohlen et al., 2017]. Then, we select the model with the highest mean intersection over union over the validation set for the final evaluation.

3.5 Evaluation metrics

As noted earlier, our datasets encompass *multi-label, pixel-wise* annotations, contrary to the conventional single-label version for semantic segmentation in natural images. Thus, this setup is different than the common segmentation problem where classes are assumed to be mutually exclusive [Lin et al., 2014]. Hence, besides the popular mean Intersection over Union (mIoU) metric [Everingham et al., 2015], we define three additional evaluation metrics for multi-label segmentation, namely: mean balanced Accuracy (mbAcc), pixel Accuracy (pAcc), and pixel Intersection over Union (pIoU). Let h, w, and c be the height, width of the image, and the number of categories, respectively. The variables $L^k, P^k \in \{0,1\}^{h \times w \times c}$ depict the ground truth labels and predictions, respectively, for a sample k in our test data.

Pixel Accuracy (pAcc). We define the number of correctly labeled values for a pixel (i,j) in image k as: $T^k_{i,j} := \sum_l [L^k_{i,j,l} \doteq P^k_{i,j,l}]$, where $l \in \{1,\dots,c\}$. Then, the pixel accuracy for image k is defined as the percentage of pixels which have an exact match with all ground truth labels of each pixel:

$$\text{pAcc}^k := \frac{1}{h \cdot w} \sum_{i,j} [T^k_{i,j} \doteq c] \tag{3.1}$$

where $i \in \{1,\dots,h\}$ and $j \in \{1,\dots,w\}$. We denote with $[x \doteq y]$ the indicator function that generates a one if the condition inside the brackets is true, and zero otherwise. In this particular case, the condition in the indicator function verifies if the two entities x and y are equal to each other. While this metric gives us an idea of the pixel-wise segmentation accuracy it has the following drawbacks: (1) the metric harshly punishes partially correct segmentation by assigning a zero accuracy for pixels if one label is misclassified and (2) the metric might be biased towards the most frequent annotations in case of unbalanced distribution of classes.

Pixel Intersection over Union (pIoU). To tackle the first drawback, we define the pIoU metric, which softens the weight given for incorrect classifications:

$$\text{pIoU}^k := \frac{1}{h \cdot w} \sum_{i,j} \frac{N^k_{i,j}}{S^k_{i,j} - N^k_{i,j}} \tag{3.2}$$

where $S^k_{i,j} := \sum_l (L^k_{i,j,l} + P^k_{i,j,l})$ and $N^k_{i,j} := \sum_l \min([L^k_{i,j,l} \doteq 1], [P^k_{i,j,l} \doteq 1])$. This metric is conceptually similar to the mIoU but for multi-label predictions.

Mean balanced Accuracy (mbAcc). The mbAcc tackles the second drawback by using a class-based weighted accuracy measure (*i.e.*, the balanced accuracy):

$$\text{bAcc}^k_c := \sum_{\ell \in \{0,1\}} (1 - \alpha^c_\ell) \cdot \sum_{i,j} \min([L^{k,c}_{i,j} \doteq \ell], [L^{k,c}_{i,j} \doteq P^{k,c}_{i,j}]) \tag{3.3}$$

where the weights $\alpha^c_\ell \in [0, 1]$ are proportional to the number of pixels labeled with class c and pixel label ℓ. More precisely, α^c_ℓ is equal to percentage of $L^k_{i,j}$ equal to ℓ:

$$\alpha^c_\ell := \frac{1}{t} \cdot \sum_{k,i,j} [L^k_{i,j} \doteq \ell] \tag{3.4}$$

where t is the number of pixels in the test split. Finally, the mean balanced accuracy is obtained by averaging the balanced accuracy across all classes:

$$\text{mbAcc}^k := \frac{1}{c} \sum_l \text{bAcc}^k_l \tag{3.5}$$

where l iterates over all classes and c is the total number of classes.

In comparison to our proposed evaluation metrics for *overlapping* segmentation, non-overlapping semantic segmentation employs the following metrics: the mean intersection over union, the mean accuracy (*i.e.*, the accuracy balanced by the classes), and the pixel accuracy (*i.e.*, accuracy for each pixel in the image). While we cannot use the mean and pixel accuracy for the multi-label classification setup, we further leverage the popular mean intersection over union as our primary metric.

Approach	Subsaml.	mIoU	pAcc	pIOU	mbAcc
Baseline Methods					
Uniform	–	1.1	3.4	4.0	50.0
Background	–	2.5	61.6	61.6	50.0
Neural Networks					
FCN-8s	×8	20.0	66.2	73.5	62.0
FRRN-A	×4	28.4	69.5	73.8	67.0
FRRN-B	×2	30.9	71.2	75.3	68.5
FRRN-B + Location	×2	33.2	73.4	77.2	70.1
DeepLab	×2	34.1	76.5	80.3	71.2
DeepLab + Location	×2	**35.8**	**77.4**	**81.2**	**72.6**

Table 3.3: Test results of the multi-label page segmentation task on our SPaSe dataset. We provide the performance based on the four previously introduced evaluation metrics for overlapping segmentation.

3.6 Evaluation

This section shows the performance of the previously defined models on different document analysis tasks. The problems that we consider on our datasets range from simple binarization, which entails in classifying a pixel into foreground or background, to the high-level task of categorizing the pixels into *25 semantic classes*. Nonetheless, all the tasks that we consider are fine-grained localization problems (*i.e.*, classifying each pixel into a set of discrete classes).

3.6.1 Semantic slide segmentation

Digital slides. In this experiment, we measure the performance of the baselines and networks in pixel-wise multi-label slide segmentation. That is, each pixel is classified into one or multiple classes of the 25 categories defined in our dataset.

As noted previously, segmentation approaches have the conundrum of (1) sub-sampling the input enlarging the receptive field for a high-level view of the objects and (2) keeping a small receptive field and being able to segment fine-grained structures. For example, the FCN-8s network reverts the down-sampling of the

Approach	Subsampl.	mIOU	pAcc	pIOU	mbAcc
Baseline Methods					
Uniform	–	0.2	0.0	0.0	50.0
Background	–	3.0	76.1	76.1	50.0
Neural Networks					
FCN-8s	×8	18.3	81.7	59.8	59.8
DeepLab	×2	35.8	88.3	**90.4**	72.2
DeepLab+Location	×2	**37.2**	**88.5**	**90.4**	**72.8**

Table 3.4: Results of the semantic segmentation on the test set of WiSe. We group our approaches into baseline methods and deep learning approaches.

feature maps through so-called upscore layers which, however, have difficulty in capturing fine-grained structures. FRRN maintains the fine-grained information by leveraging two streams: one for a large receptive field and the other one for fine-grained prediction. DeepLab leverages dilated convolutions to increase the receptive field while maintaining the size of the feature maps.

We first evaluate the baseline methods on SPaSe which we defined in the first section (Table 3.3), while in the second part we show the performance of the deep neural networks. We see that the baselines perform by far worse than the deep learning models as the neural architectures improve over the baselines between 17.5% (FCN) and 33.3% (DeepLab) in terms of mIoU. Both FRRN versions were able to outperform the mIoU achieved by FNC-8s. The deeper FRRN-B increased the mIoU by 2% in comparison to the shallower FRRN-A. On the other hand, DeepLab achieved the highest performance with 34.1%, further gaining over FRRN-B an additional 3%. Finally, we notice that the difference between FRRN-B and DeepLab in mIoU is smaller than in pIoU and pAcc. This shows that the gain was mostly influenced by improvements to the most frequent classes. In comparison, we notice that FRRN-A's performance boost over FCN-8s is mainly due to better segmentation of the less frequent classes as depicted by the higher gain in mbAcc compared to pAcc.

Since convolution layers are partially translation invariant, our models cannot use the location information, which is important in our *location dependent classes*. Thus, we tackle this shortcoming by concatenating a 2-channel map (x and y

Approach	Subsampling	mIoU	pAcc	mAcc
Baseline Methods				
Uniform	–	29.8	50.0	50.0
Background	–	41.8	83.7	50.0
Neural Networks				
FCN-8s	×8	80.7	93.7	90.4
FRRN-A	×4	80.4	93.7	88.9
FRRN-B	×2	**83.2**	**94.6**	**91.7**
DeepLab	×2	82.6	94.5	90.5

Table 3.5: Performance of the text segmentation task on the test set of SPaSe. We show the results for different evaluation metrics previously used for single-label semantic segmentation on natural images.

positional embeddings) to our input image with the location of each pixel. This simple modification improved the DeepLab model to an mIoU of 35.8% and FRRN-B to 33.2% (see DeepLab+Loc and FRRN-B+Loc in Table 3.3).

Slides captured during lectures. The most difficult of the proposed settings is the semantic slide segmentation where our models aim to assign each pixel to one or multiple labels from the 25 available classes. As for the SPaSe dataset, we implement two baseline methods to show possible biases in the data, *i.e.*, we report the random performance and the accuracies for the most common class.

As can be seen in Table 3.4, the baselines perform poorly especially in mIoU with a performance of 3% and 0.2%, respectively. By using as input solely the slide image in RGB format, DeepLab is able to achieve an accuracy of 35.8% improving the baseline methods by over 30%. Since some components in the page are location variant (*e.g.*, title, footnote) and fully convolutional networks are translation invariant, we conducted an experiment where we obtain as input additionally to the RGB values of each pixel its location in the page (DeepLab+Location). This new setting is able to achieve an improvement over the RGB-only model of 1.4% in mIoU.

Approach	Subsampling	mIoU	pAcc	mAcc
Baseline Methods				
Uniform	–	29.8	50.0	50.0
Background	–	42.4	84.9	50.0
Neural Networks				
FCN-8s	×8	83.7	95.2	91.4
DeepLab	×2	**84.4**	**95.5**	**91.3**

Table 3.6: Text segmentation results on the WiSe dataset on slides captured during lectures. We group the methods in baselines (uniform and background) and in neural networks, where we show the results of two segmentation architectures.

3.6.2 Text segmentation

Next, we explore other popular page segmentation tasks on our datasets. We first show the performance of our models for the text segmentation task which aims to assign to each pixel one of two classes: text and non-text.

Digital slides. In Table 3.5, we report the results of the models on the SPaSe test set for the text segmentation problem. Note that we consider non-text the entirety of the page which was not assigned a text-label (*i.e.*, our overall task is binary) as well as we consider text not only the text-classes but also regions belonging to paragraphs and, thus, labeled with the `text` class. We therefore leverage popular metrics for non-overlapping semantic segmentation [Long et al., 2015] (*i.e.*, as for conventional segmentation of natural images) and report the mean Intersection Over Union (mIoU), pixel Accuracy (pAcc) – accuracy over all pixels in the image, and mean Accuracy (mAcc) that calculates the mean accuracy over all classes.

While the baselines have a low performance of only 41.8% for the aimed two-class segmentation task, the deep learning models improve results by more than 40% and 30%, respectively. We see in Table 3.5 that FRRN-B has the best results on the text localization task improving its shallower counterpart by almost 3%. The reason for its strong performance for text segmentation is probably linked to the residual

Approach	Subsampling	mIoU	pAcc	mAcc
Baseline Methods				
Uniform	–	34.5	55.8	50.0
Background	–	30.8	61.6	50.0
Neural Networks				
FCN-8s	×8	76.3	87.2	86.6
DeepLab	×2	80.4	89.6	89.3
FRRN-A	×4	80.3	89.6	89.3
FRRN-B	×2	**81.0**	**90.0**	**89.4**

Table 3.7: Binarization results on the test set of the SPaSe dataset for digital slide page segmentation. This task entails the assignment for each pixel on of two available classes, *i.e.*, either the foreground- or the background-class.

stream structure as this keeps the feature maps to a constant size. Thus, it is able to segment fine structured objects like text. In comparison, DeepLab uses a pyramid of different sizes of receptive fields leading to a worse text segmentation result.

Slides during lectures. Table 3.6 illustrates the results of our baseline methods and the deep learning networks on the text task of the WiSe benchmark on presentation slides captured in the wild. As mentioned previously, for text segmentation each pixel is associated to a *single* class and, thus, this task is a *single-label, binary* classification problem. Not surprisingly, since in this setup the uniform baseline only chooses between two classes, it is able to achieve a pixel and mean Accuracy of 50%. The prior baseline, that always predicts the most frequent class (*i.e.*, `background`), obtained an overall pAcc of 83% and mIoU of 42.4%. The overall pixel accuracy of 83% shows the unbalanced nature of our slides, as by far more pixels belong to the background than are part of text. While FCN is able to improve the baseline methods by over 40%, DeepLab achieves the highest performance on all metrics, where it shows a mIoU of 84.4% and pAcc of 95.5%.

Approach	Subsampling	mIoU	pAcc	mAcc
Baseline Methods				
Uniform	–	32.9	50.0	50.0
Background	–	38.1	76.1	50.0
Neural Networks				
FCN-8s	×8	84.3	93.7	91.4
DeepLab	×2	**86.3**	**94.5**	**92.8**

Table 3.8: Binarization results on the test set of WiSe, where we aim to classify each pixel into two classes: page or non-page.

3.6.3 Binarization

Digital slides. In this section, we further address the problem of *page binarization of presentation slides*, which aims at labeling each pixel with one of two classes: foreground or background. We define *foreground of a page* all pixels that are associated to one of our 25 document categories. To that end, we train the models in the same way as on the text segmentation task, *i.e.*, we minimize the cross entropy loss with sigmoid normalization on a single output neuron (note that we do not have any overlapping regions as in the case of semantic page segmentation).

Table 3.7 shows the results of the baselines and neural networks on the test set of SPaSe. While the two baseline methods show a very low performance of a mean IoU of less than 35%, the deep networks improved performance by over 40%, achieving more than 76% in mIoU. Moreover, the more recent DeepLab model comprising dilated convolutions was able to increase accuracy in comparison to the fully convolutional FCN-8s network. We also notice a divergence in the amount of improvement in the different metrics, *e.g.*, in mIoU the deeper networks show a stronger improvement than in case of the pixel accuracy. This is, however, not surprising, as the mIoU averages the performance over *all classes*, while the pixel accuracy is stronger biased towards unbalanced data. The best performance was achieved by the FRRN-B network, which employs residual connections for a better gradient flow, obtaining an overall mIoU of 81% and a pixel accuracy of 90% on the binarization task.

Slides during lectures. Table 3.8 shows the results of the baselines and the segmentation networks on the test set of the WiSe dataset for slides collected during lectures. As for SPaSe, we leverage as evaluation metrics ones that are usually employed for single-label semantic segmentation on natural images.

The background-only method improved performance by around 5% in mIoU over the random baseline, which uniformly selects between the fore- and background class. The pixel accuracy of the background-only baseline on WiSe is higher than for SPaSe showing that the slides captured during lectures encompass more background. This is, however, not surprising as WiSe contains additional elements such as walls or the lecturer himself in the image that are not part of the slide.

As in case of SPaSe, DeepLab improved performance of around 2% on all metrics over the shallower fully convolutional FCN-8s network. Interestingly, the models on WiSe achieved a higher performance than on the digital slides of SPaSe on both the text segmentation and binarization task. A possible reason behind these results is that the slides from SPaSe were selected by a human to comprise more complex pages, while in WiSe the slides were chosen randomly from lecture videos (*i.e.*, WiSe includes simpler slides). We also notice that the difference between the datasets is higher in the binarization problem (*i.e.*, 5%) than for text segmentation (*i.e.*, around 1%). This indicates that page segmentation is an easier task on slides captured in the wild, while the text segmentation problem has a similar difficulty level for both setups (digital pages and pages captured during lectures).

Finally, the subsampling rate is a hyper-parameter expressing the down-sampling rate of the input images through the network caused by convolutional and pooling layers with strides greater than one. The results shows that this parameter is a major factor on our page analysis problems, as we show that models with lower such values achieve a higher overall accuracy. This indicates that the pages encompass fine-grained structures that require a precise boundary detection, while a higher receptive field (achieved through a strong downsampling of the input image) shows a lower importance. Next, we show qualitative results of the best performing DeepLab model on the semantic page segmentation task.

Image Ground Truth Prediction

Figure 3.9: Examples of slides along with ground truth annotations and predictions produced by our best model for multi-label page segmentation.

3.7 Qualitative results

Digital slides. In Figure 3.9, we visualize some example slides (left) with both ground truth annotations (center) and predictions (right) on the multi-label task. We see that DeepLab was able to recognize difficult classes like `plots`, `tables`, `diagrams`, and `programming code` correctly. Even though DeepLab uses a small downsampling rate, it still has some difficulties to get the exact borders of fine structures like the borders between paragraphs (*e.g.*, bottom example in Figure 3.9).

Problematic is also the title in the first example (top row), which comprises of multiple lines: while the first line was classified correctly to the `title`-class, the next two lines were partially assigned to the `text`-class. The reason behind this is probably the scarce number of slides in our dataset which include multi-line

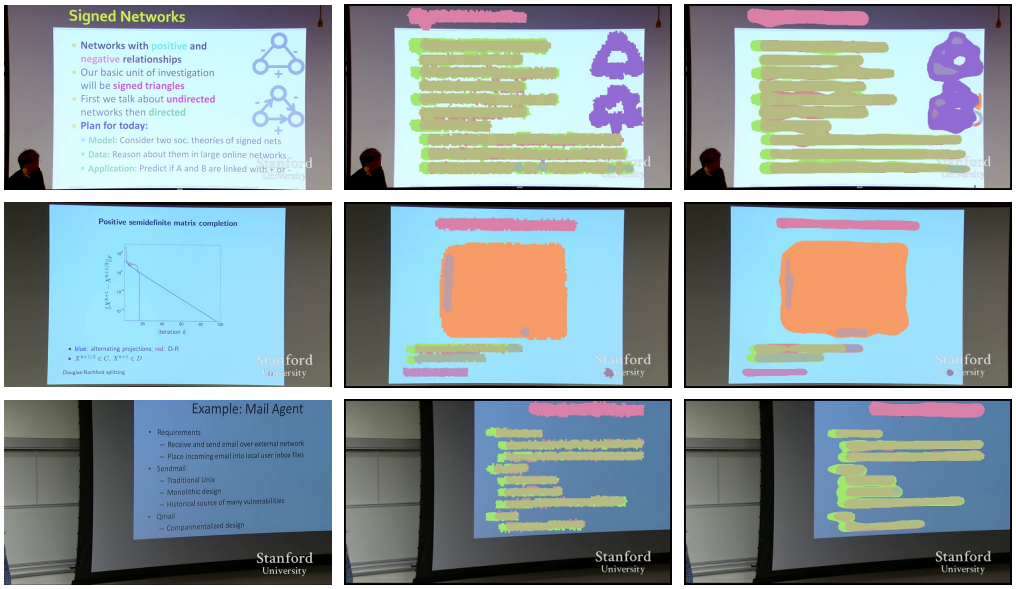


Image Ground Truth Prediction

Figure 3.10: Example of pixel-wise labeled slides: Raw images from lectures collected by [Araujo et al., 2015] (left) extended with corresponding manual annotations in our WiSe dataset (center), and the predictions of DeepLab (right).

titles. Overall, the DeepLab network was able to distinguish difficult classes and the boundary of the component localization shows a better smoothness at the region boundaries than the ground truth.

Slides during lectures. In Figure 3.10, we show examples of slides captured during lectures (left) with their corresponding manually annotated segmentation (center), and segmentation masks inferred by the DeepLab neural architecture (right). As we see, the model is able to correctly predict difficult classes like `enumerations` and `diagrams`. Even though the model shows some problems to recognize the correct edges of the current segment (*e.g.*, the contours of the diagram) it is able to produce a smoother boundary than the manual annotation (*e.g.*, `text` and `enumeration` labels). Moreover, the model recognized the document components in frontal slides (*e.g.*, the images at the top and center) as well as in tilted and cropped ones (bottom). Overall, the network correctly localized different components on complex slides captured on diverse view-points and various illumination conditions.

3.8 Summary and discussion

In this chapter, we addressed the problem of semantic page segmentation and introduced two novel datasets comprising fine-grained annotations of 2000 *digital slides* and 1330 *slides captured during lectures*. We include in total pixel-wise labels of 25 semantic classes: 4 structural, 6 image-based, and 14 textual components. Additionally, we provide a thorough analysis of the data properties and its unique features: *e.g.*, the multi-class characteristic of the pixels where a pixel is associated to multiple semantic categories *simultaneously*. Moreover, we show that our classes are partially location variant and, thus, include pixel spatial location leading to improved segmentation results. On our datasets, we establish strong baselines and demonstrate the suitability of our dataset for developing deep learning model from scratch. Finally, we show that the deep learning models are able to achieve a strong performance on all three tasks: text segmentation, binarization, and semantic segmentation improving baseline methods by over 30% in mIoU.

4 Figure Question Answering

In this thesis, we aim to analyze multi-modal content from learning materials in a question answering setup. A large amount of this data comprises figures which range from synthetically-generated images to complex diagrams (Chapter 3). While most previous methods focus on natural images with flat questions (*i.e.*, that require a single reasoning step for inferring the answer) approaches for highly-structured content were only recently considered for the first time. In this chapter, we address the Visual Question Answering (VQA) task and introduce a novel architecture aimed at capturing the interplay among individual objects in a graph representation of the image. To that end, our approach traverses the visual graph searching for information relevant for answering the current question. As not all graph components are relevant for the input query, we introduce the concept of a *question-based visual guide*, which constrains the potential solution space by learning an optimal traversal scheme. The final destination nodes alone are then used to produce the answer. We show that finding relevant semantic structures facilitates generalization to new tasks by introducing a novel problem of knowledge

transfer: training on one question type and answering questions from a different domain without any training data. Furthermore, we achieve state-of-the-art results for visual reasoning on multiple query types and diverse image and video datasets.

Contributions. We address the problem of answering questions on graphical content and introduce a novel neural path-based architecture, which traverses the visual graph using the question as guide. In comparison to other related path-based graph nets, our network is trained end-to-end via conventional back-propagation without employing any reinforcement learning techniques (*e.g.*, REINFORCE [Williams, 1992]). Moreover, we use the traversed paths to understand the reasoning behind the answering approach as well as to deduce mistakes produced by the model. Since the prediction model comprises a multi-head module, we decouple the model based on the query-type and show the task-level generalization capabilities of our model. Finally, we evaluate our proposed approach on several popular benchmarks aimed at high-level relational reasoning on which our network shows strong results.

Publications. This chapter is based on work published in [Haurilet et al., 2019d].

4.1 Task overview

Interpreting and answering subsequent questions about the semantic relationships of the complex and noisy environment is a key trait of our cognition. Extraordinary progress linked to the rise of deep learning in core vision tasks [Girshick, 2015; Krizhevsky et al., 2012; Long et al., 2015; Russakovsky et al., 2015] (*e.g.* object recognition) has created a solid basis for the new research direction of *higher-level* visual reasoning. Going beyond the conventional recognition, visual reasoning [Santoro et al., 2017; Yang et al., 2018a] decides about the necessary future actions [Johnson et al., 2017b], which is crucial for artificial intelligence applications. The compositional structure of our world makes this task especially hard, as merely recognizing individual building blocks at a lower level is not enough. Such models require precise *relational reasoning* about the entities present in the scene (`cat`, `desk`) and the interaction between each other (*e.g.*, `sleeping on`).

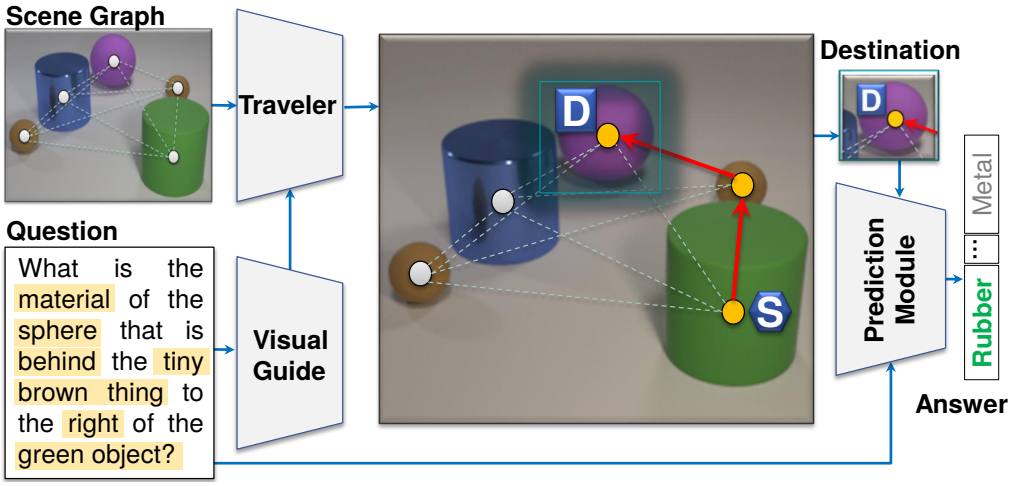


Figure 4.1: Example where the object interplay is crucial for inferring the correct answer and an overview of our proposed approach. The *visual guide* learns to give question-dependent directions to follow on the scene graph. The final answer is then produced solely from the embeddings of the reached *destination* nodes.

Visual reasoning tasks are often posed in the form of visual question answering [Johnson et al., 2017b; Santoro et al., 2017; Yang et al., 2018a], which lies in the intersection of vision and language and attempts to answer a specific question about the scene. Complex semantic associations between both language query and the visual scene entities (Figure 4.1) are characteristic for this task.

Despite the exceedingly structured nature of the visual information needed to answer open-ended questions, the majority of previous works focus on spatial feature maps obtained from a pre-trained CNN and further combined with an attention mechanism on parts of the image [Shih et al., 2016; Xu and Saenko, 2016; Yang et al., 2016]. While pre-trained CNNs offer excellent object embeddings, they face problems in relational reasoning about their large scale interactions. An excellent way to model such multi-step associations in an image are *scene graphs* [Xu et al., 2017], where the nodes represent the object and the connecting edges specify their relationship embeddings. We notice that even though the relations between objects are indispensable for the complete scene understanding, only a portion of the graph is relevant for answering a specific question. We therefore leverage the visual graph in a *selective* way through a question-dependent *visual guide*.

We aim at unifying graph-based inference with *question-specific visual guidance*, in order to identify paths with relevant information flow and present a new model for visual reasoning. Given an image-question pair, we first use the *visual guide* to create question-specific directions to follow in the graph. Next, the *graph traveler* traverses the visual graph guided by these directions and computes the probability distributions over the nodes being the final destination. We then compute the answer prediction solely from the expected destination node as visual representation for our *prediction module*. Finally, as in other QA networks, the prediction module uses this visual encoding of the destination and outputs a probability distribution of the set of possible answers or over the multiple choice answers.

While conventional graph-based models follow the graph refinement paradigm (*i.e.*, refined encodings of *all* components are used for the prediction), we maintain the original node representations, identify the key paths, and answer the question only from the expected final *destination* nodes.

We demonstrate the effectiveness of our model on three well-known datasets for different visual reasoning tasks: question-answering on video data (COG [Yang et al., 2018a]), compositional reasoning on 3D synthetic images (CLEVR [Johnson et al., 2017a]) as well as diagram question-answering with real-life figures extracted from textbooks (AI2D [Kembhavi et al., 2016]), which is much noisier while having less training data. Our model consistently outperforms previous approaches on the AI2D and COG benchmarks and shows strong performance on the CLEVR dataset.

As our model operates on semantic structures inside the scene graph, it has two beneficial properties: interpretability and generalization abilities to new tasks. An ablation study illustrates that we can easily shed light on the internal choices our model makes to produce the answer by following the final *soft path*. To evaluate the generalization capabilities, we propose a new task of knowledge transfer for VQA by splitting the training and test set based on question *types* (*e.g. query attributes* questions for training and *counting* for testing). Through knowledge obtained from training on one kind of questions, our neural network is able to derive the answers for queries for which type it has never seen before.

4.2 Visual reasoning via guided soft paths

We present a new model for visual reasoning that deals with the composite object relationships in the scene as a graph traversal problem. The challenge is that the space of potential paths in a visual graph is very large. When asked `*What is the material of the sphere that is to the left of the tiny brown thing behind the green object?*' (Figure 4.1), a human would immediately look for the green object, thereafter, at the tiny brown sphere and, then, select the sphere left of it. Likewise, our idea is to greatly constrain the solution space by learning the optimal graph traversal strategies based on question-specific decisions.

Conceptually, our visual reasoning model is composed of three main components: (1) the visual guide, (2) the graph traveler, and (3) the prediction module. The *visual guide* takes as input the question and produces direction embeddings. The *graph traveler* follows these directions and computes the *soft paths* – probability distributions over the nodes of being in the route to the nodes that include relevant information to produce the answer. The final decision is made by the *prediction module*, which exploits the found *destinations* as weights for the graph nodes and infers the final answer. We want to highlight, that the prediction module operates exclusively on the destination node representations, dismissing the preceding components of the paths. While the *visual guide* and the *prediction module* can be viewed as individual neural networks connected by the *graph traveler*, they are optimized jointly in an end-to-end training fashion.

We illustrate an overview of our model in Figure 4.4 where we show an example image extracted from the CLEVR dataset overlayed with its visual graph representation. The traveler uses both the visual graph and information extracted from the question-based guide to produce paths in the visual data. The final found destination is used to infer the answer (in this example: 'rubber').

Next, we give a general definition of the data structures we leverage throughout this chapter (Section 4.2.1), provide a mathematical foundation for computing the *soft paths* (Section 4.2.2), and we finally present our complete graph-based neural architecture for visual question answering (Section 4.2.3).

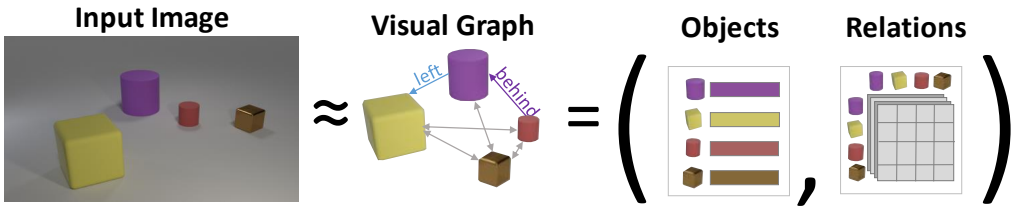


Figure 4.2: Example of a visual graph extracted from an image from the CLEVR dataset. We represent the input image as a graph, where the nodes denote object instances and the edges depict semantic relations between them. Each node represents an object instance in the scene via a feature vector (*i.e.*, the entirety of the nodes is a matrix), while the edges comprise a 3D tensor portraying all pair-wise object relations (*i.e.*, we assume that the graph is fully-connected).

4.2.1 Data structures

Next, we define the data structures that we require for our VQA model: the graph representing the input of our model, the discrete path as a set of nodes, and the soft path counterpart which enables our network to train in an end-to-end manner.

Graph. We call a *visual graph of size n* a data structure $\mathcal{G} := (V, E)$ comprising node and edge encodings that we define as follows:

1. $V \in \mathbb{R}^{n \times v}$ – v-dimensional features for each of the n visual nodes. The features can be one-hot vectors, *i.e.*, binary embedding encompassing a single one set at the location equal to the semantic class index. Another option is localizing different object instances using an object detector, *e.g.*, [Liu et al., 2016; Redmon et al., 2016]. Then, one can leverage the crops carved by the generated bounding boxes and extract feature vectors by passing the crops to a pre-trained CNN (*e.g.*, [He et al., 2016; Simonyan and Zisserman, 2015]).

2. $E \in \mathbb{R}^{n \times n \times e}$ – an e-dimensional edge representation for each pair of vertices (j, k) with $j, k \in \{1, \ldots, n\}$. The edge representation E can be a one-hot embedding of predicates (*e.g.*, `on top`, `holding`), which can be obtained using a graph generation method, *e.g.*, as in [Lu et al., 2016a], features extracted from a CNN on the image crop surrounding both objects, or by concatenating each node encoding pairwise extracted from the feature matrix V.

Path. We call $\boldsymbol{p}$ *a path of length* r *in graph* $\mathcal{G}$ a list of nodes with:

$$\boldsymbol{p}_i \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, r\} \tag{4.1}$$

We note that we do not require for the nodes in the path to be unique (*i.e.*, the path does not have to be cycle-free) and that we only consider *direct* paths. Moreover, this definition of path assumes a *discrete* assignment of each node in each reasoning step i. Thus, each vertex is traversed with absolute certainty in each reasoning step i (*i.e.*, no probabilities are assigned to the nodes in each reasoning step).

Soft path. To that end, we define the soft path paradigm, where we do not return *discrete* associations of each of a node with the path, but soften their inclusion. Formally, for each reasoning step i and vertex j in graph $\mathcal{G}$, we have an association score $\pi^i(j) \in [0, 1]$. As we aim to model a probability distribution, we require that the sum over all nodes in reasoning step i in the graph is one:

$$\sum_j \pi^i(j) = 1 \text{ where } j \in \{1, \ldots, n\} \tag{4.2}$$

Thus, a *soft path* is described by the output of the functions π^i, *i.e.*,

$$[\pi^1(j), \pi^2(j), \ldots, \pi^r(j)], \ \forall j \in \{1, \ldots, n\} \tag{4.3}$$

where we use $\pi^i : \{1, \ldots, n\} \rightarrow [0, 1]$ element-wise on each node.

Starting node. The starting node of a path is the node in the first reasoning step. In case of a soft path, it is carved by the confidence in step $i = 1$, *i.e.*, the distribution π^1.

Destination. While a destination is the last vertex in the discrete path, in case of soft paths, it is equal to the probability in the last reasoning step r.

4.2.2 Reaching the destinations

Our model builds upon the premise that by traversing the scene graph in a controlled way, we are able to identify the information relevant for the specific question. In case

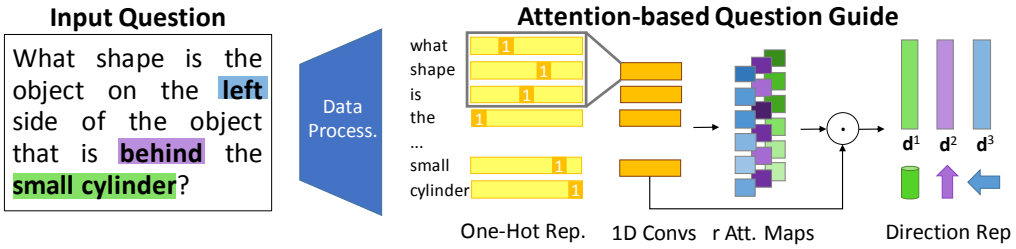


Figure 4.3: Overview of the question-based guide. We map the words in the question to a one-hot representation, which are then embedded using 1D convolutions with self-attention. We generate r attention maps, where r is the number of reasoning steps that our model produces. Thus, we obtain r vector representations, which should contain directional information for the traveler.

of discrete paths, we can compute the probability of the node v being a destination, which is equal to the sum of the probabilities of all paths ending in j:

$$\mathbb{P}(\boldsymbol{p}_r \doteq j) = \sum_{\boldsymbol{p}} \mathbb{P}(\boldsymbol{p}) \cdot [\boldsymbol{p}_r \doteq j] \tag{4.4}$$

with $[x \doteq y]$ we denote the indicator function depicting if x is equal to y and $\boldsymbol{p}_r$ represents the destination vertex of a discrete path. According to the marginalization rule, the probability of the path $\boldsymbol{p}$ is equal to:

$$\mathbb{P}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r) = \mathbb{P}(\boldsymbol{p}_1) \cdot \prod_{i=2}^{r} \mathbb{P}(\boldsymbol{p}_i | \boldsymbol{p}_{i-1}, \ldots, \boldsymbol{p}_1) \tag{4.5}$$

Our approach models a discrete Markov chain (*i.e.*, we assume the Markov property and consider discrete reasoning steps i) with the set of states equal to the nodes V in our graph $\mathcal{G}$. We obtain the probability of each path as:

$$\mathbb{P}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r) \approx \mathbb{P}(\boldsymbol{p}_1) \cdot \prod_{i=2}^{r} \mathbb{P}(\boldsymbol{p}_i | \boldsymbol{p}_{i-1}) \tag{4.6}$$

In case of $i = 1$, it is straight forward to compute the probability of the nodes in the path (*i.e.*, $P(\boldsymbol{p}_1)$) as one only needs to generate a single confidence value per node (*e.g.*, in a similar manner as conventional attention mechanisms produce a

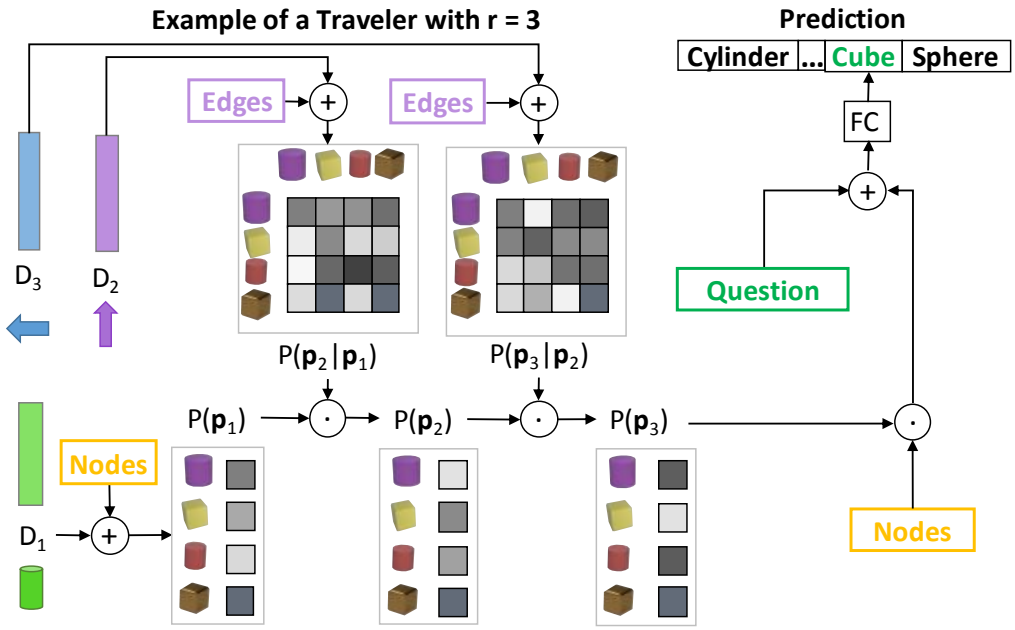


Figure 4.4: Proposed graph neural network architecture which learns traversal strategies for the scene graph (simplified for path length $r = 3$). While the visual *guide*, the graph *traveler*, and the *prediction module* are individual neural network components, they are optimized jointly in an end-to-end fashion. The visual guide takes as input the question and provides direction embeddings for the traveler to follow. The prediction module gives the final answer based only on the question and the destination nodes embeddings, the predecessors are therefore dismissed.

distribution over objects). For $i > 1$, we have to consider the transition probabilities $P(\boldsymbol{p}_i|\boldsymbol{p}_{i-1})$. Since the number of possible path options grows exponentially with the path length, we further reformulate this calculation for reasoning steps larger than one. We iteratively transform the path probability to the probability of each node laying for each step i as follows:

$$\begin{aligned}\mathbb{P}(\boldsymbol{p}_1,\ldots,\boldsymbol{p}_r) &= \mathbb{P}(\boldsymbol{p}_1)\cdot\mathbb{P}(\boldsymbol{p}_2|\boldsymbol{p}_1)\cdot\prod_{i=3}^{r}\mathbb{P}(\boldsymbol{p}_i|\boldsymbol{p}_{i-1}) \\ &= \mathbb{P}(\boldsymbol{p}_2)\cdot\prod_{i=3}^{r}\mathbb{P}(\boldsymbol{p}_i|\boldsymbol{p}_{i-1})\end{aligned}$$

Thus, the new estimation lies in the calculation of each probability $\mathbb{P}(\boldsymbol{p}_i)$. For this,

we make use of the function π^i which computes the probability of each node being in the path in an iterative way using the formulation:

$$\pi^i(j) \coloneqq \sum_k \mathbb{P}(j|k) \cdot \pi^{i-1}(k), \forall j, k \in \{1, \ldots, n\}, i \in \{2, \ldots, r\} \tag{4.7}$$

Thus, we only require the *starting probabilities* π^1 and *transition probabilities* $\mathbb{P}(j|k)$ in each reasoning step i to obtain the final destination nodes. We stop the calculation at reasoning step r and the final values become the probability of each vertex being the destination, *i.e.*, the node has information relevant for the question. Next, we define the models for obtaining the start and transition probabilities.

4.2.3 Neural graph architecture

In conventional graph neural networks for VQA, the node features V change depending on their neighbors in each training reasoning step, becoming a mixture of the initial and foreign object representations (*i.e.*, refining the vertex set) [Kembhavi et al., 2016; Santoro et al., 2017; Teney et al., 2016]. In comparison, our model keeps the semantic node representations and focuses on the network *topology*, learning to find relationships of the scene entities relevant for the current question (see Figure 4.4). We can easily shed light upon the choices of our model, as we retain the initial interpretation of its nodes and highlight the key links between them.

1. Visual guide. The *visual guide* considers the static graph as a map to be traversed using the question as the reference. That is, the guide takes as input the question, embeds it to a vector $\boldsymbol{h}$, *e.g.*, using an LSTM [Hochreiter and Schmidhuber, 1997] or a one dimensional CNN with self-attention [Gehring et al., 2016] and produces directional embeddings $\boldsymbol{d}^i$ for the traveler to follow on the graph. In case of an LSTM, we represent the question as the final hidden state, while we use weighted average over the feature maps in case of a CNN. We then use these question embeddings to generate vectors that *guides* our traveler module in the direction of the sought destination nodes (*i.e.*, nodes that comprise information for

inferring the current answer). The direction vector $\boldsymbol{d}^i$ at reasoning step i is obtained through learned fully-connected layers $\boldsymbol{d}^i := W^i_{\text{dir}} \cdot \boldsymbol{h} + \boldsymbol{b}^i_{\text{dir}}$, where $W^i_{\text{dir}} \in \mathbb{R}^{d^i \times h}$ and $\boldsymbol{b}^i_{\text{dir}} \in \mathbb{R}^{d^i}$ with the size of the direction embeddings d^i chosen empirically.

2. Graph traveler. The *graph traveler* traverses the visual graph based on the directions suggested by the *guide*. Thus, it produces prior probabilities (*i.e.*, the confidence of each node being the first one visited) and computes the transition probabilities (*i.e.*, confidence of traversing one node to the next).

For the first node in a path, we obtain the confidence by training a fully-connected layer on top of the *node representations* V from the visual graph and the *first directional embedding* $\boldsymbol{d}^1$ given by the guide:

$$\pi^1_\theta := \text{softmax}(W^1_{\text{path}} \cdot [\boldsymbol{d}^1 \parallel V] + b^1_{\text{path}}) \tag{4.8}$$

where θ is the set of all the learnable parameters in the model, *e.g.*, $W^1_{\text{path}}, \boldsymbol{b}^1_{\text{path}} \in \theta$ and $[\cdot \parallel \cdot]$ denotes the element-wise concatenation of the rows of a matrix and a vector. Thus, the weight and bias are shaped as $W^1_{\text{path}} \in \mathbb{R}^{1 \times (d^1 + v)}$ and $b^1_{\text{path}} \in \mathbb{R}$, respectively. Note that the linearity is applied node-wise (on the x-axis), *i.e.*, the output is a vector modeling a probability distribution over the nodes: $\pi^1_\theta \in [0, 1]^n$. The softmax function therefore normalizes over the nodes (the sum of the n confidences of each vertex being a starting node is equal to one):

$$\text{softmax}(\boldsymbol{x})_i = \exp(\boldsymbol{x}_i) / \sum_j \exp(\boldsymbol{x}_j), \forall j \in \{1, \ldots, n\} \tag{4.9}$$

where $\boldsymbol{x}$ is a vector, *e.g.*, in our setting, $\boldsymbol{x}$ comprises a logit for each vertex in $\mathcal{G}$.

In case of the transition probabilities, we leverage the edge features $E \in \mathbb{R}^{n \times n \times e}$ between each pair of nodes in the graph $\mathcal{G}$ and the directional embeddings:

$$\pi^i_\theta := \text{softmax}_{\text{row}}(W^i_{\text{path}} \cdot [\boldsymbol{d}^i, E] + b_{\pi_i}) \tag{4.10}$$

where $W^i_{\text{path}} \in \mathbb{R}^{1 \times (d^i + e)}$ and $b^i_{\text{path}} \in \mathbb{R}$. Note that there are in total $r - 1$ transitions (*e.g.*, for a path of length three, we compute two transition matrices), *i.e.*, $i \in$

$\{2, \ldots, r\}$. Here, the $\text{softmax}_{\text{row}}$ operates on matrices and normalizes over the outgoing edges (*i.e.*, over the rows), such as the sum over the outputs is equal to one. More formally, the function π^i has the following property:

$$\sum_j \pi_j^i = 1, \forall i, j \in \{1, \ldots, n\} \tag{4.11}$$

In the final reasoning step r, the *graph traveler* computes the probability of a node being the final destination π_θ^r (as introduced in Equation 4.7).

3. Prediction module. The *prediction module* differentiates between the problem types and generates the answer leveraging the probability distribution over the destinations (Figure 4.4). In case of query-type questions (*i.e.*, questions about the shape, color *etc.* of an object), the solution is determined from the *destination* nodes *i.e.*, soft path probabilities π^r at reasoning step r. Thereby, we estimate a *global graph encoding* $\boldsymbol{g}$ where we leverage the entire set of nodes which we weight by their probability being the destination as follows:

$$\boldsymbol{g} \coloneqq V \cdot \pi_\theta^r \tag{4.12}$$

where $\cdot$ marks the multiplication of a matrix with a vector. We then concatenate this visual global representation $\boldsymbol{g}$ with the question embedding $\boldsymbol{q} \in \mathbb{R}^q$. Then, a fully-connected layer is used to produce the final prediction over all possible answers. For *existence* questions, we answer the question with 'yes', in case that any of the destinations has a probability over 0.5, while in case of counting, we set the sum over the destination confidences as the final answer.

In *counting* and *exist* tasks, we leverage the number of destinations from our destination representation. For tasks, where the sum of the final soft path probabilities may be larger than one, as multiple destinations could be applicable (*e.g. counting* or *existence*), we use sigmoid function instead of softmax for these edge normalization. Then, we simply sum the output of the sigmoid function for each node j and use them for the final prediction (we provide more details in the next section).

4.3 Configuration details

In this section, we provide more details for implementing our model and on the learning specifics. To that end, we give insight into the parameter setup, optimization scheme, and initialization procedure of the learnable parameters in our network.

4.3.1 Question-based guide

On the AI2D dataset, for better comparability with related work, we employ the same input representation as [Kembhavi et al., 2016; Kim et al., 2017], *i.e.*, GloVe [Pennington et al., 2014] features pre-trained on 6B tokens from Wikipedia. As previous work [Kembhavi et al., 2016; Kim et al., 2017] uses Long-Short Term Memory (LSTM) modules [Hochreiter and Schmidhuber, 1997] for embedding the question sentence, we also use a single-layered LSTM with 256 hidden units for our question-based guide. We set the last hidden state of the word-level LSTM as the final global representation of the question.

For the CLEVR and COG datasets, we experienced LSTM convergence instabilities linked to significantly longer sentences (over 15 tokens). This LSTM-related issue was previously reported by others [Graves et al., 2014]. However, various works show strong potential of using 1D convolutions instead of recurrent neural networks. Such convolution-based models oftentimes exceed the accuracy of LSTMs while having a more stable learning procedure [Bai et al., 2018; Dauphin et al., 2016; Gehring et al., 2017]. We therefore opt to embed the questions using 1D convolutions with an attention module for these specific datasets.

The question-based guide encodes the words represented as one-hot vectors using a 1D convolutional neural network with self-attention. We use six convolution layers (with 32 output filters of size 3, stride 1) each with zero padding and ReLU activation. In case of COG and CLEVR, we do not share weights between the guides for each question type, however, the structure of each guide is identical. Finally, we obtain the attention module with a further 1D convolution and a sigmoid non-linearity. Even though softmax is the widespread way to normalize attention

modules, sigmoid has the benefit of weighting individual elements (*i.e.*, words) independently to the total amount (*i.e.*, each of the word confidences does not influence the values of the other words in the sentence).

Our goal is that, *e.g.*, the word `sphere` is weighted in the same way in both 'How many *spheres* are there?' and 'How many *green spheres* are there?'. This would not be the case for softmax, as softmax would give both `sphere` and `green` a high weight and, thus, sphere would automatically have a smaller weight as it has to share the amount with `green` after normalization. In the final layer, the number of hidden units corresponds to the maximal path length r (*i.e.*, we get a different attention map over the words for each step in the path). The final representation of the traversal directions (*i.e.*, for each $i \in \{1, \ldots, r\}$) is obtained via a weighted sum of the words in the question based on the distribution of the previously defined attention maps. Thus, if several attributes are relevant for the specific reasoning step, the values in the the directional embeddings increase.

4.3.2 Graph traveler

The graph traveler uses the directional embeddings produced by the guide to traverse the graph in search of the destination nodes. To obtain the confidence $\pi^i(j)$ for each reasoning step i and vertex j being in the path, we concatenate each node for $r = 1$ and each edge when we have $r > 1$ with the directional embeddings followed by two fully-connected layers. The first layer has a size of 256 and ReLU activation, while the second layer is a single hidden unit representing the confidence of the starting nodes and transition probabilities, respectively.

As we constrain $\pi^r(j)$ to be in the interval $[0, 1]$, we use either sigmoid (for counting and existence tasks) or softmax (for other tasks). The nonlinearities are both applied on either the starting nodes *i.e.*, $\pi^1(j)$ or on each edge (*i.e.*, for each node pair). In case of the sigmoid normalization, we replace the sum operation of Equation 5 with the maximum function applied on the input edge confidences, in order to hold the premise that all confidence values are at most one, *i.e.*, $\pi^r(j) \leq 1$.

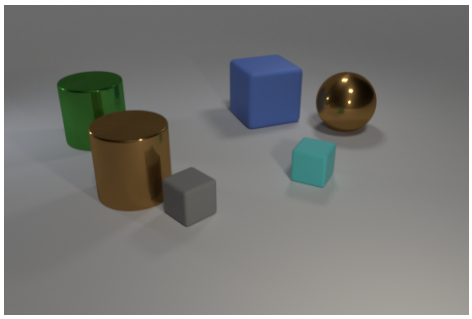
4.3.3 Prediction module

Datasets for visual reasoning cover a wide variety of task formats (*e.g.*, counting, diagram question answering) and, therefore, differ greatly in their solution modes (*e.g.*, multiple choice vs. free result format), which we need to take into account in the prediction module output (examples of the datasets are shown in Figure 4.5).

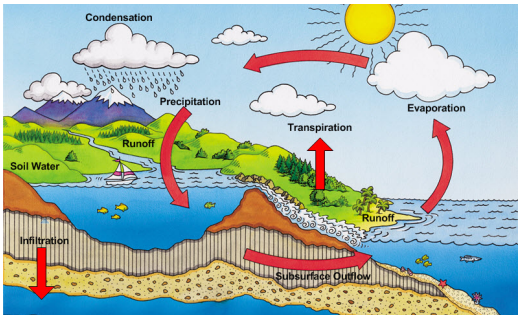
Since AI2D is a dataset in a multiple choice setting, we need to input the multiple answers to the model. As an input to our prediction module, we concatenate each of the possible multiple choice answers with the final representation of the question and of the destination node individually. This is followed by two fully-connected layers with the last layer containing a single output unit corresponding to the confidence of the current answer being the correct one.

In contrast, COG and CLEVR are both open-ended datasets, and, thus, we obtain the final prediction by leveraging the set of all *seen* answers in the training set. More precisely, the prediction module is carved as a classification task where the final layer is fully-connected with the number of output units equal to the number all of possible answers across all tasks. For COG, we use two different streams in the same way as related work [Yang et al., 2018a], *i.e.*, we differentiate between: pointing to an object in the graph and other type of answers in text format (*e.g.*, 'yes', 'circle', 'red'). In case of pointing questions, the prediction module consists of a fully-connected layer with a single output unit for each node in the graph, which are normalized using softmax (*i.e.*, sum over all nodes will equal to one), while the other answer-types comprise either yes/no or attribute-based predictions. As we previously specified, for yes and no answers, we use sigmoid activation for the soft paths, while for the attribute-based questions we leverage softmax.

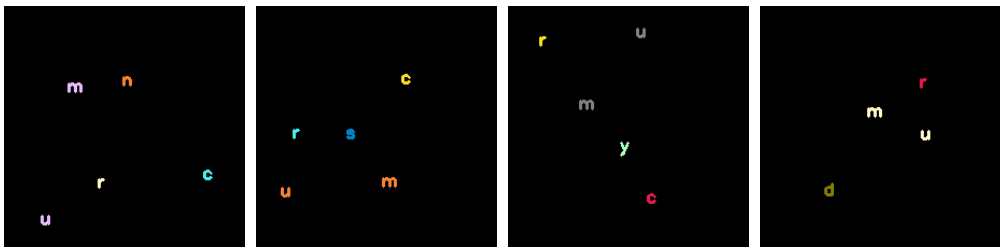
In case of CLEVR, we have five different types of questions, where we use sigmoid in all counting-related tasks (*exist, count and compare numbers*), and softmax otherwise (*query attributes and compare attributes*). The prediction modules of softmax related tasks consist of a simple fully-connected layer with the number of output units equal to the number of possible answers in the training set. This final layer is also using a softmax normalization over the answers present in the training



CLEVR. How many green cylinders are left of the large sphere?



AI2D. What is the name of the process of a liquid converting to the gaseous state?



Frame 1 Frame 2 Frame 3 Frame 4

COG. Is the color of the current r and latest c and the color of m and u equal?

Figure 4.5: Examples extracted from the CLEVR, AI2D, and COG benchmarks.

set. In case of the sigmoid-based tasks, we use a greedy procedure in combination with the cross entropy loss. For example, in the case of *exist*-type questions, we use the neuron with the highest activation of the nodes in the final step r (*i.e.*, destination confidences) as the label in our loss function, while for the target we use zero if the correct answer is 'no', otherwise we input a one.

During the test phase we predict a 'yes' if any of the nodes have a final confidence of over 0.5, otherwise the model outputs a 'no'. In case of *counting*, we use the cross entropy loss over the confidences in time step r as the prediction. For a ground truth answer equal to k, we assign as the target a vector that has the same size as the number of nodes in the graph. In this vector we set a one if the particular node value is in the top-k highest confidence in the prediction (*i.e.*, confidence in step r), otherwise the particular value in the target vector is set to zero. In the test phase, we count the number of activations that have a higher value than 0.5.

Dataset	Type	# Images	# Instances	# Questions
Textbook Graphics				
AI2D [Kembhavi et al., 2016]	Diagrams	5k	9.1	15k
Synthetic Datasets				
COG [Yang et al., 2018a]	Videos	11M	9.6	44M
CLEVR [Johnson et al., 2017a]	3D-Forms	100k	6.5	700k

Table 4.1: Benchmarks for visual reasoning that we for evaluating our network based on soft paths. We report the task type, number of images/videos, average amount of instances per sample, and number of questions.

4.3.4 Optimization

Learning setup. Except for the weights of the pre-trained GloVe model in AI2D, all weights are initialized randomly following the Xavier initialization [Glorot and Bengio, 2010]. Biases are initialized with zeros. Network weights are optimized using Adam [Kingma and Ba, 2015] with an initial learning rate of 0.00025. For other parameters, we use the default values in TensorFlow: 0.9 for the exponential decay rate for the first moment estimates β_1 and 0.999 for the second one β_2. The small initial learning rate is common for graph neural networks (*e.g.*, [Santoro et al., 2017]) as larger ones often cause convergence instabilities.

The models are trained at most 30 epochs using early stopping with the validation set performance as an indicator. All models are initialized from scratch in TensorFlow. We choose a maximal path length r empirically on the validation data. The question-based guide uses multiple 1D convolution layers with 32 hidden units, while the final fully-connected layers of the graph traveler have the size of 128.

Computational complexity. For each step, we have a computational complexity of $O(E)$ where E is the number of edges. Note that this is also the case for graph convolutions [Kipf and Welling, 2017] which we discussed in Chapter 2. The training time for 30 epochs on a single GeForce GTX 1080 Ti lasts around one hour for the AI2D and 20 hours for CLEVR, and 100 hours in total for the COG video dataset.

4.4 Answering questions by following paths

4.4.1 Dataset overview

We perform several studies on three challenging datasets for visual reasoning with diverse query types (overview in Table 4.1). All datasets cover visual samples, task queries with the ground-truth solutions (in open-ended or multiple choice form), and graph annotations for the images. In Section 4.4.2, we evaluate our model on video sequences, then, in the task of diagram question answering (Section 4.4.3) and on highly compositional reasoning problems on 3D synthetic images (Section 4.4.4). We further discuss how different path lengths r impact the performance (Section 4.4.5), evaluate how well our model generalizes to previously unseen tasks (Section 4.4.6) and, finally, visualize concrete examples of soft paths (Section 4.4.7).

4.4.2 Visual reasoning on videos

Setup. In this section, we use the COG [Yang et al., 2018a] dataset as a test bed for both spatial and temporal reasoning. The dataset comprises over 11 million questions on videos. While the videos are of synthetic 2D scenes, it specifically targets temporal memory and logical deductive reasoning about video input, being difficult for humans [Yang et al., 2018a]. The task is to deduce the correct answer while taking into account changes of the scene in four different query types: pointing, yes/no, conditional, and attribute-related questions. Higher number of scene entities is also characteristic for the dataset.

Results. We demonstrate the effectiveness of our model in Table 4.2. Additionally to the original Working Memory [Yang et al., 2018a] approach, we compare our model to three baselines: (1) random performance, (2) a question-only model consisting of a 1D CNN over the question words followed by fully-connected layers, and (3) a graph-based approach, where instead of computing the answer from the destination nodes of the found paths, we use a joint embedding of the question and all of the nodes in the graph as input and use fully-connected layers to make a prediction.

Video Reasoning Method	Atts.	Condit.	Point	Yes/No	All
Baselines					
Random	1.9	8.4	17.5	50.0	26.6
Question-only [Yang et al., 2018a]	1.6	2.3	19.4	49.7	27.4
Memory Networks					
Working Mem.† [Yang et al., 2018a]	–	–	–	–	93.7
Graph-based Methods					
Question+Nodes (Ours)	73.7	63.5	92.5	57.9	63.3
Soft Paths (Ours)	**99.2**	**98.4**	**100.0**	**95.0**	**97.2**

Table 4.2: Results for visual reasoning on videos on the test set of COG for different tasks: pointing, existence, conditional questions, and questions about object attributes. † Best model selected from 50 trained networks (opposed to the other models that were evaluated after a single optimization run).

Our model yields the best recognition rates in all query types. The distinction from the natural language-based benchmarks becomes obvious, as the *question-only* approach exceeds the random baseline by less than 1%. *Visual* reasoning is therefore decisive for this benchmark. The yes/no questions have been the major source of the unreliability of our model. Our analysis of these confusions indicates occasional difficulties in case of 'and' connections in the question (*e.g.*, 'Shape of last magenta object equal shape of last lavender object and shape of now mint object equal shape of last olive object?'). Nonetheless, our model achieves excellent performance of 100% for pointing questions and establishes a new state-of-the-art overall accuracy of 97.2%.

4.4.3 Diagram question answering

Setup. Next, we evaluate our approach on real-life images in the diagram understanding task. The AI2D [Kembhavi et al., 2016] dataset encompasses images extracted from school textbooks of various subjects and evaluates understanding of causal relations in these figures. As middle school pupils are required to learn from such diagrams, reason, and answer questions about them, this dataset represents an

Diagram QA Method	Overall Accuracy
Baselines	
Random	25.00
Classical VQA Methods	
VQA [Agrawal et al., 2017]	32.90
Graph Neural Networks	
DQA-Net-DSDP [Kembhavi et al., 2016]	38.47
DQA-Net-DGGN [Kim et al., 2017]	39.73
DQA-Net [Kim et al., 2017]	41.55
Soft Paths (Ours)	**43.45**

Table 4.3: Results on the AI2D dataset for diagram question answering on real images extracted from school textbooks [Kembhavi et al., 2016]

excellent realistic testbed for visual reasoning. As we are dealing with real-life data, AI2D is smaller and noisier than other datasets we used for testing with 666 lessons of total 5k diagrams and 15k questions.

Results. In Table 4.3, we compare our model with a multitude of published approaches including three graph-based methods. As AI2D is evaluated in multiple choice form with four possible options, random choice performance is 25%. Overall, there is a clear benefit of using structured approaches. Our graph traversal-based model consistently outperforms state-of-the-art graph neural networks and, therefore, confirms the effectiveness of focusing on traversal schemes and the found destination nodes, instead of the message passing paradigm.

4.4.4 VQA on 3D synthetic images

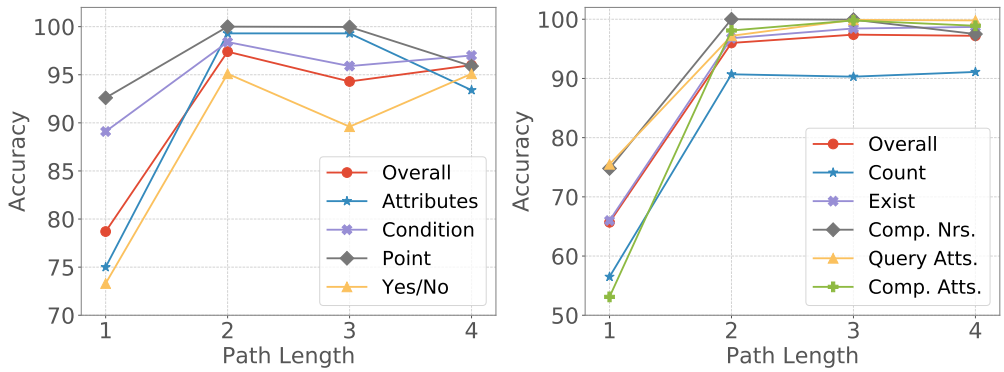
Setup. The Compositional Language and Elementary Visual Reasoning dataset (CLEVR) [Johnson et al., 2017a] is a widely used diagnostic benchmark for compositional understanding of 3D scenes for different tasks, such as counting, finding attributes of objects based on their relations with other instances, and comparison between object attributes. Long reasoning chains demanding memory-related tasks

Approach	Count	Exist	Comp. Nrs.	Query Attrs.	Comp. Attrs.	All
Human [Johnson et al., 2017b]	86.7	96.6	86.5	95.0	96.0	92.6
Qtype [Johnson et al., 2017b]	34.6	50.2	51.0	36.0	51.3	41.8
Classical VQA Methods						
LSTM [Johnson et al., 2017b]	41.7	61.1	69.8	36.8	51.8	46.8
CNN [Johnson et al., 2017b]	43.7	65.2	67.1	49.3	53.0	52.3
CNN+SA [Santoro et al., 2017]	64.4	82.7	77.4	82.6	75.4	76.6
QGHC [Gao et al., 2018]	91.2	78.1	79.2	89.7	86.8	86.3
FiLM [Perez et al., 2017]	94.3	99.1	96.8	99.1	99.1	97.7
Compositional Models						
N2NMN* [Hu et al., 2017]	68.5	85.7	84.9	90.0	88.7	83.7
PG(9K)* [Johnson et al., 2017b]	79.7	89.7	79.1	92.6	96.0	88.6
PG(700K)* [Johnson et al., 2017b]	92.7	97.1	98.7	98.1	98.9	96.9
Memory Networks						
Work. Mem. [Yang et al., 2018a]	91.7	99.0	95.5	98.5	98.8	96.8
MAC† [Hudson and Manning, 2018]	**97.1**	**99.3**	96.8	99.1	99.1	**98.9**
Graph Neural Networks						
CNN+RN‡ [Santoro et al., 2017]	90.1	97.8	93.6	97.9	97.1	95.5
Soft Paths (Ours)	91.3	98.6	**99.6**	**99.5**	**99.8**	97.5

Table 4.4: Visual reasoning results for different tasks on the CLEVR test set [Johnson et al., 2017a]. * denotes the use of extra supervision in form of program labels, ‡ denotes the use of data augmentation, † denotes the use of pre-trained models.

and absence of question-based biases are distinctive for this benchmark. Although it is comprised of synthetic scenes, conventional VQA models often face significant difficulties on CLEVR as they tend to focus on the dataset bias [Gao et al., 2018; Johnson et al., 2017b; Santoro et al., 2017].

Results. We report results on all five problem types of the CLEVR benchmark: (1) counting (*e.g.*, *How many cubes are there*), (2) existence (*e.g.*, *Are there any yellow cubes in front of the sphere*), query attributes (*e.g.*,), and questions about comparing numbers and attributes of objects. A high number of novel methods have been recently proposed to tackle CLEVR reasoning tasks, which we group based on their way of addressing object relations and compare to our model in Table 4.4. We achieve state-of-the-art accuracy of over 99% on three tasks (comparing numbers



(a) COG (b) CLEVR

Figure 4.6: Performance for different maximal path lengths r on the validation sets of COG (left) and CLEVR (right).

and two attribute-related problems) and repotr a strong overall performance (97.5%) surpassing humans (92.6%), and the recent graph-based method [Santoro et al., 2017] based on edge refinement (95.5%).

4.4.5 Impact of path length on performance

As we explicitly focus on *relations* in the scene, we compare variants of our model to measure the effect of different restrictions of the *soft path* at length r. Figure 4.6 illustrates changes of accuracy in relation to r for different COG and CLEVR tasks. The model benefits immensely from considering paths of length two or more, *e.g.*, for the *query attributes* task, the percentage of correct answers rises from 53.1% ($r = 1$) to 98.1% ($r = 2$), further improving to 99.8% ($r = 3$), confirming the significance of causal connections in the scene.

Starting at $r = 4$ for CLEVR and $r = 3$ for COG, we observe a slight decline in overall performance, which we link to the extend of chained questions in the datasets. For example, in a question 'What is the material of the sphere behind the tiny brown thing to the right of the green object?' (Figure 4.1) the reasoning chain consists of two pairwise relationships. In general, enforcing longer paths than necessary for the question is not a problem in our architecture, as it permits self-loops. However, the option of including more nodes than required might result

in higher level of noise, as the overall search space becomes larger. This slight accuracy drop should be viewed with caution, as it is also connected to the nature of the questions in the dataset, *i.e.*, it is expected to increase with the amount of entities mentioned in the question. Nonetheless, when further increasing the path length the performance stabilizes, *e.g.*, for COG, our model achieves 95.6% when the maximum length is set to eight (*i.e.*, $r = 8$).

4.4.6 Performance on unseen tasks

Humans have an impressive ability to address new tasks of increasing difficulty by transferring solutions from familiar problems. Similarly, our motivation for focusing on the scene *structure* is to develop a model which processes queries by decomposing them into granular tasks, which then could be easily re-used to answer questions our model has never seen before.

To evaluate our assumption, we propose a new challenging benchmark for visual reasoning on problems not previously seen during training. We regard three tasks from the CLEVR dataset: *query attributes*, *existence*, and object *counting*. In our proposed evaluation setup, the model is trained on one of these tasks and is intended to solve another one. Consider the *existence* task, where we output `yes` if in the last reasoning step r there is at least one destination node with probability over 0.5 (see Section 4.2.3). As the node representations are not refined throughout the process, we can extend our model to *counting* without additional training, by merely using the *counting* prediction module version, *i.e.*, summing the number of destinations with an activation over 0.5, as we describe in Section 4.2.3. For the *query attribute* task, we select the node with the maximal activation.

We report the performance of our model on previously unseen tasks in Figure 4.7. Our approach successfully applies the knowledge it had acquired from *counting* or *existence* to previously unseen query types. These two tasks are especially re-usable as they involve a universal granular question: whether objects are present in the scene, or not. In case of learning on the attribute-based questions, we assume that the destinations are always available (as we question specific attributes of the node

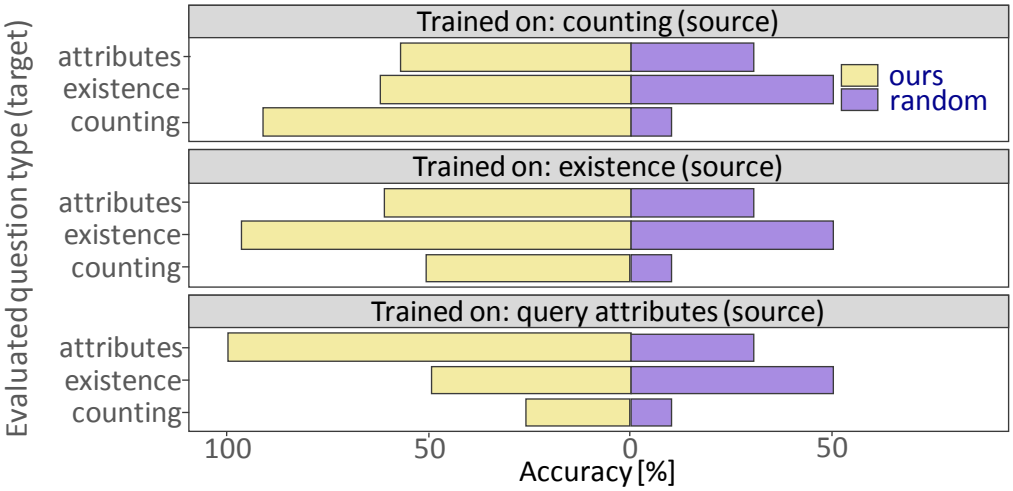


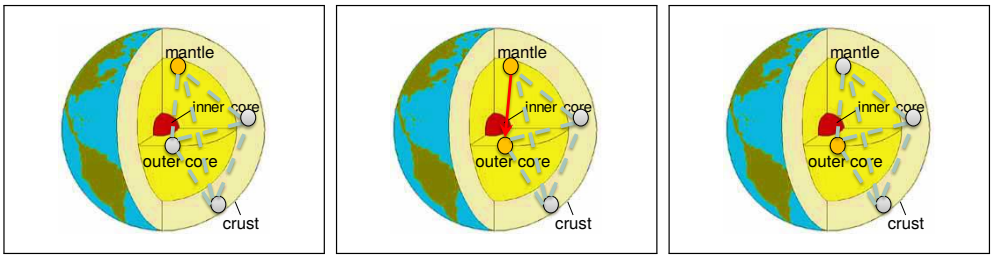
Figure 4.7: Generalization to unseen tasks: our model is trained on one query type and evaluated on a different one (*e.g.*, trained on counting and tested on existing).

and not their presence). Re-usability of the learned information is therefore lower. Training on the *counting* queries turned out to be most beneficial for solving new problems. We assume, this is due to *counting* being a more composite task as it covers both checking for object presence and determining whether the objects have certain properties (*e.g.*, 'What number of brown balls are the same size as the metal object?'). Our model trained on the *counting* task was able to solve the query attribute problem in 56.4% of the time, surpassing random chance (30.6%) by 25.8%.

Obviously, solving previously unseen tasks is per design a much harder problem than conventional supervised visual reasoning and the recognition rates are considerably lower. Apart from the lack of supervision, language expressions not present during training pose an additional challenge (*e.g.*, 'how many' if the model was trained on the *existence* task and evaluated on *counting*). Still, our model consistently outperforms the random chance baseline being able to address new tasks without costly annotations of training samples.

4.4.7 Qualitative results

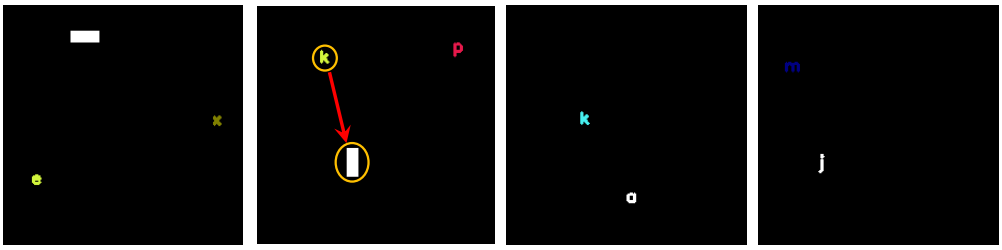
An important property of our model is the ability to trace back the underlying *reasoning* behind the final answer by analyzing the traversed paths for inference.



(a) Starting node ($i = 1$) (b) Transition (c) Node at $i = 2$

Question: What is between mantle and inner core? || **Answer:** Outer Core

Figure 4.8: A correctly answered question with the generated path on a diagram.



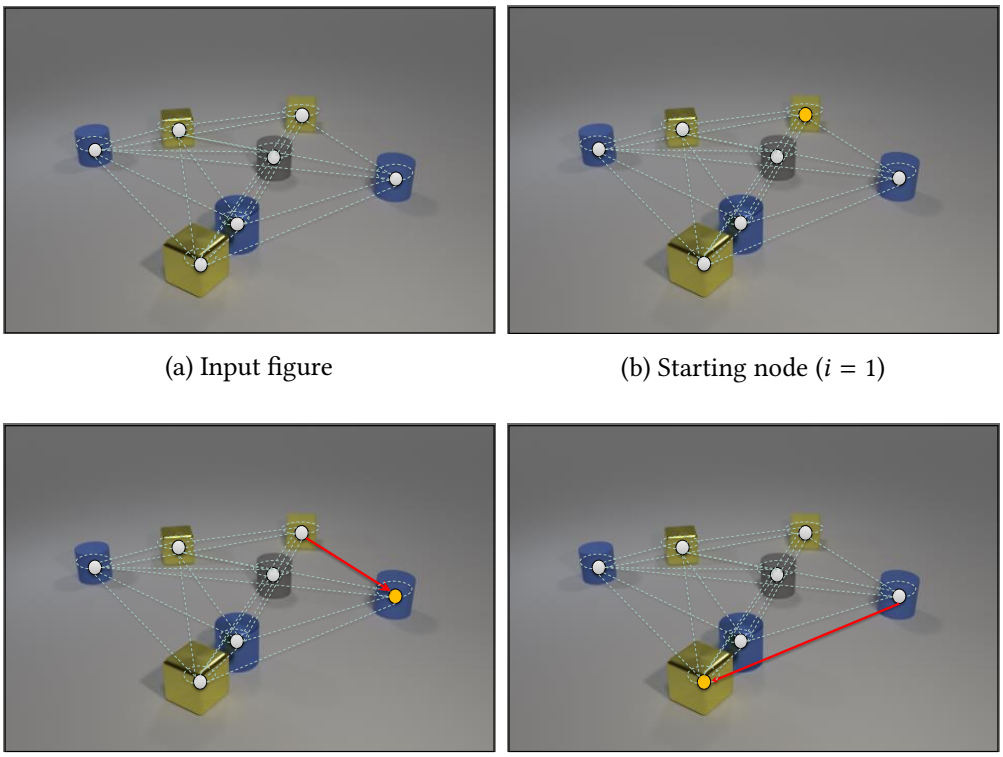
Frame 1 Frame 2 Frame 3 Frame 4

Question: Does a white object exist in the current frame (*i.e.*, frame 2) to the right side of the latest lime object? || **Answer:** True

Figure 4.9: An example answer generated by our model on the COG dataset.

In Figure 4.8, we revisit the final soft paths of our model on an example from the AI2D textbook diagram benchmark. Our model solves the query 'What is between mantle and inner core' with a soft path of length 2 by starting at the `mantle` and, next, choosing the destination `outer core`. As the correct destination was selected by our model, it easily infers the correct answer: `outer core`.

Figure 4.9 visualizes four frames from the COG dataset and an accompanying question with an answer inferred by our model. Since the question is related only to objects in the second frame, the generated soft path is fully included in a single image: in the first reasoning step the model selected the lime object (*i.e.*, the `letter k`) and, then, selected as the destination node the white object. Since the sought object was found by our approach, the model inferred the correct answer 'yes'.



(a) Input figure (b) Starting node ($i = 1$)

(c) Second step in the path ($i = 2$) (d) Third step in the path ($i = 3$)

Question: There is a tiny rubber thing that is right of the matte cube; are there any yellow cubes in front of it? || **Answer:** 'Yes'

Figure 4.10: A correctly predicted answer of a compositional question comprising three reasoning steps and its traversed path to the destination node.

In case of CLEVR, most queries are very long and strongly compositional for which we require to generate paths of length 3. In Figure 4.10, we visualize an example figure, question, and predicted answer of our soft path architecture. We show the nodes with the maximal probability at each reasoning step i. Edges which belong to the path are marked with red arrows, starting with the source node (*i.e.*, the `small yellow cube`) and ending in the final destinations, which are the only graph components used as input in the prediction modules (*i.e.*, the `large yellow cube`). We see that the network found in the first step the `matte cube`, then followed it to the right where it found a `tiny rubber thing`. This new object was then used

to get to the destination by searching objects `in front` of it that are both `yellow` and have a `cube` shape. Finally, the network selects if such an object exists, and since it found a matching object, it directly infers the correct prediction 'yes'. We show more examples of the answers with the associated paths in Appendix A.

4.5 Summary and discussion

In this chapter, we presented a novel approach for compositional visual reasoning, where we employ a graph neural network architecture to tackle far-reaching relationships in the scene. Our framework learns how to traverse the graph in a controlled way and, then, answers the question based on the reached destination nodes of the found paths. Our model exceeds state-of-the-art methods on two challenging datasets for visual reasoning: on videos (COG) and diagram question answering (AI2D) as well in the three tasks on 3D synthetic data (CLEVR). At the same time, our model is highly interpretable as the graph trails directly shed light on the underlying reasoning, showing that our model breaks complex instructions into smaller tasks. Furthermore, we demonstrate the positive impact of focusing on relevant semantic *structures* on the ability to reuse the acquired knowledge for novel tasks. In this new benchmark setting, our model was trained on a certain question type (*e.g.*, *existence*) and could successfully handle tasks of a different kind (*e.g.*, *counting*) without any further training. Our experiments show encouraging evidence that modern visual recognition approaches could benefit further from structured methods especially in high-level understanding of global causal relations. In the next chapter, we address the task of multi-modal question answering that aims at reasoning on *both* visual and textual data *simultaneously*.

5 Multi-Modal Reasoning

In the last chapter, we considered the task of visual question answering, where we leveraged methods for high-level understanding on the detected graphical content in the page. However, a multitude of questions cannot be answered solely based on such figures, but external knowledge of the surrounding text is often necessary. Multi-modal machine comprehension is a task in machine learning that deals with extracting knowledge from multiple modalities like diagrams and text. Particularly, Textbook Question Answering (TQA) focuses on questions based on the school curricula, where the text and diagrams are extracted from textbooks.

In this chapter, we propose a novel deep model that can handle different knowledge modalities for answering difficult questions from textbook content. We compare three different information representations that are encountered in TQA: a visual representation learned from images, a graph representation of diagrams, and a language-based representation learned from the accompanying text. We evaluate our model on the TQA dataset that contains text and diagrams from lessons of the sixth grade curricula. Even though our model obtains competing results compared to state-of-the-art, we witness a significant gap in performance compared to

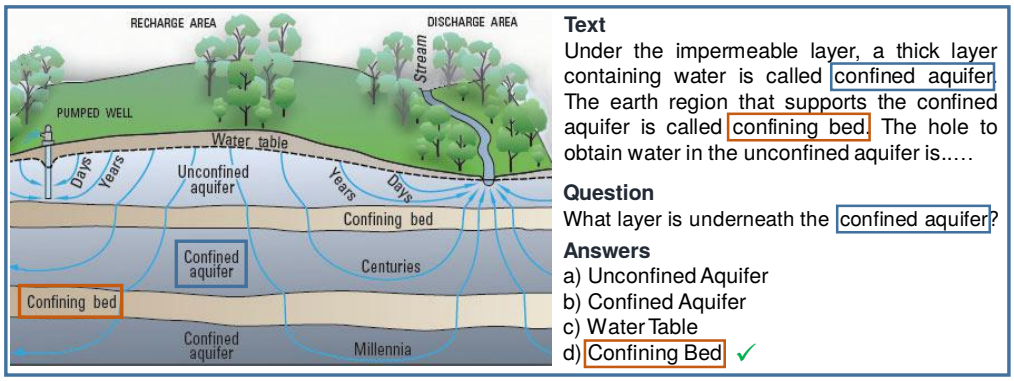


Figure 5.1: Overview of textbook question answering task. Our model needs to answer a question based on knowledge gained from text, a diagram, or both.

humans. In this work, we perform a thorough analysis of our model and discuss possible improvement strategies by visualizing the distribution of the multiple types of errors produced by our proposed neural network.

Contributions. To that end, we introduce a novel network for recognizing the text content which we need to extract for the multi-modal question answering approach. Then, we propose an architecture specialized on answering questions on both visual content in the form of diagrams and the surrounding text extracted from sixth grade textbooks. Finally, we provide an in-detail evaluation of the performance of our model for different dataset settings.

Publications. In this chapter, we present a text recognition approach published in [Bender*, Haurilet* et al., 2019] and a novel architecture for reasoning on textbook content introduced in [Haurilet et al., 2018].

5.1 Task overview

While at a very early age humans can answer basic questions about their environment, we start to analyze and understand graphics, (*e.g.*, diagrams) at a much later time. In school years, children learn to analyze and understand complex illustrations, and are capable to extract important information and answer difficult questions

about them. The type and style of these illustrations have many different forms in terms of colors, structure types, and structure difficulty. While some illustrations in textbooks are easy, like simple drawings, in later school years we see more difficult types of figures like diagrams, plots, and tables. Diagrams are especially complex, since they can comprise different types of nodes encompassing other figure types, *e.g.*, drawings, text, and even natural images. Furthermore, we have various relationship types between the entities, *e.g.*, textual descriptions linked to the nodes or directly to the edges. We also have directed relations, usually represented with edges marked with an arrow sign, while some relations are not explicitly marked (example in Figure 5.1 of a query related to both the text content and the diagram).

While answering questions on natural images has received widespread attention in the computer vision community for several years, inferring answers on textbook content has been scarce. In this chapter, we address this task where we first shortly discuss how to extract relevant information from such pages. We then introduce an edge-centric neural architecture that leverages the relations between multi-modal nodes encompassing both sentences and information extracted from illustrations.

To that end, we compare different knowledge representations for our model: (1) the text-based model, where we use the surrounding text for answering the questions, (2) the image-based model represents the image by extracting the features of a pre-trained CNN, and (3) the graph-based representation embeds the diagram as a graph where the nodes consist of the detected text and its location. We investigate the predictions of our model to find out the reasons for the large gap to human performance. To that end, we select a subset of incorrectly answered questions in the test set of TQA and analyze the reasons why the approach failed.

5.2 Analyzing textbook content

For enabling the use of methods for multi-modal question answering, we first need to process the underlying textbook content. We discuss *how* to recognize the different components in documents in Chapter 3 where we leverage deep networks

for fine-grained localization. While this pre-processing is sufficient for graphical content (as one only needs their location), textual content requires a further pre-processing step entailing the recognition of characters and mapping these to textual format (*e.g.*, natural language, LaTeX). As mentioned in Chapter 2, the mapping of visual representation of text to character sequence is called *text recognition* and was addressed in the past [Liao et al., 2019; Wang et al., 2012]. While in Chapter 2, we discuss different approaches for text recognition, in Appendix B, we introduce a novel architecture for this task. Our network comprises a CNN for encoding the visual input and an RNN for generating the underlying text sequence. Our approach achieves state-of-the-art results on the IM2LATEX-100k dataset [Deng et al., 2017].

5.3 Method overview

We define the multi-modal comprehension task in the context of question answering in a multiple choice setting. That is, the model is given knowledge $\mathcal{K}$ from a textbook lesson (*e.g.*, a set of sentences, a set of nodes, or visual features) and a representation of the question Q. Based on such information, the model needs to select the correct answer from a set of m answers $\mathcal{A} := \{A\}$ with $m := |\mathcal{A}|$.

However, the knowledge $\mathcal{K}$ can comprise a large amount of data (*e.g.*, several pages of text) while the information relevant for answering the current question lies only in a small portion of it (*e.g.*, in a single sentence). Since this is mostly the case for the text- and graph-based data, to avoid overfitting of our model, we employ a pre-processing step where we filter out unrelated sentences and nodes.

Our approach relies on the basic intuition that for each question Q, there is a set of s supporting entities $\mathcal{S} := \{S\}$ with $S \in \mathcal{K}$ and $s := |\mathcal{S}|$ that would help in verifying the correctness of each (Q, A) pairs. Thus, our approach consists of two main steps: (1) Selecting n supporting sentences/nodes from $\mathcal{K}$ for a given question Q and; (2) Based on $(Q, \mathcal{S})$, verify the correctness of each answer $\forall A \in \mathcal{A}$. Next, we show how we select the supporting knowledge entities from $\mathcal{K}$ and, then, we describe our neural network that we use to infer the answer.

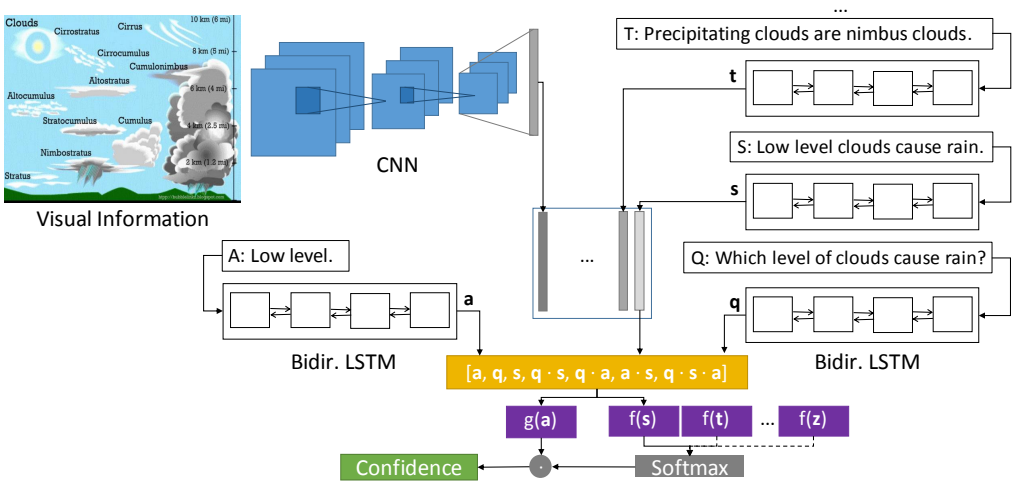


Figure 5.2: Architecture of the image+text deep neural network. The model comprises three different parts: (1) visual and textual embedding module, (2) a multimodal fusion network, and (3) the prediction module. Note that T, S, Q, and A are matrices that represent the knowledge entities, question, and answer, respectively, where each row encompasses a vector depicting a word encoding. These variable-sized matrices are decoded to a vector format using pre-trained bidirectional recurrent networks applied on each word embedding. Then, we use the final hidden state of the recurrent networks as the final fixed vector representations denoted with $\boldsymbol{s}$, $\boldsymbol{t}$, $\boldsymbol{a}$, and $\boldsymbol{q}$. These vectors are combined via pairwise and triplewise similarity, which are used to compute the *confidence* of the supporting entities in being relevant for the input query (marked with the function f) as well as the confidence in the current answer (denoted with g).

5.4 Selecting supporting nodes and sentences

To select the set of supporting knowledge for a certain question Q, we measure the similarity of a vector representation $\boldsymbol{k}$ of each knowledge entity $K \in \mathcal{K}$ (in the provided text and diagrams) and the vector embedding of the current question $\boldsymbol{q}$. That is, we first embed the question and knowledge entities to a common d-dimensional representation space using text encoders such as recurrent neural network or pre-trained transformers. Next, we calculate the similarity between the encoded input query $\boldsymbol{q}$ and each knowledge entity $\boldsymbol{k}$ using the similarity measure $\sigma : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$ (*e.g.*, the cosine similarity). Then, the top-s most similar

knowledge entities are included to the set of supporting sentences/nodes $\mathcal{S}$. Given the supporting entities and the question, we use the deep neural network presented in Figure 5.2 to verify each answer A from the set of multiple-choice answers $\mathcal{A}$.

5.5 Answering questions through edge refinement

First, our approach encodes the inputs to: (1) a question embedding $\boldsymbol{q}$, (2) a representation of each supporting entity $\boldsymbol{s}$ (vector encoding of elements from the set of supporting sentences and nodes, *i.e.*, $S \in \mathcal{S}$, and (3) each multiple choice answer $\boldsymbol{a}$ using separate fully-connected layers (*i.e.*, without weight sharing). Then, these embeddings are concatenated with their pairwise and triplewise similarity using element-wise multiplication, *i.e.*, for each answer, question, and knowledge entity:

$$\phi(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{s}) := [\boldsymbol{s} \parallel \boldsymbol{q} \parallel \boldsymbol{a} \parallel \boldsymbol{s} * \boldsymbol{q}, \boldsymbol{s} * \boldsymbol{a} \parallel \boldsymbol{q} * \boldsymbol{a} \parallel \boldsymbol{q} * \boldsymbol{a} * \boldsymbol{s}] \tag{5.1}$$

with $\parallel$ marking the concatenation and $*$ denoting the elementwise multiplication.

Then, we split the output of this layer into two streams: (1) the first stream captures the confidence of the answer A to be the correct one, (2) while the second stream weights the confidence of the model in the knowledge subset $\mathcal{S}$ for being suitable to verify (Q, A). We calculate this confidence through an attention module using softmax normalization. Therefore, we encode each $S \in \mathcal{S}$ using a fully-connected layer followed by the softmax normalization modeling a probability distribution over the s supporting entities. Finally, the two streams are fused using element-wise multiplication. During testing, the answer with the highest confidence is selected as the prediction of our architecture.

Next, we introduce several networks for textbook question answering, which operate on different input modalities: text-, graph-, and image-only. To have a better understanding of these networks leveraging the different modalities, we propose two additional networks: a graph-baseline that only leverages nodes for inference and a model that leverages the fusion of the image with the text embeddings. Next, we describe our network and our implemented baselines .

Text-based network. Our text-only model uses the surrounding text to generate the answers of the question. To that end, the input to our approach comprises the entire textual content available in the lesson associated to the current input query. Thus, the knowledge data encompasses the set of sentence pairs in the lesson, *i.e.*, each supporting entity S is equal to a pair of sentences. We set a fixed number of sentences s for each setting and choose the sentences based on the confidence to be supporting the input query (as described in Section 5.4).

Graph-based network. In a similar manner as for the text-based network, since the number of nodes in the diagrams is high, we select a set of supporting nodes based on the question. Thus, we choose s nodes that have the highest similarity to the question, where the similarity is shaped by $\sigma(\boldsymbol{q}, \boldsymbol{k})$ where $\boldsymbol{k}$ is a vector encoding of each edge in the graph. In comparison to the text-based model, in the graph setting, we leverage an edge representation of the supporting nodes (*i.e.*, for each edge in the diagram that contains a supporting node). To that end, we encode each edge by concatenating the vector representation of the source and target node.

Image-based network. The image-based network receives additionally to the question and answer pair, a global visual representation of the diagram I extracted from a CNN pre-trained on ImageNet [Russakovsky et al., 2015].

Graph baseline. This baseline leverages the top-1 supporting node (*i.e.*, the node that is closest in cosine similarity with the input query) and infers the answer from the set $\mathcal{A}$ which has the highest similarity to the top-1 node.

Text+Image network. In case of the Text+Image setting, we include the visual information as another vector in the supporting sentences set (Figure 5.2).

5.6 Evaluation setting

Dataset overview. TQA [Kembhavi et al., 2017] is a dataset for multi-modal machine comprehension encompassing over 600 lessons and exercises from the sixth

Method	True/False	MC	Text	Diags
Baselines				
Random	50.1	22.9	33.6	25.0
Memory Networks				
MemN + VQA [Kembhavi et al., 2017]	50.5	31.1	38.7	31.8
MemN + VQA + HT [Li et al., 2018a]	50.3	28.1	36.9	29.8
MemN + DPG [Kembhavi et al., 2017]	50.5	30.1	38.7	32.8
Other Approaches				
BiDAF + DPG [Kembhavi et al., 2017]	50.4	30.5	38.7	32.7
Challenge	–	–	45.6‡	35.9
IGMN [Li et al., 2018a]	**57.4**	**40.0**	**46.9**	**36.4**
Graph Networks				
Our [InferSent]	61.9	36.2	46.4	33.4
Ours [SkipThought]	60.2	36.4	45.6‡	34.0

Table 5.1: Comparison of different approaches on the validation set of the TQA benchmark. ‡ These two results cover the same model.

grade curricula. This benchmark has a wide span of lessons comprising life science, earth science, and physical science textbooks with 26k multi-modal questions that relate to the illustration, to the text content, or to both of them.

Around half of the questions are associated to the text content-only (*i.e.*, *text questions*), while the other ones include additionally an accompanying diagram (*i.e.*, *diagram questions*). While the diagram questions comprise only multiple choice, the text questions include additional true/false questions, where the only possible answers are either true or false (one only needs to decide if the input query is correct).

Implementation details. As a similarity metric σ for sorting the supporting sentences, we leverage the cosine similarity: $\sigma(x, y) := (x \cdot y)/(\|x\| \cdot \|y\|)$. We empirically choose $s = 4$ for the multiple choice setting and $s = 2$ for true/false queries. We encode each sentence to a global vector representation using different pre-trained models. If not otherwise specified, we use the SkipThought [Kiros et al., 2015] encoding, while for some experiments, we also provide results for InferSent [Conneau

Modality	Accuracy
Baselines	
Image-only	33.2
Text-only	33.9
Multi-Modal QA	
Graph baseline	29.2
Graph-only	33.3
Text + Image	34.0

Nr. of supp. entities	Accuracy
Diagram-only	
2	33.3
3	32.9
Text-only	
3	33.8
4	33.9
5	33.4

Table 5.2: Results on the diagram questions (Diags.) on the validation set of TQA for different modalities and number of supporting entities.

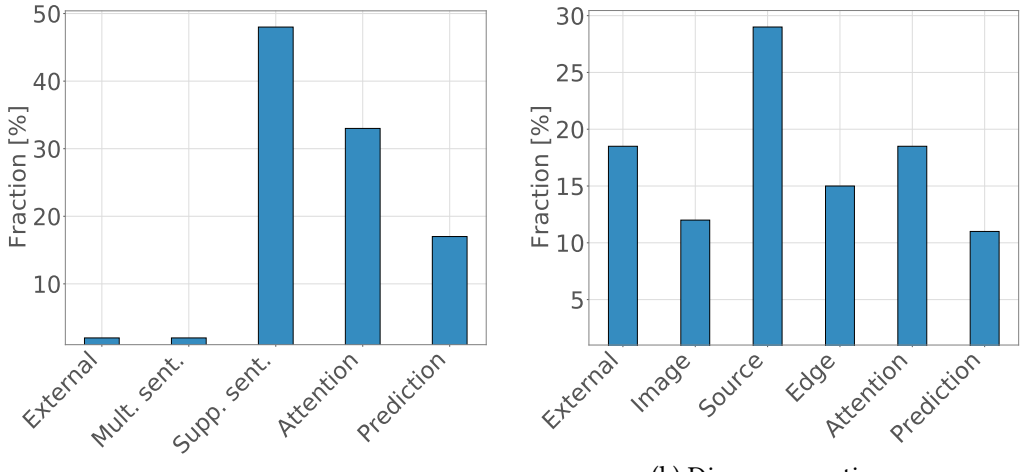
et al., 2017]. The image we represent using a Residual Network [He et al., 2016] pre-trained on ImageNet [Deng et al., 2009]. Our model is trained using Adam [Kingma and Ba, 2015] for stochastic optimization with an initial learning rate of 0.01.

5.7 Evaluation

Next, we visualize the performance of our network in comparison with related methods and present an analysis for different parameter settings of our model.

5.7.1 Final results

As we see in Table 5.1, our network outperforms state-of-the-art in the true/false questions and obtains competitive results in the entire text-only task. In the case of the diagrams setup, while our model has a lower performance than state-of-the-art, it achieves a higher accuracy than complex models such as BiDAF. Moreover, we notice that InferSent [Conneau et al., 2017] obtains a higher accuracy in the true/false questions than SkipThought [Kiros et al., 2015]. A possible reason for such a difference in performance is the way InferSent was trained. While SkipThought is trained in an unsupervised setting, InferSent is trained in a similar scenario as the true/false task aiming to find the relation type between a pair of sentences (*i.e.*, no relation, contradiction, and entailment). Finally, our model won the text-track of the Textbook Question Answering Challenge achieving the highest performance on the test set.



(a) Text questions

(b) Diagram questions

Figure 5.3: Distribution of the problems of the model in the TQA task.

5.7.2 Comparison between modalities

In Table 5.2 (left), we show the performance of our model for three different knowledge modalities: visual, graph-based, and textual. While when using both the visual and text-based data we obtain the highest performance, the text-only setting has almost an identical accuracy. Nonetheless, our model outperforms the graph baseline independent on the modality type. From all setups of our model, the image-only version obtained the worst accuracy which, however, can be explained with the use of a CNN pre-trained on natural images and not graphs.

5.7.3 Impact of the number of supporting sentences

In Table 5.2 (right), we evaluate our model for different number of selected supporting sentences and nodes. As we see, the graph-only version has a higher performance when leveraging few supporting nodes (*i.e.*, two supporting nodes) as the three node settings show a drop in performance. In a similar manner, the text-based model shows the highest accuracy at four supporting sentences, while selecting five sentences produces a drop of around 0.5%. This decrease in performance of the network when using a higher number of supporting entities is presumably due to overfitting, since we pass a larger amount of data to our model.

Question
Human actions that increase the risk of soil loss include

Answers
(a) Logging ✓
(b) Terracing
(c) Tree planting
(d) No tree planting

Supporting Sentences
(1) Human actions that can increase soil erosion are described below
(2) Other human actions that put soil at risk include logging, mining, and construction
(3) Show how farming practices can increase soil erosion
(4) There are several other ways to help prevent soil loss

Figure 5.4: Example where the attention focuses on an unimportant sentence.

5.8 Error analysis

In this section, we discuss the properties of our model and show the sources of errors which we categorize based on the parts of the network that produced them.

Text-based task. To get a better overview of the common errors of our model, we group them into the following categories (Figure 5.3 left): (1) necessity of external knowledge to answer the question (*External*), (2) the required information spreads over more than one sentence (*Mult. sent.*), (3) the supporting sentences selected by our model do not contain the relevant one (*Supp. sent.*), (4) the attention module failed to attend to the relevant sentence (*Attention*), and (5) the prediction module was not able to provide the correct answer (*Prediction*).

In Figure 5.3, we show the distribution of the problem types for true/false and multiple choice questions for 100 randomly picked questions in the dataset. For true/false, most of our mistakes are due to the prediction module, followed by the supporting sentence and attention module. Finding if two sentences are contradictory or have the same statement is a difficult task, especially when we have multiple statements in a single sentence. Finally, Figure 5.4 visualizes an example of an error caused by the attention module (*i.e.*, *Attention*-type error).

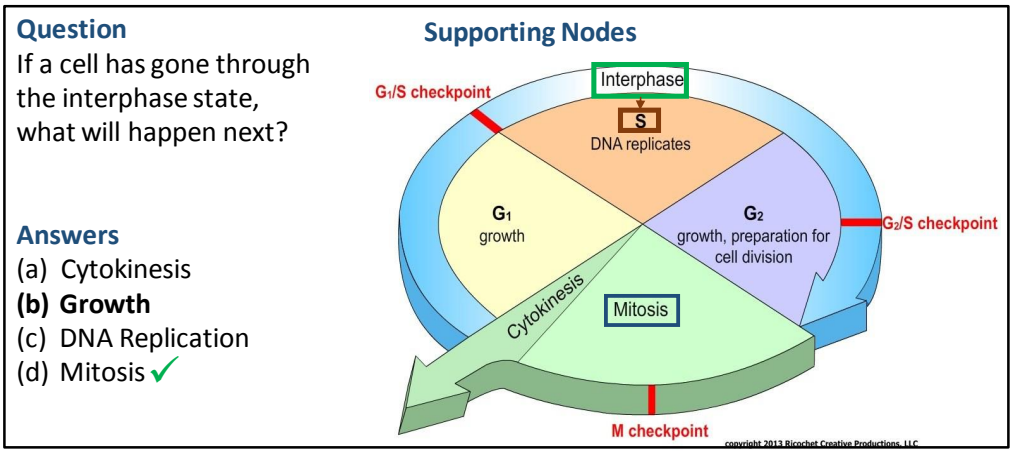


Figure 5.5: Example, where our approach selected the correct source node (green), but failed to select the correct target node (in red the predicted target node and in blue the correct one). This category we call 'Edge' in the results table.

Diagram-based task. Next, we analyze the diagram-based question types where we group the errors into: (1) necessity of *external knowledge* (*External*), (2) the required information is in the visual data (*Image*), (3) the supporting node selected by our model does not contain the relevant information (*Source*), (4) the source and edge nodes are incorrect (*Edge*), (5) the attention module failed to focus on the relevant sentence (*Attention*), and (6) the prediction module did not provide the correct answer (*Prediction*). An example of the *Edge* error type is shown in Figure 5.5.

Table 5.3 (right) visualizes the distribution of errors our approach produces in the graph-only setting. Similar as for the text-based questions selecting the supporting entities causes many mistakes (*i.e.*, *Source*). Thus, this indicates that a better selection process can greatly improve performance by better filtering unimportant information. Including more (supporting) nodes to the network is a possible solution, however, as we showed in Table 5.2 this leads to overfitting. While the error analysis shows that we can achieve at most 13% improvement through the visual modality, our results indicate that additional visual information only slightly improves the performance of our network. A reason for this slight increase in performance (compared to the 13%) is that our model was not able to learn complex visual features from such few training samples. On the other hand, as almost 20%

of questions are incorrectly answered due to the lack of external knowledge and the network shows strong results on the text-based task, augmenting the method with the surrounding text has a high potential to further improve performance.

5.9 Summary and discussion

In this chapter, we introduced a novel neural architecture for multi-modal question answering in a multiple choice setup. While soft paths assumes that the input query comprises all necessary information to traverse the graph *without a priori knowledge* of its data, in this chapter, the model pre-processes the knowledge base and then decides where to attend. Our approach first selects relevant information based on the current question which are passed to our neural network which gives the final decision which entities are relevant for predicting the answer. We compare the network for different knowledge modalities: text-, image-, and graph-based, and show that the textual model has the best performance in all tasks. Furthermore, we analyze the mistakes that our model produces and uncover the difficulties that our model encountered. Finally, our approach won the first place in the Textbook Question Answering Challenge where it obtained the best performance on the test set of TQA.

6 Visual Reasoning on Pages

Imagine looking for a specific answer in a technical document. Especially if one is new to the material, it is hard to keep all information in mind and one spends time searching for the right answer. Besides saving precious time of an average user, automatic visual reasoning on documents comes with immerse benefits for visually impaired people, detaching the accessibility of knowledge from the biological vision. Automatic understanding of documents can be posed in the form of visual question answering [Agrawal et al., 2017]. Such models mostly focus on scenes in natural images and have experienced major progress in recent years, due to advances in natural language processing [Conneau et al., 2017; Pennington et al., 2014; Vaswani et al., 2017] and computer vision [Huang et al., 2017; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015], where novel architectures significantly improved the representations of both, images and text. While there are few works addressing figures, they are restricted to a very specific entity type, such as diagrams or tables [Jauhar et al., 2016; Kembhavi et al., 2016]. Presumably due to the insufficient datasets for training these models, the research of visual reasoning on complete pages is far behind.

While other QA methods address different figures (*e.g.*, photographs, diagrams) individually in the previous chapters, the challenges for their understanding are

not the same. Interpreting *natural images* requires robustness to changes in illumination and shading, approaches for reasoning on *diagrams* face highly-structured data and should incorporate external knowledge, while *plots* entail other difficulties as data points include real numbers. Although previous models attempt to interpret a single graphical element type, many applications, such as information retrieval from pages, would require understanding both the underlying page components (*e.g.*, title, tables) and their interplay. While humans are excellent in putting all these page elements together and have a natural grasp of how they relate, automated visual reasoning over such compositional pages is a wide-open research question.

While in the previous chapters, we focused on answering questions related to different page entities (either figures alone or combining text and graphical elements), such focused representation of the data strongly restricts the possible question types. In Chapter 4, we followed soft paths on figures of a fixed type where the information necessary to answer the current question lied in the given input image. Thus, we assumed that the input query is associated to a *single* and *known* knowledge source. In case of MoQA in Chapter 5, the relevant information either lied in the (available) diagram or the associated text content and, thus, the model basically needed to decide between the two modality types. However, such assumptions frequently do not hold, as questions can relate to several components (*e.g.*, comparing the semantic information in two figures) or can be based on the overall structure of the page (*e.g.*, counting the number of figures on a page). Thus, extracting different entities such as natural images from the document or even discarding the overall structure of the page strongly restricts the type of questions our models can handle.

We address these shortcomings and aim to bring VQA to a setting, where synchronous understanding of different types of graphics and their interaction is crucial to find the correct answer. To that end, we introduce the problem of *visual reasoning on pages* (Figure 6.1). Specifically, we focus on question answering on *presentation slides* where properties such as high variety in layout, content type, figure-to-text ratio, diverse placement, and enhancements of the graphical elements make this document type especially hard for the VQA algorithms.

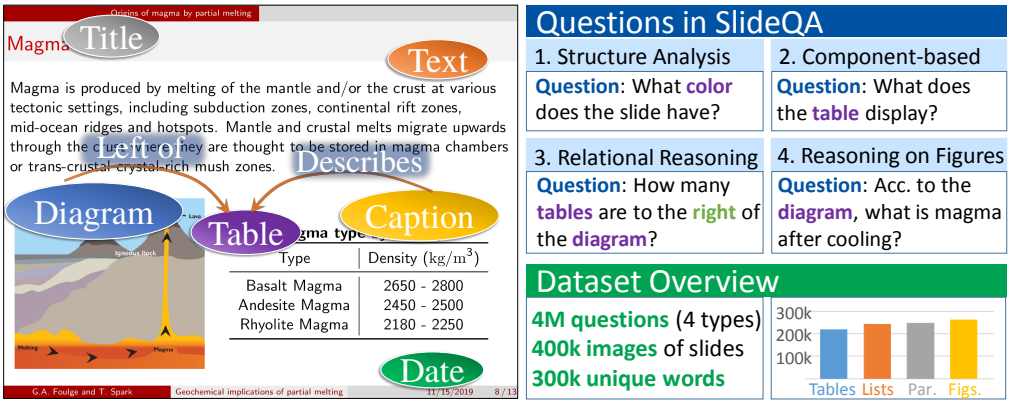
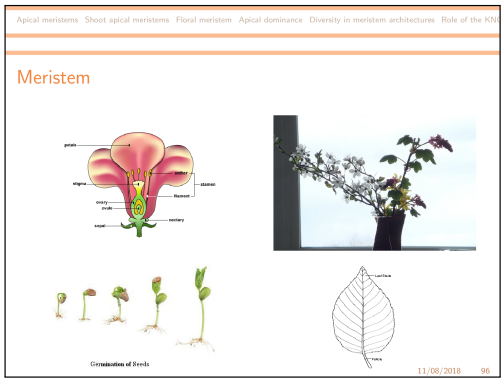


Figure 6.1: Overview of the slide question answering task with an example page and its corresponding structure (left), example questions grouped into four representative classes (top-right), and statistics of our dataset (bottom-right).

From the application point of view, presentation slides are a wide-spread messaging pathway in business and education, used for exchanging knowledge between people (*e.g.*, providing learning material for students) and accompanying oral talks. Still, understanding presentation slides has been barely touched by research with only three published works addressing fundamental computer vision problems related to them: slide retrieval from videos [Araujo et al., 2015] and slide segmentation which we introduced in Chapter 3.

Contributions. In this chapter, we collect the *SlideQA* dataset comprising 4M questions on 400k slide images. Solving the *SlideQA* reasoning tasks requires simultaneous interpretation of multiple types of page components (*i.e.*, diagrams, photographs), structural elements (table and enumerations) and text categories (*e.g.*, title, slide number, date, affiliation) and understanding their interplay. To produce a high diversity of slides and increase difficulty, we propose a new collection strategy that combines multiple knowledge sources from the internet to further generate realistic slides using LaTeX. We collect text, enumerations, and tables from Wikipedia, natural images from the popular VQA-v2 dataset [Goyal et al., 2017], and semantic information about them from GQA [Hudson and Manning, 2019], image captions from the COCO [Lin et al., 2014] and diagrams from the AI2D [Kembhavi et al.,

Q: Which **figure** contains more people?
A: The right one.



Q: How many **diagrams** are **lower** than the **natural image**? **A:** 2

Figure 6.2: Examples of questions and slides with different page component types (*e.g.*, natural images, page number) from the SlideQA dataset.

2016] datasets. Content collection using authentic web-based knowledge sources assures the realism of the slide, while automatic LaTeX generation allows us to freely parameterize the slide properties facilitating high variety of component placement, colors, figures, and content type. In Figure 6.2, we show two example slides from our SlideQA dataset with associated questions.

We adopt and evaluate various baselines and off-the-shelf neural architectures for reasoning on natural images [Agrawal et al., 2017; Hudson and Manning, 2018; Yang et al., 2016]. Motivated by the vulnerability of conventional models to the large vocabulary size of our benchmark, we present FUSE, a model specialized for our task of visual reasoning on entire pages. Our model is fully-convolutional in both the visual and textual encoder and fuses the multiple modalities using an attention mechanism at *character-level*. Thereby, the network generates the answer as tokens *simultaneously* (*i.e.*, without employing any recurrent layers) and outperforms conventional VQA methods on our SlideQA dataset.

Publications. This chapter is encompassed in [Haurilet et al., 2021b].

6.1 Dataset collection

To address the lack of visual reasoning benchmarks operating on complete pages, we collect the *SlideQA* dataset featuring 400k presentation slides and 4M questions with varying degree of reasoning complexity (examples provided in Figure 6.2). In the following, we describe the collection mechanism that we have developed to reach multiple design goals: a high variety of layouts (*Section 6.1.1*), diverse interplay of graphical and textual content (*Section 6.1.2*), and semantic soundness, *i.e.*, page entities should be consistent in their topic (*Section 6.1.3*).

6.1.1 Choice of layout

To produce a high variety of layouts, we utilize multiple controls available in LaTeX. We choose uniformly from 29 different layout modes, 68 colors, 5 inner- and 3 outer-themes of the Beamer LaTeX package resulting in 30k *different layout configurations*. We define three types of slides: (1) *title-slides* covering the title and author information, (2) *overview-slides* displaying the presentation structure (*i.e.*, table of contents), and (3) *content-slides*, the most vital page type, as it carries the information about the topic. Consequently, 90% of our dataset is attributed to *content-slides*, while *title-* and *overview-slides* each form 5% of the data.

6.1.2 Graphical and textual content

Text, tables, and enumerations. To ensure the realism of our slides, we employ Wikipedia articles for collecting text, table, and enumeration data in our dataset. From the 5.7M pages available in Wikipedia, we have selected the 500k longest articles. In total, we have extracted 1.2M tables, 2M bullet points, and 6M paragraphs which constitute the basis of our slides.

Figures. We collect the figures from the public AI2D diagram benchmark [Kembhavi et al., 2016] and the VQA-v2 dataset [Goyal et al., 2017] for question answering on natural images (details in Table 6.3). To obtain image captions, we utilize the

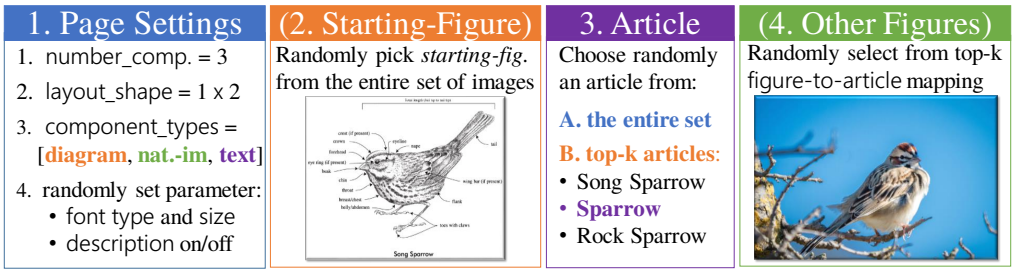


Figure 6.3: Overview of the slide generation process: (1) layout and component types selection, (2) if a figure is included in the component types, a starting-figure is randomly chosen, (3) article selection either randomly from all 500k articles (if step 2 is skipped), or based on the figure_to_article mapping (for at least one figure), (4) selecting the remaining figures (if the slide should contain multiple images).

COCO dataset [Lin et al., 2014] and randomly select one of the five available descriptions. We select these benchmarks as a basis for our dataset as they provide additionally to image captions, semantic annotations (*e.g.*, text labels in the diagrams), and control over the choice of figure types.

6.1.3 Slide generation engine

Component choice. To ensure that our slides are semantically sound, we introduce a mechanism for selecting document components that encourages the page content to fall into the same topic (overview given in Figure 6.3). First, we select the number and type of components (**Step 1** in Figure 6.3) that we include in the current slide (*e.g.*, a diagram, a natural image, and a paragraph). ↪ If no figures exist in the set of selected components, we randomly sample an article from the Wikipedia data (**Step 3.A**). ↪ Otherwise, we randomly pick a *starting-figure* (**Step 2**), and based on it, we choose an article that is semantically close using the *figure-to-article function* (**Step 3.B**) that we define as follows.

Figure-to-article mapping. For every *starting-figure*, we compute the similarity between the words in the set of captions (for natural images) or description-nodes (for diagrams) and the article titles of Wikipedia.

To that end, we iterate over all captions for the current figure as follows:

$$\sigma(\mathcal{C}, t) = \frac{1}{|\mathcal{C}|} \sum_{\mathcal{S} \in \mathcal{C}} \left(\frac{1}{|\mathcal{S}|} \cdot \sum_{w \in \mathcal{S}} \cos(w, t) \right) \tag{6.1}$$

where t is a title from an article, cos is the cosine similarity, and w is the embedding of a word (*i.e.*, wor2vec [Mikolov et al., 2013]) in the set of captions $\mathcal{C}$. We then randomly sample one of the top-k most similar articles to the descriptions of the current image and use the text content and tables in the corresponding article to produce the slide. We set a fixed threshold for the number of selected articles ($k = 40$, totaling to 8M article instances) enabling a balanced and diverse mapping, opposed to a variable threshold, *e.g.*, based on similarity values. If more figures need to be included (additionally to the starting-figure), we use a similar mapping *from the current article* to the top-k most similar figures as further described.

Article-to-figure mapping. If the number of figures in the slide is over one, we need to select additional graphical content besides the starting-figure (**Step 4**). As for the figure-to-article mapping, we select the remaining page components using the similarity function σ. In this case, we use the title of the article chosen for the current slide, to get the top-40 most similar natural images and diagrams, and randomly select the remaining graphical content.

Why not figure-to-figure mapping? An alternative method to collect the remaining figures from an article is to directly use a figure-to-figure mapping from the starting-figure to its closest images. However, this could lead to an inconsistency in the overall topic of the slide. For example, a starting-figure has the caption: 'A dog and a cat lying on the bed.' and our figure-to-article mapping picks the article 'Dog'. A figure-to-figure mapping has the possibility to focus on the word cat and select *cat images* for the remaining graphical content in the slide. This is, however, incorrect as the overall topic of the slide is *dog* and not *cat*.

Page structure. As page components are put together through our algorithm, we can directly infer the underlying structure as a *graph*, where the *nodes* stand for semantic entries (*e.g.*, title, affiliation, diagram) and *edges* represent relations between them (*e.g.*, top, describes). Note that our proposed FUSE model operates directly on the image input in an end-to-end fashion and does not require the structure graph. Still, we provide the graphs as additional annotations opening the possibility for future research of graph-based methods, such as architecture based on soft paths. We further leverage the page structure to generate the question-answer pairs.

6.1.4 QA collection process

For generating the question-answer set, we harness the layout information, page structure (*i.e.*, placement of the components and the relationship type between components), and semantic information from the figures (*e.g.*, number of cats in the image). Our collection process employs 40 *patterns* grouped into *four representative reasoning tasks* (examples in Figure 6.2), which are now discussed in detail.

1. Global structure understanding. In our first task, we address the *structure understanding* of the page image. For example, there are questions about the type of slide, the color of the slide template, and the presence of specific components, such as date or page number.

2. Questions about components. The second task is related to gaining knowledge from the slides and targets the information present in the document. It is usually linked to the *content* of captions, text, and enumerations.

3. Relational reasoning. The third group of questions shows if the network is able to *understand* the structure of the page and *to reason* about the found entries (*e.g.*, how many components are *left* of the *diagram*). To solve this task, the model has to localize the relevant page components (*e.g.*, the *diagram*) and, then, generate an answer based on the found region (*i.e.*, find how many components are left of it). Longer reasoning chains are typical for this task.

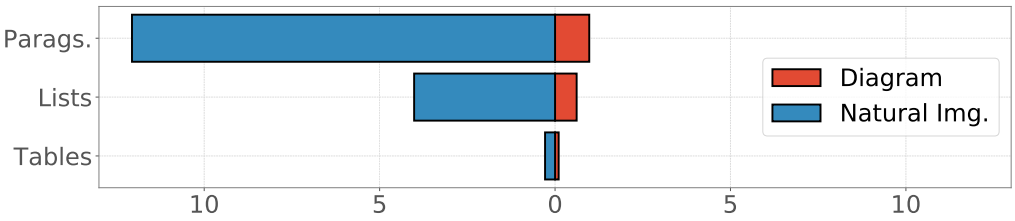


Figure 6.4: Analysis of articles dependent on the figure-mapping type. The diagram-mapping selects more frequent articles with shorter document components (paragraphs, lists, and tables) than the image-to-article function.

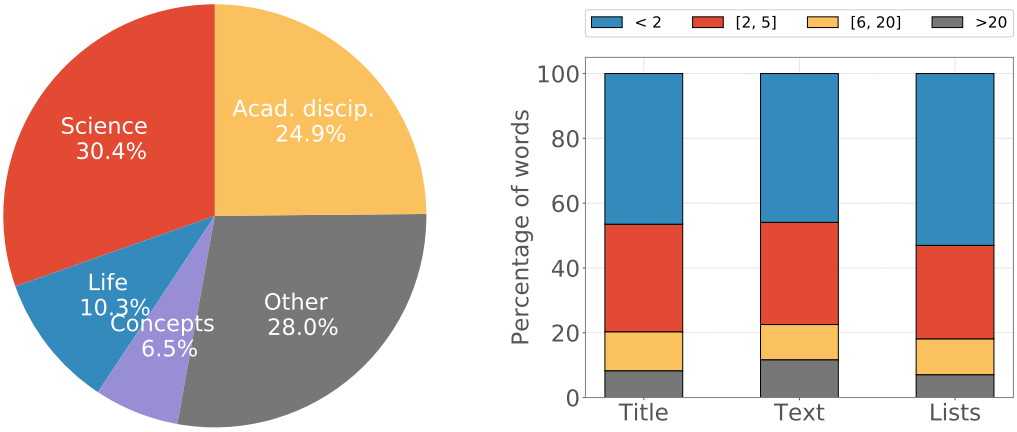
4. Reasoning on figures. Finally, the last task entails *fine-grained* understanding of the *figure content*. This task is generally harder than the *questions about components*, as it requires greater details and the models need to both select the relevant figure (*e.g.*, find the *left* diagram in the slide) and find the correct answer based on its *semantic information* (*e.g.*, counting the number of cats in an image).

6.2 Data analysis

We now elaborate on the key properties of *SlideQA* and analyze the articles selected by our mapping scheme, the generated slides and question-answer pairs. First, we discuss the properties of the figure-to-article mapping schemes as well as the key attributes of our generated presentation slides. Then, we analyze the questions and the necessary reasoning capabilities for answering the queries.

6.2.1 Articles and slides analysis

Comparison of components between page type. In Figure 6.4, we show the average number of collected tables, lists, and paragraphs by our figure-to-article mapping strategy. As we see, the articles selected by the diagram-to-article function varies in the amount of content to the articles chosen by the natural images. That is, the articles in the top-k diagram-mapping include on average more paragraphs, lists, and tables. The reason behind this lies in the diagram-mapping selection of a



(a) Main-categories generated by the the diagram-to-article mapping.

(b) Distribution of the frequency of words in the corpus for text, titles, and lists (in %).

Figure 6.5: Distribution of the articles selected by our collection scheme and analysis of the text content in the SlideQA dataset.

high number science-related articles, which are often shorter. Even though there is a large discrepancy between the different articles for the two mapping functions, since we select the document components using a uniform distribution, the slides do not include a distinction in the component types distribution.

Distribution of selected articles. Wikipedia groups its articles topic-wise into 39 major categories. We now utilize these topic groups for an initial analysis of our proposed figure-to-article function in order to gain insight into the selected articles. As the diagrams are often related to anatomy (human body), biology (food chains), and astronomy (solar system), we expect our articles to lie in the science field. Figure 6.5a depicts the major topics distribution of the articles included in the top-40 list of our diagram-to-article scheme. Our assumption is, therefore, confirmed as most articles are in the science (30%), academic disciplines (24.9%), and life sector (10.3%).

Vocabulary of the text content. An important property of QA datasets is the word distribution of the collected written content, as this directly impacts the difficulty of answering text-related questions. Figure 6.5b shows that a large fraction

Figure type	# Slides	Rows [#] / Height [cm]	Cols. [#] / Width [cm]
Diagram	**132k**	3.88 [cm]	**4.95** [cm]
Nat. Image	**132k**	**4.07** [cm]	4.73 [cm]
Table	109k	3.44 [#]	2.97 [#]

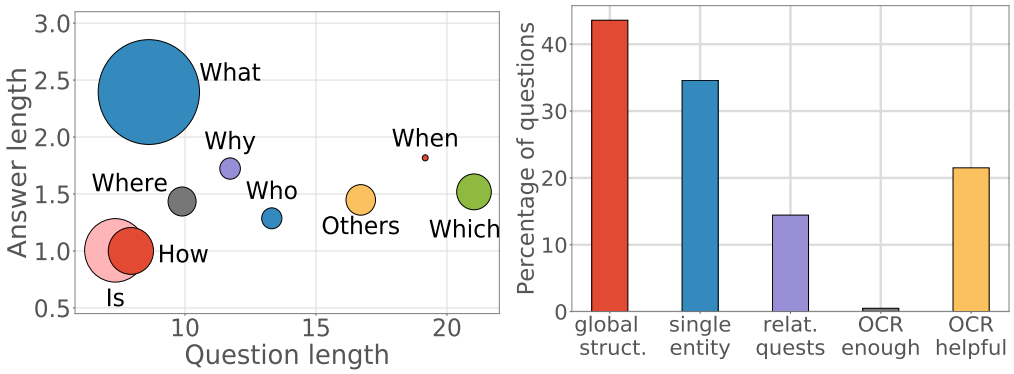
Table 6.1: Number of slides containing graphical and structural elements and their average size (in [cm] for figures and in number of rows and columns for tables).

Text type	# Slides	# Sent./Component	# Words/Sentence
Title	**400k**	1.00	1.05
Description	180k	1.63	4.77
Paragraph	124k	2.72	**21.13**
List	122k	**3.01**	12.28

Table 6.2: Number of pages including text, number of sentences for each appearance, and number of words per sentence.

of words (*e.g.*, 50% in the text-body) are *unique*. As a consequence, our answer set inevitably contains *out-of-vocabulary words* making *SlideQA* especially challenging for some visual reasoning models, as we further show in the evaluation section.

Analysis of the graphical and textual document elements. In Table 6.1 and Table 6.2, we illustrate the distinct properties of the document components covered by our dataset. There is a noticeable gap between the text classes as it comes to the amount of included text: While on average only a few words are present in *titles*, the *paragraph* class usually covers over 50 tokens, which surpasses the number of words per sentence in a title by a factor of 20. Moreover, *lists* have around half as many words per sentence as the *paragraphs*, while, on the other hand, comprising twice as many sentences per page than *figure-descriptions*. In case of the graphical components, the properties are more similar: both *diagrams* and *natural images* are frequent in the slides and more than 50% of the dataset contains at least one figure type. Due to scaling differences (diagrams have often a larger height), diagrams



(a) Queries and answer length based on quest. type (radius ≃ # questions).

(b) Distribution of questions with varying compositionality degree.

Figure 6.6: Analysis of the question and answer set in our proposed SlideQA benchmark and its comparison with related VQA datasets.

and natural images vary slightly in their average size (*i.e.*, diagrams are generally wider). Finally, since some articles do not include tables, we see a lower number of slides containing tables than natural images and diagrams.

Parsed components collected from Wikipedia. We extracted 1.2 million tables in the top 500k largest articles in the Wikipedia dump with 21 million bullet points and 6 million paragraphs. We discarded the final rows and columns for too long and too wide tables, as well as, removed sentences in the long bullet points and paragraphs. In case of enumerations, we randomly pick at most five bullet points and for paragraphs, we select the first four sentences.

Vocabulary of the words in the answers. In SlideQA, we counted 15k unique words in the *questions* and 320k *unique answers*. The questions can easily be represented using a one-hot embedding for training the networks, *i.e.*, without even reducing the query vocabulary. In contrast, due to the large number of unique answers, the prediction module would require 320k output neurons that leads to an explosion in parameters in the final layer (in the millions). Thus, to capture the entirety of the answers in the dataset, one would require a different strategy than the classification-driven strategy of current VQA models.

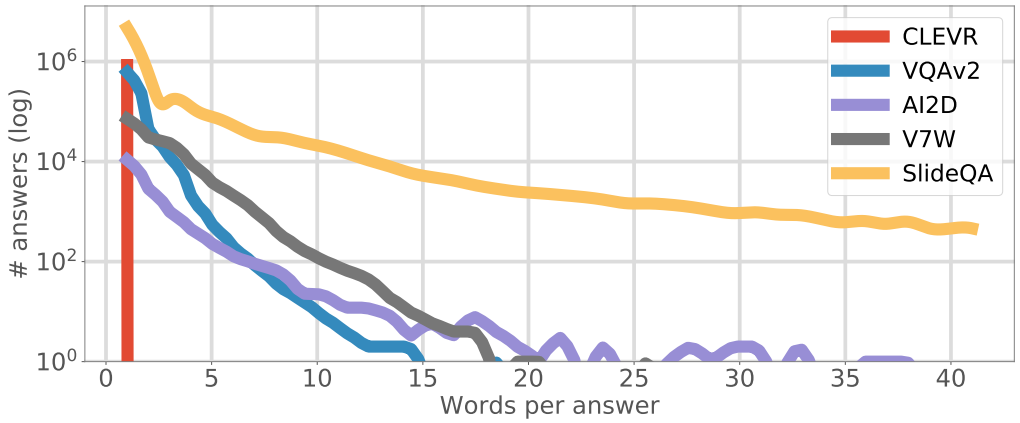


Figure 6.7: Average number of structural components (tables, lists, paragraphs) in each article selected by our mapping function for diagrams and natural images.

6.2.2 Overview of the QA set

Query-types. In total, we generate 4M open-ended questions associated with eight main query types (overview in Figure 6.6a). Most queries belong to the *which* type representing often short questions with long answers. In comparison, the questions of the *which* type are on average more than twice as long. This is not surprising, as in case of *which* the queries describe multiple attributes for discriminating between objects, while for *what* questions attributes or easily found objects are pursued. Queries of the types *how* and *is* revolving around counting and existence type questions have the shortest answer length (of around one word).

Reasoning. Figure 6.6b illustrates the percentage of questions that require different levels of reasoning complexity: (1) global structure (*i.e.*, overall layout), (2) single entity centered around a single component, and (3) relational questions. The visualization indicates that the dataset is mostly balanced in the length of reasoning chains especially in the global structure and the single-entity questions. Relational queries are less common (around 15% of the dataset), as some questions only apply if more than one component is in the page. Finally, we show that almost all questions cannot be answered by using Optical Character Recognition (OCR, *e.g.*, [Smith, 2007; Wei et al., 2018]) *alone* with OCR being helpful in less than 25% of cases.

Dataset	Reference	#Imgs	#Quest	+Text	+Comp.
Natural Images					
DAQUAR [Malinowski and Fritz, 2014]	NIPS'14	1k	12k	✗	✗
VQA [Agrawal et al., 2017]	ICCV'15	**205k**	760k	✗	✗
COCO-QA [Ren et al., 2015a]	NIPS'15	69k	118k	✗	✗
Visual7W [Zhu et al., 2016]	CVPR'16	47k	328k	✗	✓
VQA-2 [Goyal et al., 2017]	CVPR'17	**205k**	**1.1M**	✗	✗
Visual Gen. [Krishna et al., 2016]	IJCV'17	**113k**	**1.4M**	✗	✓
TextVQA [Singh et al., 2019]	CVPR'19	28k	45k	✓	✗
TallyQA [Acharya et al., 2019]	AAAI'19	**165k**	183k	✗	✗
GQA [Hudson and Manning, 2019]	CVPR'19	**113k**	**22M**	✗	✓
Geometric Forms					
S.-of-CLEVR [Santoro et al., 2017]	NIPS'17	10k	200k	✗	✓
CLEVR [Johnson et al., 2017a]	CVPR'17	**100k**	700k	✗	✓
COG [Yang et al., 2018a]	ECCV'18	**11M**	**44M**	✗	✓
Plots & Tables					
TabMCQ [Jauhar et al., 2016]	ACL'16	63	9k	✓	✓
FigureQA [Kahou et al., 2018]	ICLRW'17	**100k**	**1.3M**	✓	✗
DVQA [Kafle et al., 2018]	CVPR'18	**300k**	**3M**	✓	✗
Diagrams					
AI2D [Kembhavi et al., 2016]	ECCV'16	5k	15k	✓	✓
TQA [Kembhavi et al., 2017]	CVPR'18	13k	26k	✓	✗
Visual Reasoning on *entire* Pages					
SlideQA (Ours)	–	**400k**	**4M**	✓	✓

Table 6.3: Overview of VQA datasets. We group the datasets based on the figure types and compare them based on number of images and questions. We include if the images contain text and component annotations (*e.g.*, object instances).

Long-range dependencies in the answers. Other relevant characteristics of QA datasets are the *answer length* and the *number of unique answers*. The first property requires the algorithm to understand long-range dependencies, while the latter enforces the need for answering open-ended questions. SlideQA includes more queries with long answers than other popular benchmarks (overview in Figure 6.7) totaling to 320k unique answers. While related datasets on VQA include answers that are around 5-10 tokens on average, SlideQA encompasses answers comprising up to 20 words in length. Finally, since SlideQA comprises answers with a large number of words, we propose a novel architecture that employs self-attention to generate the answers at character-level.

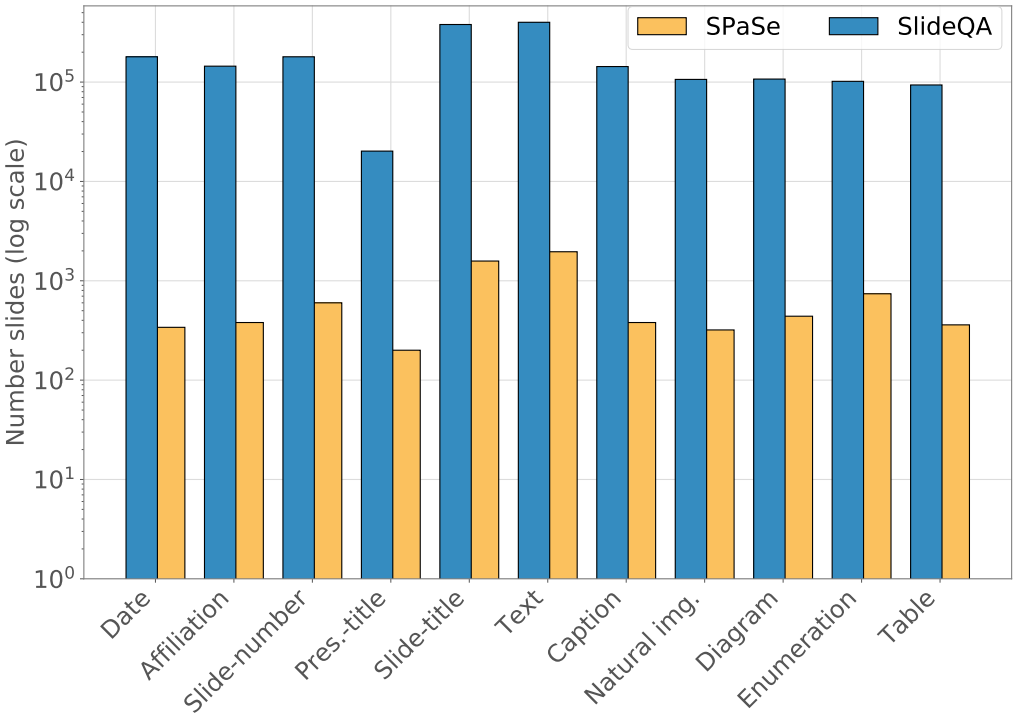


Figure 6.8: The number of slides containing different page components in our proposed SlideQA benchmark for QA on pages and our SPaSe dataset for semantic page segmentation comprising 2000 slides that we introduced in Chapter 3.

6.3 Comparison of SlideQA with related datasets

Number of images and questions. In Table 6.3, we show an overview of VQA datasets, which we group based on the visual input data type: (1) natural images, (2) synthetic images of geometric forms, (3) plots and tables, (4) diagrams, and (5) images of entire pages. Additional to the publication references, we show the number of images and questions included in these benchmarks, which we mark for a threshold of 100k in case of figures and 1M for the images. Several datasets comprise more than 100k images: While the datasets including natural images have at most 205k figures, synthetic images comprise up to 11M images. The synthetically generated FigureQA and DVQA on plots also show a large amount of questions and images totaling to over 100k images with over 1M queries. In

comparison, our SlideQA benchmark comprises 400k images with 4M questions covering four query types ranging from overall slide type to relational questions. Another key aspect of the VQA datasets is the available additional annotations of the visual content marked with 'Comp.' in the Table 6.3, as these labels enable the use of graph neural networks. Finally, we show that some datasets include figures with text and mapped labels, which are mostly available in more document-related datasets (*e.g.*, AI2D, TQA, and SlideQA).

Comparison with SPaSe. In Figure 6.8, we show the distribution (y-axis, log scale) of document components (x-axis) for our SlideQA benchmark and SPaSe, the dataset introduced in Chapter 3. As we see, the SlideQA dataset is more comprehensive in each document component type than SPaSe where we have by an order of 100 more pages in all categories. Namely, while SPaSe has around 1000 slides for each class, SlideQA includes around 100 000. Nonetheless, SlideQA shows a similarly distribution of the occurrence for the different document component classes in the pages as in our human-generated SPaSe data.

6.4 Methods

In this section, we present several baselines that we implemented on our SlideQA dataset as well as stronger attention- and graph-based networks. Then, we describe our attention-based FUSE architecture, which fuses the question and visual information using self-attention at character-level.

6.4.1 Baselines

a. ANS illustrates the prior in our dataset and predicts for all questions the most frequent answer (*i.e.*, 'Yes').

b. QTYPE predicts the most frequent answer for each question type (*i.e.*, one of the four pre-defined categories).

c. QUES is a blind model that uses as input only the question embedded by an LSTM [Hochreiter and Schmidhuber, 1997]. This approach was previously used on other VQA datasets, *e.g.*, 'deeper-LSTM' in VQA [Agrawal et al., 2017] and 'LSTM' in CLEVR [Johnson et al., 2017a].

d. IMG-only is an MLP on the visual vector-embedding extracted from a pre-trained ResNet-101 [He et al., 2016] (*i.e.*, the output of the global average pooling).

e. Q+IMG merges the vector representation of the image with the last hidden state of the LSTM applied to the question. Then, an answer is generated by two subsequent fully-connected layers on the concatenation of the two features.

f. SAN [Yang et al., 2016] employs multiple rounds of spatial attention refinement for a better image representation.

g. RN [Santoro et al., 2017] represents pairwise relations between regions in the image through concatenation with the question. The resulting relation embedding is used to obtain the answer by employing several fully-connected layers.

6.4.2 The FUSE network

Existing VQA methods usually extend the image/question encoder either with fully-connected layers [Agrawal et al., 2017; Ma et al., 2016; Malinowski and Fritz, 2014; Perez et al., 2017] or RNNs [Guo et al., 2019; Qin et al., 2019; Xu et al., 2015; Yang et al., 2019] to generate an answer. Both paradigms, however, have a downside in the challenging *SlideQA* setting. Fully-connected layers struggle in case of a high number of possible predictions or in case of out-of-vocabulary words, as they treat VQA as a classification task over the static set of answers seen during training. While RNN-based approaches can handle these issues, they are known for convergence problems especially for long-range dependencies [Graves et al., 2014; Haurilet et al., 2019d] as it is the case in SlideQA. Motivated by these limitations on our task, we adopt a different design and introduce **FUSE**, a network including

***FU**lly-convolutional* visual and textual encoders. Our network is able to generate tokens at character-level ***S**imultaneously* through a multi-modal ***E**ncoder* with multiple token-wise attention maps.

Modules. In Figure 6.9, we give an overview of our proposed architecture on an example slide and question from our dataset. The network comprises: (1) a visual encoder, (2) a question representation, (3) a multi-modal embedding scheme, and (4) a prediction module. As in several approaches for VQA [Santoro et al., 2017], (1) ***the visual encoder*** embeds the image using a 2D-CNN applied directly on the raw pixels generating a visual representation $I \in \mathbb{R}^{h \times w \times d}$. For (2) ***the question encoding***, we employ 1D convolutional layers as in our soft paths architecture (Chapter 4) ensuring a faster and more stable training procedure than in the case of RNN-based question embedding schemes. Thereby, we employ several convolutions on the words in the query and then discard the variable-sized dimension via global average pooling (*i.e.*, variable in the number of words), which results in a fixed-sized representation (*i.e.*, an f-dimensional vector $\boldsymbol{q}$).

Next, (3) ***the multi-modal module*** joins the embedded visual data with the text representation. We refine the visual data I by employing an attention mechanism in the spatial dimensions based on the input query $\boldsymbol{q}$. In particular, we generate an attention map of the size $h \times w$ that aims at localizing each token (*i.e.*, characters of the answer) in the image. Thus, we compute the kth attention map as follows:

$$A_{i,j,k} = \phi_k^\intercal \cdot [\boldsymbol{q}, I_{i,j}] + \boldsymbol{b}_k \tag{6.2}$$

resulting in a tensor A of the shape $h \times w \times t$. Thereby, we leverage the learnable parameters $\phi_k \in \mathbb{R}^{(d+f)\times t}$ and $\boldsymbol{b}_k \in \mathbb{R}^t$. Next, we apply softmax normalization on the spatial dimensions such that each matrix k slice sums to one. To obtain the final visual representation, we combine the 3D-tensor generated by the CNN in (1) and the attention maps A in (3) by *outer product* and flatten the spatial dimensions using sum, *i.e.*, $v = \sum_{h,w} A_{h,w} \otimes I^{h,w} \in \mathbb{R}^{t \times d}$. Then, we *fuse* $\boldsymbol{q}$ and the visual data v using concatenation followed by a fully-connected layer with g units resulting in a matrix m.

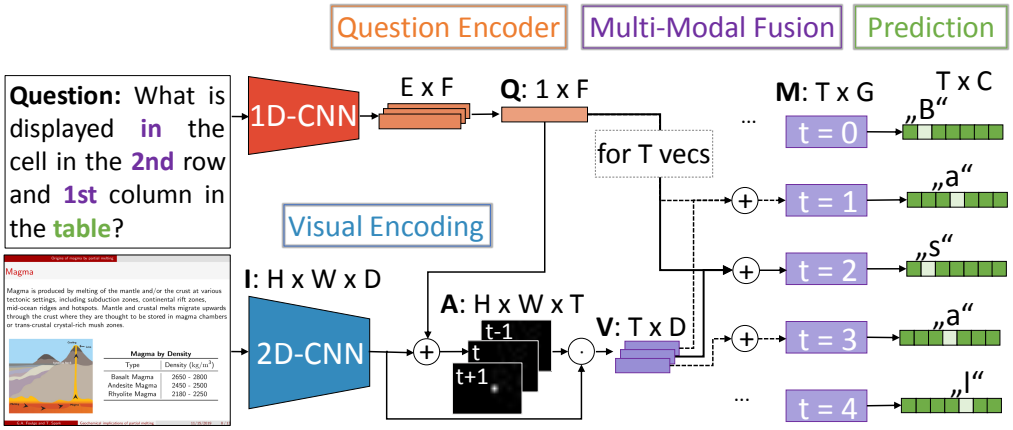


Figure 6.9: Overview of our **FUSE** network that leverage attention mechanisms on the spatial dimensions at character-level.

Finally, (4) ***the prediction module*** embeds the multi-modal representation $M \in \mathbb{R}^{t \times g}$ using a fully-connected layer with softmax normalization. As we deal with out-of-vocabulary words, the set of token classes comprises the *symbols* (*e.g.*, letters) seen during training (their number being c=150), in contrast to the conventional word-level counterpart. Thus, we obtain t probability distributions over our set of c selected symbols. To generate sequences that are shorter than the maximum number of tokens t=200, we stop when facing the specialized 'END' token to mark an end of a sentence (*e.g.*, as in [Xu et al., 2015]). We then concatenate the generated characters into a string to produce the answer (*e.g.*, *`Basalt Magma'* in Figure 6.9).

6.5 Parameter setup and learning procedure

In this section, we give an overview of the parameter setup and learning procedure of our baselines and the FUSE network.

6.5.1 Baselines

IMG. For the image-only baseline, we extract 2048-dimensional feature vectors of the slide-images from a ResNet-101 [He et al., 2016] pre-trained on ImageNet [Rus-

sakovsky et al., 2015]. These visual representations are included into an multi-layered perceptron comprising two fully-connected layers. The first layer contains 512 hidden units with ReLU activation, while the second layer models a probability distribution over the answers (*i.e.*, uses softmax normalization). Since we have a high number of answers, we remove answers that appear less than 15 times in the training set, resulting in 3970 possible answers in total. We train the model for 30 epochs using a learning rate of $1e-5$.

QUES. In case of the question-only baseline, we first embed the questions using an LSTM cell on the one-hot encoding of the words in the input query. To enable batch training, we pad the question using zero values and crop where necessary to a fixed size of 10 words per query. We use the last hidden state of the LSTM cell to obtain our final query representation and use two subsequent fully-connected layers. The first layer consists of 512 hidden units with ReLU activation, while the second generates a probability distribution over the answers (*i.e.*, the number of units is equal to the number of answers). The number of answers we keep identical to the IMG baseline. We train QUES also for 30 epochs using a learning rate of $1e-5$.

QUES+IMG. In the QUES+IMG baseline, we concatenate the 512-dimensional question embedding generated by an LSTM (as in QUES) with the 2048-dimensional visual representation (as in IMG). Then, we use a multi-layered perceptron for the prediction module comprising two fully-connected layers in the same way as in QUES: a 512-dimensional layer with ReLU activation and a prediction layer with softmax normalization. We train this baseline with the same parameter and learning settings as in QUES and IMG.

6.5.2 FUSE network

Next, we give more details about the parameters and learning procedure of the proposed FUSE architecture. In Table C.1 in Appendix C, we show an overview of the parameters and output shapes of each layer in our network.

1. Visual embedding. The generated slides have a high resolution of $1512 \times 1134 \times 3$ pixels – too large for most GPUs to handle and prone to overfitting. Thus, we resize our images to a fixed size of $168 \times 224 \times 3$ using nearest interpolation and process the pixels ranging from 0 to 255 to smaller values using the same processing setups as in ResNet. That is, the images are converted from RGB to BGR, then we zero-center each color channel according to the ImageNet dataset. We encode these images with twelve 2D-convolutional layers with ReLU activation, as described in Table C.1. We do not use any data augmentation.

2. Question encoder. We represent the question as a set of words encoded using one-hot embedding (*i.e.*, binary representation with a single one for mapping to the word in the vocabulary). Therefore, we pre-process the question by removing punctuation and transforming the characters to lower case. We use these pre-processed questions to create a dictionary of the 5k most frequent words from the training set. Due to the variable number of words per question, we are not able to apply batch training directly on this representation. Thus, we map all questions to a fixed size (25 words) by either removing words for too long queries or apply zero-padding for short ones. Finally, we apply one dimensional convolutions with ReLU activation without any data augmentation or word embeddings, as we show in Table C.1.

3. Multi-modal fusion module. In the first step of the multi-modal module, we want to *fuse* the visual and textual embeddings into a new representation that we further use to generate the attention maps. However, since the question is a vector while the visual features are in the form of a 3D tensor, we need to bring the two variables to a common shape. We achieve this by tiling the question by the height and width of the visual tensor. Then, we concatenate the tiled query with the visual feature on the last dimension resulting in a $16 \times 14 \times 160$ tensor. To obtain the attention maps, we apply multiple rounds of convolutions and, finally, use softmax on the *spatial* dimensions for each of the 200 attention maps. We want to note that the number of attention maps is equal to the number of output tokens that our model is able to generate, since each attention map focuses on parts of the image

for generating the corresponding token. Lastly, we obtain the final encoding by employing the outer product between the attention maps and multi-modal features followed by addition over the location dimensions (x- and y-axis) and resulting in a 200×160 matrix. Thus, we have a 160-sized vector encoding for each token that we want to generate for the current question.

4. Prediction network. We use three fully-connected layers to compute each token as shown in Table C.1. That is, for each token we re-embed the 160-sized vector using two fully-connected layers followed by the prediction layer with 150-hidden units. Each of the 150 hidden units represents a probability of a character class (*e.g.*, 'a', 'X', '?'). Thus, to form the final answer, for each token from the 200 available ones, we compute the index with the maximum activation from the 150 character categories. We obtain 200 indices which we simply map directly to characters based on the computed vocabulary. Since the answers have a variable length (*e.g.*, some answers are shorter than 200 tokens), we use 'END' tokens to mark the end of a string. For example, to generate the answer 'yes' our model predicts the following tokens: 'y', 'e', 's' followed by 197 'END' symbols. The 'END' symbols are discarded and we keep the three preceding characters.

Learning procedure. We train our model by minimizing the cross entropy loss using Adam [Kingma and Ba, 2015] as an optimization scheme with an initial learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-8$, and a minibatch size of 32. The weights of the convolution and fully-connected layers are initialized using Glorot distribution [Glorot and Bengio, 2010] while the biases are all set to zero. We select the questions for training pseudo-randomly: we pick a large set of images (a batch of around 16k) that are loaded into memory and we train *all* questions assigned to them (around 10 questions per image) which are picked randomly. This technique allows a faster training procedure as we strongly decrease the loading time of the large number of images into memory. We provide more information on the parameters of FUSE network in Appendix C.

Approach	Structure	Component	Relational	FigureQA	Overall
Baseline Methods					
ANS	31.7	0.0	0.0	14.2	15.0
QTYPE	31.7	21.1	30.3	13.6	26.5
IMG	31.3	0.4	0.0	14.2	15.0
QUES	69.6	27.3	57.0	24.3	49.8
Q+IMG	81.1	29.2	56.6	20.3	55.2
Attention-based and Graph Networks					
SAN	77.5	29.4	58.0	**25.0**	54.2
RN	88.8	33.1	62.4	23.7	61.0
FUSE	**89.8**	**45.0**	**65.2**	18.7	**65.6**

Table 6.4: Test accuracy in the standard split of the SlideQA dataset for each of our representative question types and the overall accuracy.

6.6 Experiments

6.6.1 Final results

We adopt the testbed of previous VQA datasets [Agrawal et al., 2017; Johnson et al., 2017a] and use the accuracy on exact-matches over all question-answer pairs in the test set as the main evaluation metric. Table 6.4 illustrates the overall performance of the employed algorithms for each task as well as the average accuracy over all four task types. Our baselines that do not use the question to generate an answer (ANS, QTYPE, and IMG) clearly underperform with accuracies under 27%. In comparison, the performance of the question-only model QUES rises to 49.8%, which is similar to the accuracy obtained by this method on other popular VQA datasets (*e.g.*, 46.8% in CLEVR [Johnson et al., 2017a], 50.5% in VQA [Agrawal et al., 2017]). When using both the question and image as input, the correct answer rate increases to an accuracy of over 53% for our VQA approaches SAN, Q+IMG, and RN. Our proposed FUSE model reaches an overall accuracy of 65.4%, exceeding all previous methods and achieving the highest performance in the three page-related tasks. An exception to this are the figure-type questions, where the leading SAN algorithm specifically designed for FigureQA reaches an accuracy of 25.0%.

Approach	Structure	Component	Relational	FigureQA	Overall
Baseline Methods					
ANS	31.7	0.0	0.0	14.3	15.0
QTYPE	31.7	21.2	30.2	13.7	26.5
IMG	31.0	0.6	0.0	13.7	14.9
QUES	69.8	27.4	57.1	22.7	49.8
Q+IMG	78.6	28.7	56.5	19.2	53.8
Attention-based and Graph Networks					
SAN	78.2	29.6	57.7	**24.6**	54.5
RN	87.8	32.3	62.1	22.3	60.1
FUSE	**89.8**	**39.5**	**64.7**	17.3	**63.5**

Table 6.5: Performance for the layout-agnostic setting (distinct layouts between the training and test set).

Approach	Structure	Component	Relational	FigureQA	Overall
Baseline Methods					
ANS	31.7	0.0	0.0	14.5	15.0
QTYPE	31.7	21.3	30.2	13.6	26.5
IMG	31.4	0.5	0.0	14.2	15.1
QUES	67.3	27.4	57.1	22.6	48.8
Q+IMG	69.7	28.8	56.2	20.4	50.0
Attention-based and Graph Networks					
SAN	74.7	29.5	57.9	**24.0**	53.0
RN	67.3	27.4	57.1	23.7	48.9
FUSE	**80.6**	**37.1**	**63.9**	17.2	**58.6**

Table 6.6: Performance of the models in the color-agnostic setting.

6.6.2 Layout- and color-agnostic setting

Next, we analyze our models in the layout and color-agnostic setting in order to target their generalization capabilities (Table 6.5). To achieve this, we split our data by layout and color such that 35% of layout and color types are excluded from training and are seen during evaluation for the first time. As expected, in both of our proposed setups, the baseline methods ANS, QTYPE, QUES, and IMG have a similar performance as in the standard setting (Table 6.4).

In case of the *layout-agnostic setting* the accuracy of the stronger VQA networks (*i.e.*, Q+IMG, SAN, RN, and FUSE) only slightly decreases (Table 6.5) demonstrating good generalization across different page types. Thereby, the largest gap between the layout-agnostic and conventional settings lies in the component-based query type, *i.e.*, the networks have difficulties in finding and analyzing document entities.

In comparison, in the *color-agnostic setting*, the color palettes which we used for training and testing are disjoint (green and blue shades are kept for evaluation-only, the rest of colors are included for training). This setup is harder for the VQA models, as we observe a higher performance drop (Table 6.6).

Finally, in both color- and layout-agnostic settings, FUSE achieves the highest overall accuracy compared to previous models totaling to: 63.5% and 58.6%, respectively. Overall, the models trained on SlideQA generalized well to previously unseen page variants, given that handling such domain shifts is challenging.

6.7 Summary and discussion

In this chapter, we introduced the task of visual reasoning on complete pages, which extends the conventional QA problem to the simultaneous understanding of different structural, graphical, and text-based elements. We addressed this task and collected the SlideQA dataset comprising over 400k slides and 4M questions spanned over four task groups. We conducted an extensive analysis of our dataset and demonstrated its unique properties and evaluated several baselines and VQA networks. Our experiments demonstrate that SlideQA tasks are difficult for modern algorithms, which we link to the diversity in component types included in the SlideQA setting as well as the large vocabulary size in the answer-set. Finally, we designed the FUSE architecture for reasoning on pages that combines convolutional layers with multiple token-wise attention maps at character-level obtaining a considerable gain in performance over previous VQA methods.

7 Generating Semantic Graphs

As we discussed in the previous chapters, graphs offer a natural way for modeling multi-step associations inside the data, leading to excellent performance of graph neural networks in tasks with long reasoning chains (*i.e.*, question answering). However, graph networks for question answering require both *inference from the graph* and *building the graph itself* from the input. While in the previous chapters we focused on reasoning using graphs, we further discuss *how to generate such graphs*.

Even though many previous question answering works partially skip the graph inference step by assuming the nodes as given through manual annotations [Kim et al., 2017; Sankar et al., 2019; Teney et al., 2016], the research of automatic graph generation is rapidly gaining attention [Garcia and Bruna, 2018; Ma and Zhang, 2019; Zhang et al., 2019]. However, existing graph generation models cannot operate on *any* kind of input modality type but are crafted for a specific task. For example, methods such as [Krishna et al., 2016; Li et al., 2017; Lu et al., 2016a; Xu et al., 2017; Yang et al., 2018b] designed for visual data usually entail object detectors as a pre-processing step, causing considerable annotation overhead in the form of bounding boxes. The detected objects then represent the nodes while the pairwise image crops around them are often used as a starting point for edge prediction. In case of visual

question answering the bounding boxes are then discarded completely and only the graph itself is used for inference. In tasks, such as graph auto-encoding [Feng and Duarte, 2018; Gidaris and Komodakis, 2019; Pan et al., 2018; Simonovsky and Komodakis, 2018], a set of nodes is passed as input and their embeddings are further refined during the encoding process. Due to the mechanism of such methods and the variable-sized dimensionality of the node set, the encoding size of the graph is directly dependent on the initial number of nodes, *i.e.*, the dimensionality of the encoding changes based on the size of the input.

Contributions. We aim to make a step towards generic graph generation from feature vectors and introduce a new approach for predicting the *entire* graph, *i.e.*, *both* the nodes and the relations *directly* from a global representation of the data. The challenge arises from the unordered nature of the nodes, which hinders the direct computation of the loss between the predicted and ground truth graph. To address this, we introduce the *Node Association Procedure* (NAP) that iteratively associates the predicted nodes and edges with the ground truth. Based on these associations, we re-order the nodes in the ground truth facilitating end-to-end training of our neural architecture. With the proposed NAP optimization scheme, the underlying network is trained to generate both nodes and edges without employing any object detectors or assuming the node set as given. Overall, our learning procedure comprises four steps: (1) generating the nodes and edges from a vector, (2) computing an assignment cost between the predicted and ground truth graphs, (3) matching the estimated nodes and edges based on the assignment cost, and (4) calculating the loss and updating the parameters of the network.

We demonstrate the potential of our approach on synthetic (Graph2Graph, MNIST-Graphs, CLEVR [Johnson et al., 2017a]) and real-world datasets (VRD [Lu et al., 2016a], Visual Genome [Krishna et al., 2016], COCO-Images, and COCO-Captions [Lin et al., 2014]) for node estimation and graph generation. While methods for visual graph generation like [Krishna et al., 2016; Li et al., 2017; Lu et al., 2016a; Xu et al., 2017; Yang et al., 2018b] rely on ground truth bounding boxes, our approach is able to estimate the nodes and edges from a *single vector* without

employing any object detection techniques. In comparison to previous methods for graph generation from text content, which build upon recurrent models to sequentially predict a node at a time, our approach generates the entire graph at once. While previous graph auto-encoders include a bottleneck layer that comprises the list of nodes (*i.e.*, a matrix with a varying dimension dependent on the number of nodes), our approach flattens the entire graph into a *single fixed vector*, which then is used to re-assemble the input graph.

Publications. This entire chapter is based on [Haurilet et al., 2021a].

7.1 Task overview

In this chapter, we address the problem of building *complete* graphs (*i.e.*, both nodes and edges) from single feature vectors, which we denote as *semantic graph generation*. As a preliminary, we define the semantic graph data structure and provide a formal definition of the task. We then proceed with our proposed learning scheme and the accompanying model in Section 7.2. We include an overview of all the variable and function definitions throughout this chapter in Table 7.1.

Semantic graph. First, we describe the target entity for graph generation – the *semantic graph*. This data structure refers to a fully-connected graph $\mathcal{G} = (\boldsymbol{v}, E)$ with semantic information associated to the n vertices $\boldsymbol{v}$ and to the edges E. Let $\mathcal{C} = \{1, \ldots, c\}$ be a set of c categories and $\mathcal{R} = \{1, \ldots, r\}$ be a set of r relationship types. We enrich the graph with semantic information and constrain that the nodes and edges take values in these finite semantic sets: $\boldsymbol{v} \in \mathcal{C}^n$ and $E \in \mathcal{R}^{n \times n}$.

Semantic graph generation. We define *semantic graph generation* as the task of finding a function Ψ that maps the input to its underlying semantic graph:

$$\Psi : \mathcal{D} \rightarrow \bigcup_{n \in \mathbb{N}} \left(\mathcal{C}^n \times \mathcal{R}^{n \times n} \right) \tag{7.1}$$

where $\mathcal{D}$ is the set of data samples, *e.g.*, $\mathcal{D} \subseteq \mathbb{R}^{h \times w \times 3}$ for images where h and w stand for the height and width of the image, respectively.

Input Data and Graph Representation	
$\mathcal{G} = (\boldsymbol{v}, E)$	Graph comprising a tuple of nodes and edges
$\boldsymbol{v} \in \mathcal{C}^n$	Vector with n entries representing the vertices in $\mathcal{G}$
$\boldsymbol{v}_i \in \mathcal{C}$	Class associated to the ith vertex
$n \in \mathbb{N}_0$	Number of nodes in the graph $\mathcal{G}$
$\mathcal{C} = \{1, \dots, c\}$	Set of c semantic node classes
$c \in \mathbb{N}$	Number of node categories
$E \in \mathcal{R}^{n \times n}$	Edge matrix of size $n \times n$
$E_{i,j} \in \mathcal{R}$	Class of edge between the ith and jth vertex
$\mathcal{R} = \{1, \dots, r\}$	Set of r semantic edge classes
$r \in \mathbb{N}$	Number of relation categories
$\mathcal{G}^* = (\boldsymbol{v}^*, E^*)$	Ground truth graph of nodes and edges
$\boldsymbol{v}^* \in \mathcal{C}^{n^*}$	Vertex vector with n^* entries
$n^* \in \mathbb{N}$	Number of nodes in the graph $\mathcal{G}^*$
$E^* \in \mathcal{R}^{n^* \times n^*}$	Ground truth edge matrix of shape $n^* \times n^*$
G^*	Set of all ground truth graphs $\mathcal{G}$ used during training
$\bar{\mathcal{G}} = (\bar{\boldsymbol{v}}, \bar{E})$	Supergraph of $\mathcal{G}$ with $\bar{n} - n$ additional `empty` vertices
$\bar{\boldsymbol{v}} \in \bar{\mathcal{C}}^{\bar{n}}$	Extension of the vector V with $\bar{n} - n$ `empty` vertices
$\bar{E} \in \mathcal{R}^{\bar{n} \times \bar{n}}$	Extension of the edge matrix to comply with $\bar{\mathcal{G}}$
$\bar{\mathcal{C}} = \mathcal{C} \cup \{c + 1\}$	The category node set with an additional `empty` class
$\bar{n} \in \mathbb{N}, \bar{n} \geq n$	Number of nodes in the supergraph $\bar{\mathcal{G}}$ of $\mathcal{G}$
Network Structure	
$\mathcal{D}$	Set of all data samples
$d \in \mathcal{D}$	A data point from the dataset
$\gamma : \mathcal{D} \to \mathbb{R}^m$	Global vector representation of a data sample
$\phi_i : \mathbb{R}^m \to \mathbb{R}^f$	Node-specific representation defined $\forall i \in \{1, \dots, n\}$
$\nu : \mathbb{R}^f \to [0, 1]^{c+1}$	Mapping of each ϕ_i to a node class distribution
$\varepsilon : \mathbb{R}^f \times \mathbb{R}^f \to [0, 1]^r$	Mapping of the node encodings to edge confidence values
Θ^τ	Set of all parameters at iteration τ in our network
$\Theta_i^{\gamma \to \phi} \in \mathbb{R}^{f \times m}$	Weights in the function ϕ_i that encodes γ
$\Theta^{\phi \to \nu} \in \mathbb{R}^{(c+1) \times f}$	Weight matrix in the function ν that maps ϕ_i to objects
$\Theta^{\phi \to \varepsilon} \in \mathbb{R}^{r \times 2f}$	Weight matrix for estimating the relation distribution
$m, f \in \mathbb{N}$	Hyper-parameters in our network
Matching Cost	
$\bar{n}^* \in \mathbb{N}$	We expand the ground truth graph to match the prediction
Note that: $\bar{n} = \bar{n}^*$	We use $\bar{n}$ and $\bar{n}^*$ to mark indexing differences
κ^V	Cost for matching the nodes based on the node activation
κ^E	Cost for matching the nodes based on the edge confidence
κ^G	Cost based on both the node- and edge confidence
Node Association and Loss Function	
$A \in \{0, 1\}^{\bar{n} \times \bar{n}}$	Association between the prediction and ground truth
$\bar{\boldsymbol{v}}^A \in \bar{\mathcal{C}}^{\bar{n}}$	Vertex vector $\bar{\boldsymbol{v}}^*$ ordered by the matrix A
$\bar{E}^A \in \mathcal{R}^{\bar{n} \times \bar{n}}$	Edge matrix $\bar{E}^*$ ordered by the matrix A
$\bar{\mathcal{G}}^A = (\bar{\boldsymbol{v}}^A, \bar{E}^A)$	Ground truth graph with ordered nodes and edges
$\mathcal{L}^V : [0, 1]^{c+1} \times \bar{\mathcal{C}} \to \mathbb{R}$	Classification loss of the predicted and ground truth nodes
$\mathcal{L}^E : [0, 1]^r \times \mathcal{R} \to \mathbb{R}$	Classification loss for the edges

Table 7.1: Overview of the variables and functions in this chapter.

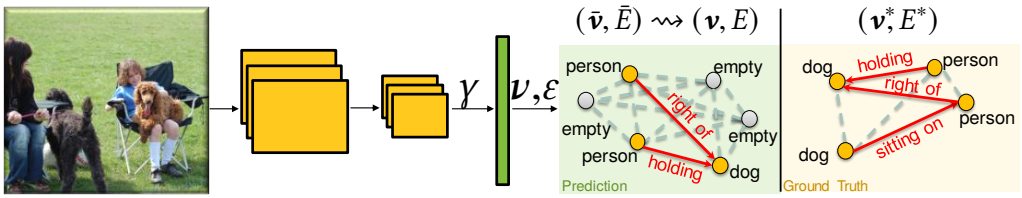


Figure 7.1: Overview of our neural architecture. We embed the input data into a single global vector γ using convolutions, fully-connected layers, or graph convolutions. Then, we inflate the vector into the node $\bar{\boldsymbol{v}}$ and edge classes $\bar{E}$ by selecting the elements with maximum values in ν and ε, respectively. During inference, we remove all vertices classified by the network with the `empty` class (marked gray) and keep the rest of the graph as our prediction $(\boldsymbol{v}, E)$. To the right, we show an example of a ground truth graph $(\boldsymbol{v}^*, E^*)$ that contains a different number of nodes and edges than our prediction.

7.2 Methodology

Conceptually, our graph generation model comprises two steps: (1) *supergraph generation* (Section 7.2.1), where the given input data is embedded into a global vector and then inflated into an enlarged set of nodes with their associated relationships, and (2) *graph inference* (Section 7.2.2), which reduces this oversized representation to the essential underlying graph $\mathcal{G}$. As previously mentioned, computing the loss directly is hindered by the order mismatch between the generated and the ground truth nodes. To address this, we introduce the *NAP learning scheme* (Section 7.2.3), which computes a fitting assignment between the nodes and, therefore, enabling supervised training of our model.

7.2.1 Architecture for graph generation

Extending the graphs. To enable batch training and to compare the prediction with the target graph as required during training, we want to bring our graph output to a fixed encoding size. To that end, we estimate a large wrapping graph representation – a *semantic supergraph* with an arbitrary but fixed number of nodes (example in Figure 7.1). Note, that we compute the supergraph of *both the prediction and the target* requiring that they match in size and, thus, enabling comparison of their entities.

Algorithm 1: Network optimization through the Node Association Procedure (NAP)

Input: data samples $\mathcal{D}$, graphs G^*, number epochs τ_{total}
Initialization: Θ^0
// iterate over epochs
for $\tau \in \{1, \ldots, \tau_{total}\}$ **do**
  // iterate over graphs or graph batches
  for $(d, \mathcal{G}^*) \in (\mathcal{D}, G^*)$ **do**
    // estimate the graph (1)
    Calculate feature vector $\gamma^{\Theta^\tau}(d)$;
    Infer the node confidences $\nu_i^{\Theta^\tau}$ using Eq 7.3;
    Get their relation confidences $\varepsilon^{\Theta^\tau}$ (Eq 7.4);
    // get the matching cost (2)
    Compute the matching costs (Eq 7.9 and Eq 7.10);
    Calculate total cost $\kappa^{\mathcal{G}}$ as in Eq 7.11;
    // associate the nodes (3)
    Get node association matrix A (Eq 7.12);
    // update weights (4)
    Calculate the loss between $\mathcal{G}$ and $\mathcal{G}^*$ (Eq 7.15);
    Update parameters to $\Theta^{\tau+1}$;
  end
end

We refer to $\bar{\mathcal{G}} = (\bar{\boldsymbol{v}}, \bar{E})$ as a semantic supergraph of $\mathcal{G}$ with $\bar{n}$ vertices, if it extends $\mathcal{G}$ with $\bar{n} - n$ vertices so that all its vertices and edges have the same associated classes. Formally, we constrain that the hyper-parameter $\bar{n}$ is at least n, *i.e.*, $\bar{n} \geq n$ and that an injective function α (for graph-to-supergraph node mapping) exists with $\alpha : \{1, \ldots, n\} \rightarrow \{1, \ldots, \bar{n}\}$, $\bar{\boldsymbol{v}}_{\alpha(i)} = \boldsymbol{v}_i$, and $\bar{E}_{\alpha(i),\alpha(j)} = E_{i,j}$, $\forall i, j \in \{1, \ldots, \bar{n}\}$. Note that the function α is bijective if and only if $\bar{n} = n$. Moreover, we restrict that the additional vertices of $\bar{\mathcal{G}}$ are of a specialized `empty` class which we map to the index $c+1$. We therefore extend the set of semantic node classes to $\bar{\mathcal{C}} := \mathcal{C} \cup \{c+1\}$. Thus, we require that the additional nodes (that are not selected by α) are mapped to $c+1$, *i.e.*, $\bar{\boldsymbol{v}}_i = c+1, \forall i \in \{1, \ldots, \bar{n}\} \setminus \alpha(\{1, \ldots, n\})$. With these constraints, we can model graphs up to an order of $\bar{n}$.

Node recognition. Let $\gamma : \mathcal{D} \to \mathbb{R}^m$ be a function mapping the input data $d \in \mathcal{D}$ to a *global vector representation* (*e.g.*, using a CNN followed by average pooling). Note that, for ease of notation, we skip the inputs to previously defined functions (*e.g.*, we use γ instead of $\gamma(d)$). We further leave out possible parameterizations, which are marked with Θ, from the function definitions.

To map the vertices to their respective class, we estimate the confidence for each of the $\bar{n}$ nodes being assigned to each class in $\bar{\mathcal{C}}$. First, we transform the global vector γ to the *node-specific features* ϕ:

$$\phi_i : \mathbb{R}^m \to \mathbb{R}^f,\ \phi_i(\gamma) \coloneqq \mathrm{ReLU}(\Theta_i^{\gamma \to \phi} \cdot \gamma) \tag{7.2}$$

where i iterates over all $\bar{n}$ vertices and $\Theta_i^{\gamma \to \phi} \in \mathbb{R}^{f \times m}$ is a weight matrix where f is a fixed hyperparameter. Note that we use a different weight matrix $\Theta_i^{\gamma \to \phi}$ for each vertex in the supergraph. We leverage the supergraph representation $\bar{\mathcal{G}}$ at this point ensuring a fixed number of node-specific transformations ϕ_i. We then apply a second mapping to each node-specific feature ϕ_i followed by softmax normalization to obtain the *node-wise distributions* ν over the classes $\bar{\mathcal{C}}$ (the extended category set):

$$\nu : \mathbb{R}^f \to [0,1]^{c+1},\ \nu(\phi_i) \coloneqq \mathrm{softmax}(\Theta^{\phi \to \nu} \cdot \phi_i) \tag{7.3}$$

where $\Theta^{\phi \to \nu} \in \mathbb{R}^{(c+1) \times f}$ is a weight matrix. We include the additional category that marks that no semantic class is assigned to the node (*i.e.*, the previously defined `empty` node category) leading to $c + 1$ possible classes. For simplicity of notation, we shorten $\nu(\phi_i)$ to ν_i throughout this chapter.

During inference, we assign to each vertex i the class with the highest probability score estimated by the probability distributions ν_i (more details on the node inference in Section 7.2.2). Note, that the parameters $\Theta_i^{\gamma \to \phi}$ and $\Theta^{\phi \to \nu}$ are *learned* jointly using our NAP learning scheme (Section 7.2.3).

Edge recognition. Next, we leverage the previously found node-specific features ϕ_i to estimate the *edge-embeddings* ε. To that end, we re-encode the feature

vectors ϕ_i pair-wise and map them to a probability distribution estimate over the possible relationship types $\mathcal{R}$ as follows:

$$\varepsilon : \mathbb{R}^f \times \mathbb{R}^f \rightarrow [0,1]^r,\ \varepsilon(\phi_i, \phi_j) \coloneqq \text{softmax}(\Theta^{\phi \rightarrow \varepsilon} \cdot (\phi_i \parallel \phi_j)) \tag{7.4}$$

where $\phi_i \parallel \phi_j$ denotes the concatenation between ϕ_i and ϕ_j, $\Theta^{\phi \rightarrow \varepsilon} \in \mathbb{R}^{r \times 2f}$ is a learnable weight matrix, and the indices take values in $i, j \in \{1, \ldots, \bar{n}\}$. In a similar manner as for the vertices, we use $\varepsilon_{i,j}$ for $\varepsilon(\phi_i, \phi_j)$. The softmax normalization is applied over all possible relationship types of every node pair (i, j), *i.e.*, modeling a probability distribution over the r output values.

Concerning node-specific features. The node-specific features ϕ_i have a special role in our approach, since we aim for a joint semantic space as an intermediate representation for inferring both the node classes and their relationship types. The global vector is, therefore, mapped to ϕ_i, which is associated to a node with a unique index i in the graph, while the edges are inferred using these same ϕ_i through pair-wise combination. An important effect of modeling the edge-embeddings ε pair-wise from the node-specific features is that only a *single* weight matrix $\Theta^{\phi \rightarrow \varepsilon}$ is needed for all edges. Alternatively, one could estimate ε directly from the global vector γ. However, this would lead to an explosion in parameters as it requires a *distinct* weight matrix for each node-pair in the supergraph.

7.2.2 Graph inference

The previously described steps infer a supergraph $\bar{\mathcal{G}}$, *i.e.*, a wrapper representation, that might include additional unnecessary nodes and edges. Next, we describe the graph inference process, where the placeholder nodes are removed, therefore, refining the final graph $\mathcal{G}$ (as in Figure 7.1). To each node with index i, we assign the class that scores the highest confidence value in the distributions ν_i:

$$\bar{\nu}_i \coloneqq \arg\max_{k \in \bar{\mathcal{C}}} \nu_{i,k} \text{ where } i \in \{1, \ldots, \bar{n}\} \tag{7.5}$$

We obtain the final vertices $\boldsymbol{\nu}$ by discarding the ones associated to the `empty` class

(*i.e.*, to class index $c+1$) in the vector $\bar{\boldsymbol{v}}$. Let $\mathcal{I}$ be the set of indices of the non-`empty` vertices in $\bar{\boldsymbol{v}}$, *i.e.*, $\mathcal{I} \coloneqq \{i \in \{1,\dots,\bar{n}\} \mid \bar{\boldsymbol{v}}_i \neq c+1\}$ and let $\iota : \{1,\dots,|\mathcal{I}|\} \rightarrow \mathcal{I}$ be a bijective function. Then, we estimate $\boldsymbol{v}$ as:

$$\boldsymbol{v}_j \coloneqq \bar{\boldsymbol{v}}_{\iota(j)} \text{ where } j \in \{1,\dots,|\mathcal{I}|\} \tag{7.6}$$

To refine the edges, we leverage the estimates of the function ε and select the relationship type associated to the strongest activation by considering only the edges that do not comprise any `empty` nodes:

$$E_{i,j} \coloneqq \arg\max_{r \in \mathcal{R}} \varepsilon_{\iota(i),\iota(j),r} \text{ where } i,j \in \{1,\dots,|\mathcal{I}|\} \tag{7.7}$$

In a similar manner, we can also estimate the edge classes of the supergraph $\bar{E}$:

$$\bar{E}_{i,j} \coloneqq \arg\max_{r \in \mathcal{R}} \varepsilon_{i,j,r} \text{ where } i,j \in \{1,\dots,\bar{n}\} \tag{7.8}$$

While we define our output for fully-connected graphs only, we can enable the model to predict sparse graphs by expanding the relation categories $\mathcal{R}$ with an additional `no-relation` class that is discarded if it produces the highest activation.

7.2.3 NAP learning scheme

At the heart of our approach is the Node Assignment Procedure (NAP) – a parameter optimization scheme that enables our network to train end-to-end (Algorithm 1). The NAP method comprises a node and an edge cost, which are used to match each ground truth node to a predicted vertex. These assignments are, then, leveraged to compute the loss of the previously described graph generation network.

Node matching cost. Due to the arbitrary ordering in prediction and ground truth, we cannot establish a direct matching between the vertices. To address this, we define a matching confidence for associating each node in the ground truth to a node in the prediction. We aim for linking each predicted node with index i with a ground truth vertex with index i^* if and only if they portray the same entity. Note that by definition, the size of the ground truth is equal to the

predicted supergraph (*i.e.*, $\bar{n}^* = \bar{n}$). To generate these connections, we define a *cost function* $\kappa^{\mathcal{G}}$ that models the confidence of i and i^* being associated. The cost function $\kappa^{\mathcal{G}}: \{1, \dots, \bar{n}\} \times \{1, \dots, \bar{n}^*\} \to \mathbb{R}$ takes high values if it is unlikely that i and i^* belong to the same entity, and gives a low estimate otherwise.

To obtain the association cost based on the ν estimates, we leverage the class of the ground truth nodes $\bar{\boldsymbol{v}}^*$. Therefore, we set the cost to the negative of the network confidence modeled by the node distributions ν:

$$\kappa^V : \{1, \dots, \bar{n}\} \times \{1, \dots, \bar{n}^*\} \to \mathbb{R}, \quad \kappa^V(i, i^*) := -\nu_{i, \rho(i^*)} \tag{7.9}$$

where $\rho(i^*) := \bar{\boldsymbol{v}}^*_{i^*}$ is the value at index i^* in the supergraph. We estimate the cost of aligning the two vertex indices i and i^* by considering the best edge match:

$$\kappa^E: \{1, \dots, \bar{n}\} \times \{1, \dots, \bar{n}^*\} \to \mathbb{R}, \quad \kappa^E(i, i^*) := -\sum_{k^* \in \{1, \dots, \bar{n}^*\}} \max_{j \in \{1, \dots, \bar{n}\}} \left(\nu_{j, \bar{\boldsymbol{v}}^*_{k^*}} \cdot \varepsilon_{i, j, \bar{E}^*_{i^*, k^*}} \right) \tag{7.10}$$

The edge assignment cost of i and i^* considers the matching confidence of all edges originating in i^*. For all vertices with index k^*, we find a node with index j that best matches k^* in terms of the relation prediction ε and node confidence ν. The confidence values of all the best edge matches (*i.e.*, relation classes and target nodes) are added and amount to the cost of aligning i and i^*. The final assignment cost is a linear combination of both cost functions κ^V and κ^E:

$$\kappa^{\mathcal{G}}(i, i^*) := w^V \cdot \kappa^V(i, i^*) + w^E \cdot \kappa^E(i, i^*) \tag{7.11}$$

We use the weights $w^V, w^E > 0$ when estimating *both* nodes and edges, while we set $w^V > 0$ and $w^E = 0$ for node recognition-only, as we do not estimate any edges.

Estimating the graph assignment. Next, we leverage the previously defined cost function $\kappa^{\mathcal{G}}$ between the ground truth and predicted nodes to estimate the *node assignment matrix* $A \in \{0, 1\}^{\bar{n} \times \bar{n}}$. To establish that, we minimize the function:

$$\min_A \sum_i \sum_{i^*} \kappa^{\mathcal{G}}(i, i^*) \cdot A_{i, i^*} \text{ with } \sum_i A_{i, i^*} = 1, \sum_{i^*} A_{i, i^*} = 1 \tag{7.12}$$

where $i, i^* \in \{1, \dots, \bar{n}\}$ iterate over all nodes in the supergraph.

We pose the graph assignment task as the problem of minimum weight matching in bipartite graphs, where the goal is to find a mapping between two distinct node subsets that are connected by weighted edges shaped by $\kappa^{\mathcal{G}}$. For addressing this problem, we leverage the Munkres algorithm [Kuhn, 1955], a popular assignment technique previously employed in machine learning [Bewley et al., 2016; Hamuda et al., 2018; Sahbani and Adiprawita, 2016; Xu et al., 2020]. The output is a binary matrix A with $A_{i,i^*} = 1$ if the ground truth vertex index i^* is matched to i. Finally, we leverage A to re-order the ground truth graph $\bar{\mathcal{G}}^*$.

Re-ordering the nodes. To address the unordered nature of the nodes and facilitate the comparability to the prediction, we re-order the ground truth vertices using the previously estimated matrix A. Since A is a binary matrix, we re-order the node classes via multiplication: $\bar{\boldsymbol{v}}^A := A \cdot \bar{\boldsymbol{v}}^*$. In a similar manner, we re-order the edges (both rows and columns): $\bar{E}^A := A \cdot \bar{E}^* \cdot A^\intercal$.

Loss Estimation and Parameter Update. After the re-ordering phase, we compute the loss $\mathcal{L}$ of nodes and edges using cross entropy, as we deal with a classification task in both cases:

$$\mathcal{L}^V(\boldsymbol{\nu}_i, \bar{\boldsymbol{v}}_i^A) := -\sum_{h \in \bar{\mathcal{C}}} \left(\left[\bar{\boldsymbol{v}}_i^A \doteq h \right] \cdot \log(\boldsymbol{\nu}_{i,h}) \right) \tag{7.13}$$

Thereby, $[\cdot]$ is the indicator function that returns the value one if the given condition is true, and zero otherwise. The node loss is estimated over all i in $\{1, \ldots, \bar{n}\}$. Then, we set the edge loss using the cross entropy over the relations:

$$\mathcal{L}^E(\varepsilon_{j,k}, \bar{E}_{j,k}^A) := -\sum_{h \in \mathcal{R}} \left(\left[\bar{E}_{j,k}^A \doteq h \right] \cdot \log(\varepsilon_{j,k,h}) \right) \tag{7.14}$$

Note that we use the edges *after* the node re-ordering step. The final cost is computed as a linear combination of the node and edge loss:

$$\mathcal{L}(\bar{\mathcal{G}}, \bar{\mathcal{G}}^A) := \ell^V \cdot \sum_i \mathcal{L}^V(\boldsymbol{\nu}_i, \bar{\boldsymbol{v}}_i^A) + \ell^E \cdot \sum_j \sum_k \mathcal{L}^E(\varepsilon_{j,k}, \bar{E}_{j,k}^A) \tag{7.15}$$

where $\ell^V > 0$ and $\ell^E \geq 0$ are hyper-parameters weighting the node and edge loss.

Dataset	#Nodes	#Classes	#Edges	#Samples	Annot. Type	Data Type
Graph2Graph	5	2-20	2	250k	Fully	Graphs
COCO-Images	1-15	91	–	123k	–	Natural Img
COCO-Captions	1-15	91	–	617k	–	Captions
MNIST-Graphs	5	10	4	210k	Fully	Digits
CLEVR	3-10	3	4	90k	Fully	3D Shapes
VRD	1-10	100	70	5k	Sparse	Natural Img
Visual Genome	1-10	150	50	108k	Sparse	Natural Img

Table 7.2: Overview of the datasets we used in our evaluation. We compare by the number of nodes n, semantic node categories c, relation types r, and number of data samples. Moreover, we report the annotation and data types.

The index i takes values in $\{1, \dots, \bar{n}\}$, *i.e.*, we iterate over the nodes in the super-graphs. Therein, some nodes are encouraged to have a stronger activation either towards the `empty` class or towards a category in $\mathcal{C}$. In case of the edge indices, as we do not have any labels associated to the ingoing and outgoing edges of the `empty` vertices, we only consider the nodes that are not associated to the `empty` class in the ground truth. Thus, we only iterate over the edges connecting non-`empty` vertices, *i.e.*, j and k take values in $\{1, \dots, \bar{n}\}$, where $\bar{\mathbf{v}}_j^A \neq c+1$ and $\bar{\mathbf{v}}_k^A \neq c+1$. We update all the parameters in our network (denoted with Θ) using Adam [Kingma and Ba, 2015].

7.3 Datasets overview

We evaluate our approach for graph auto-encoding (Graph2Graph), visual node and graph generation (COCO-Images, VRD, Genome, MNIST-Graphs, CLEVR) as well as for inferring graphs from text (COCO-Captions). Table 7.2 shows an overview of these datasets, where we differentiate between the number of annotated graphs and the number of nodes and relation types. Note that in case of COCO and CLEVR we do not have any labels in the test set and, thus, we use the validation set for evaluation and report the number of graphs without the official test set. Since graph annotations are difficult to acquire, in VRD and Genome, we encounter missing edge annotations, which we denote with 'sparse' in Table 7.2. Finally, we include the type of the input data, which we use to construct the graph entities: visual input (*e.g.*, natural images), textual (captions), and random graphs.

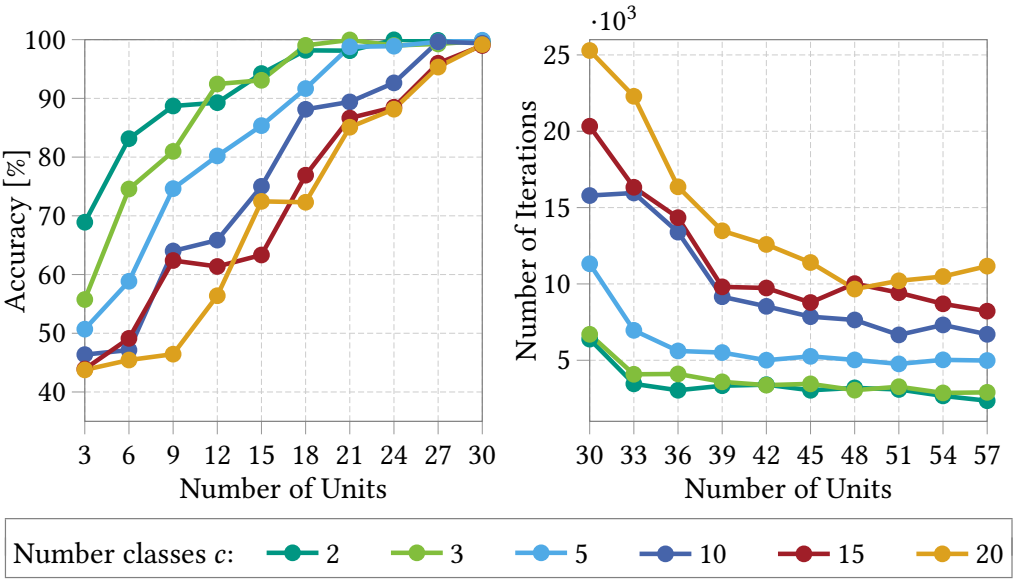


Figure 7.2: Analysis of NAP on the *Graph2Graph* dataset. Accuracy for small vector capacities (left) and number of iterations for a near perfect performance when varying the bottleneck sizes (right).

7.4 Experiments

In this section, we conduct several studies of our NAP learning scheme. First, we analyze two fundamental properties of NAP: its ability to recover graphs that are embedded into a vector (Section 7.4.1) and how the learned coordination of the neurons ν links the nodes to the classes (Section 7.4.2).

Then, we demonstrate NAP for inferring graphs from different input modalities (*e.g.*, text and images) and datasets (Section 7.4.6 for node generation and Section 7.4.7 for inferring complete semantic graphs). In this line of experiments, we evaluate additionally to our propose NAP scheme, additional baseline methods and establish an upper bound for our model which leverages additional supervision.

7.4.1 Vector capacity for graph auto-encoding

First, we evaluate how well NAP is able to extract graphs that are embedded into a vector by designing a graph auto-encoding experiment. To that end, we generate

	Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9
A single occurences of each class in the ground truth										
Node 0	0.0	100.0	1.4	0.0	60.7	3.7	36.0	0.1	3.5	6.9
Node 1	0.9	0.0	4.9	0.0	6.6	14.7	0.0	99.9	30.1	59.2
Node 2	17.0	0.0	61.2	0.0	1.9	24.9	64.0	0.0	20.8	3.4
Node 3	0.3	0.0	23.3	100.0	2.6	22.0	0.0	0.0	30.8	25.2
Node 4	81.7	0.0	9.2	0.0	28.2	34.6	0.0	0.0	14.9	5.3
Two occurences of each class in the ground truth										
Node 0	15.4	50.0	3.5	0.0	33.4	8.8	22.5	14.3	5.1	13.6
Node 1	6.2	0.0	8.2	5.5	13.5	14.2	4.6	50.0	20.5	31.4
Node 2	27.1	0.0	39.4	40.0	6.7	23.8	49.9	4.7	23.4	8.8
Node 3	3.1	0.0	33.0	50.0	5.3	24.5	2.9	18.9	31.3	31.0
Node 4	48.2	50.0	16.0	4.5	41.1	28.8	20.1	12.2	19.7	15.1

Table 7.3: Proportion [%] of the five nodes in V for predicting each of the ten digits

the *Graph2Graph* dataset of random graphs with an order of five, where each node is associated to one of c classes while the edges are binary. The vector embeddings γ are obtained using several layer of graph convolutions [Kipf and Welling, 2017] followed by average pooling over the node representations.

NAP is able to retrieve the original graph from the vector representation with near perfect performance if the embedding is large enough (Figure 7.2 left), while the impact of the embedding size is rather small (*e.g.*, an embedding with only 12 neurons is enough to achieve at least 50% accuracy over the nodes and edges for $c \leq 20$). Nonetheless, increasing the embedding size reduces the number of iterations during training that NAP requires to learn to extract the entire graph (Figure 7.2 right).

7.4.2 Activation analysis of the node-wise distributions

As we do not place any external restrictions on the node order, a key property of NAP is the learned coordination between the neurons. To evaluate this property, we take a closer look at the neurons ν, which learn to associate the vertices with the classes. We perform this analysis on our generated MNIST-Graphs dataset, where we placed MNIST digits [LeCun et al., 1998] randomly on a large blank image, producing five nodes ($n = 5$) per image with four relation types (*e.g.*, left).

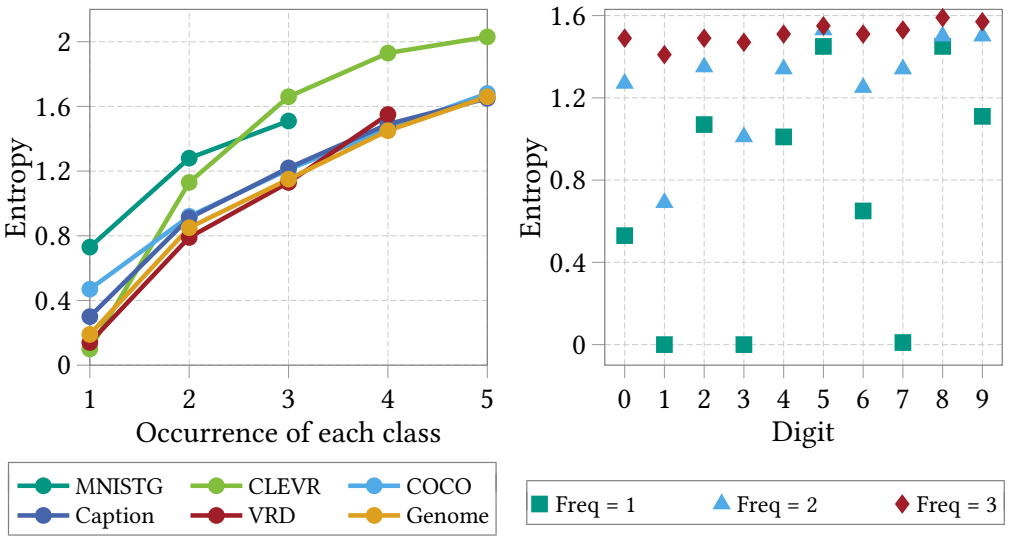


Figure 7.3: Entropy of the nodes averaged over each class in $\mathcal{C}$ for each of our datasets (left) and separate entropy for each of the classes in MNIST-Graphs (right).

Table 7.3 shows the activation proportion of the ν neurons for each of the ten classes. Therein, we constrain that the ground truth graphs $\mathcal{G}^*$ contain *a single instance* (top) or exactly *two instances* (bottom) of each targeted class. For the single instance case, we observe that the network learns to specialize different neurons for distinct classes, *e.g.*, the neuron with the ID 0 is the *only* one predicting the class 1. This property is facilitated by NAP, as it encourages the nodes that have a strong match for a specific class to activate even stronger into that direction. In a similar manner, when we have two instances in $\mathcal{G}^*$ for some digits (*e.g.*, class 1 and 3) we have mostly two neurons predicting them.

7.4.3 Entropy of the node-neurons

In the previous section, we visualize the assignments of each of the predicted nodes $\bar{\boldsymbol{\nu}}$ to the classes $\mathcal{C}$ on the MNIST-Graphs dataset. Thereby, when we analyzed a specific class $c \in \mathcal{C}$, we constrained that we only have a *single* instance in the ground truth for class c, *i.e.*, if $\sum_i \left[\boldsymbol{\nu}_i^* \doteq c\right]$ is equal to one (for $i \in \{1, \ldots, n^*\}$). As

we deducted from that example, many nodes were activating for specific classes, thus, the node-purity was high (*i.e.*, the entropy for each node was low). Next, we analyze this property across all our datasets.

In Figure 7.3 (left), we show the resulted average entropy over all nodes in the prediction for five different frequencies of the classes (*e.g.*, node entropies when we have exactly 3 occurring instances in $\boldsymbol{v}^*$). The purity of the nodes is decreasing by the number of instances in the scene. Finally, Figure 7.3 (right) visualizes the entropy for the different digits in MNIST-Graphs. Some of the nodes have specialized to a digit class (*e.g.*, digit one), while other have several nodes recognizing them (*e.g.*, digit 5).

7.4.4 Evaluation metrics

In the following, we evaluate our NAP scheme for node prediction and graph generation on a wide range of input data and annotation types. Due to the high divergence in data types and the difference in data representation of the node and edges, we leverage several kinds of evaluation metrics for comparing our models.

For measuring the node recognition performance, we use the Jaccard index and F1 score which are frequently used for comparing sets. In case of the edges, we cannot use the same metrics as for the nodes, as their correctness directly depends on the match between the nodes. For example, it does not suffice to predict that the `left` relation is present in an image, but one has also to specify the source and target nodes. Thus, we leverage first the edge accuracy *after* matching and implement several metrics over triplets (relation, source, and target) depending on the ground truth graph types. In case of fully-connected graphs (CLEVR and MNIST-Graphs), we consider the triplets as sets and use the same metrics as for the nodes (*i.e.*, Jaccard index and F1 score). For the sparse graphs, we use the same metrics as previous work [Xu et al., 2017], *i.e.*, the recall@k over the triplets.

Node recognition. Since the nodes of a graph can be seen as multisets comprising the object classes found in the scene, we employ evaluation metrics typically used for set comparison. To that end, we leverage the Jaccard index and F1 score between

the vertex vectors $\boldsymbol{v}$ and $\boldsymbol{v}^*$. Note that we use $\boldsymbol{v}$ after removing the `empty` vertices from the prediction $\bar{\boldsymbol{v}}$ (Section 7.2.2). We first define a function that counts the number of classes in the vectors $\boldsymbol{v}$ and $\boldsymbol{v}^*$, as follows:

$$\chi : \bigcup_{k \in \mathbb{N}} (\mathcal{C}^k \times \mathcal{C}) \rightarrow \mathbb{N}, \quad \chi(\boldsymbol{v}, h) \coloneqq \sum_{i \in \{1, \ldots, \dim(\boldsymbol{v})\}} [V_i \doteq h] \tag{7.16}$$

where $\dim(\boldsymbol{v})$ is the number of entries in the vector $\boldsymbol{v}$, while with $|\cdot|$ we marks the size of a set (*i.e.*, the number of its elements).
We then define the intersection between these sets by counting the number of objects of each class h in both $\boldsymbol{v}$ and $\boldsymbol{v}^*$:

$$\Omega : \bigcup_{k \in \mathbb{N}, k^* \in \mathbb{N}} (\mathcal{C}^k \times \mathcal{C}^{k^*}) \rightarrow \mathbb{N}_0, \quad \Omega(\boldsymbol{v}, \boldsymbol{v}^*) \coloneqq \sum_{h \in \mathcal{C}} \min(\chi(\boldsymbol{v}, h), \chi(\boldsymbol{v}^*, h)) \tag{7.17}$$

where the function min selects the minimum value of the two inputs.
In a similar manner, we estimate the union between the objects in the nodes:

$$\Upsilon : \bigcup_{k \in \mathbb{N}, k^* \in \mathbb{N}} (\mathcal{C}^k \times \mathcal{C}^{k^*}) \rightarrow \mathbb{N}, \quad \Upsilon(\boldsymbol{v}, \boldsymbol{v}^*) \coloneqq \sum_{h \in \mathcal{C}} \max(\chi(\boldsymbol{v}, h), \chi(\boldsymbol{v}^*, h)) \tag{7.18}$$

where max selects the maximum of $\chi(\boldsymbol{v}, h)$ and $\chi(\boldsymbol{v}^*, h)$.
Note that we only consider non-`empty` ground truth graphs in our evaluation and define the Jaccard index as follows:

$$\text{Jacc}^V : \bigcup_{k \in \mathbb{N}, k^* \in \mathbb{N}} (\mathcal{C}^k \times \mathcal{C}^{k^*}) \rightarrow [0, 1], \quad \text{Jacc}^V(\boldsymbol{v}, \boldsymbol{v}^*) \coloneqq \frac{\Omega(\boldsymbol{v}, \boldsymbol{v}^*)}{\Upsilon(\boldsymbol{v}, \boldsymbol{v}^*)} \tag{7.19}$$

As a second metric, we compare the node sets based on their F1 score:

$$\text{F1}^V : \bigcup_{k \in \mathbb{N}, k^* \in \mathbb{N}} (\mathcal{C}^k \times \mathcal{C}^{k^*}) \rightarrow [0, 1], \quad \text{F1}^V(\boldsymbol{v}, \boldsymbol{v}^*) \coloneqq 2 \cdot \frac{\Omega(\boldsymbol{v}, \boldsymbol{v}^*)}{\Omega(\boldsymbol{v}, \boldsymbol{v}^*) + \Upsilon(\boldsymbol{v}, \boldsymbol{v}^*)} \tag{7.20}$$

Edges. Since the edges are directly connected to the nodes (as they depict their relation), we compute the edge performance by leveraging the accuracy after matching the ground truth and predicted nodes in the supergraphs. Let $\mathcal{I}$ be a set of indizes that point to all non-`empty` vertices in $\bar{\boldsymbol{v}}^A$. We then use the re-ordered ground truth edges $\bar{E}^A$ and predictions $\bar{E}$ and compute their accuracy:

$$\text{Acc}^E\colon \bigcup_{k\in\mathbb{N},\,k^*\in\mathbb{N}} (\mathcal{R}^{k\times k}\times\mathcal{R}^{k^*\times k^*}) \to [0,1],\ \text{Acc}^E(\bar{E},\bar{E}^A) \coloneqq \frac{1}{|\mathcal{I}|^2}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}\left[\bar{E}_{i,j} \doteq \bar{E}^A_{i,j}\right] \tag{7.21}$$

While the accuracy over the edges gives some insight into the strengths of the models, it is dependent on our matching scheme. Thus, we define next the evaluation metrics that consider *sets of triplets*.

Triplets. The properties of the edge annotations have a strong deviation across datasets: (1) variations in the number of relationship types: we have up to 70 different relation types, (2) sparseness of the graphs, and (3) completeness and amount of noise of the edge annotations (*i.e.*, the manually annotated datasets contain incorrect labels). Thus, we leverage different metrics depending on these properties.

First, for MNIST-Graphs and CLEVR, we consider the triplets as sets and compare them using the Jaccard index and F1 score as for the nodes. To that end, we define a counting function for triplets:

$$\chi^{\text{graph}}(\mathcal{G},s,t,e) \coloneqq \sum_i\sum_j([\boldsymbol{v}_i \doteq s]\cdot\left[\boldsymbol{v}_j \doteq t\right]\cdot\left[E_{i,j} \doteq e\right]) \tag{7.22}$$

where i and j are iterating over $i,j \in \{1,\dots,\dim(\boldsymbol{v})\}$, $e \in \mathcal{R}$ and $s,t \in \mathcal{C}$. For ease of notation, we leave out in the following the function definitions as they are identical to their node counterpart (except that we input triplets instead of nodes). We then define the intersection and union as follows:

$$\Omega^{\text{graph}}(\mathcal{G},\mathcal{G}^*) \coloneqq \sum_{s\in\mathcal{C}}\sum_{t\in\mathcal{C}}\sum_{e\in\mathcal{R}}\min(\chi^{\text{graph}}(\mathcal{G},s,t,e),\chi^{\text{graph}}(\mathcal{G}^*,s,t,e)) \tag{7.23}$$

$$\Upsilon^{\text{graph}}(\mathcal{G}, \mathcal{G}^*) \coloneqq \sum_{s \in \mathcal{C}} \sum_{t \in \mathcal{C}} \sum_{e \in \mathcal{R}} \max(\chi^{\text{graph}}(\mathcal{G}, s, t, e), \chi^{\text{graph}}(\mathcal{G}^*, s, t, e)) \tag{7.24}$$

Then, we estimate the Jaccard index over the triplets as follows:

$$\text{Jacc}^{\text{triplet}}(\mathcal{G}, \mathcal{G}^*) \coloneqq \frac{\Omega^{\text{graph}}(\mathcal{G}, \mathcal{G}^*)}{\Upsilon^{\text{graph}}(\mathcal{G}, \mathcal{G}^*)} \tag{7.25}$$

Thus, the Jaccard index is simply the intersection of the sets over their union.
In a similar manner, we compute the F1 score:

$$\text{F1}^{\text{triplet}}(\mathcal{G}, \mathcal{G}^*) \coloneqq 2 \cdot \frac{\Omega^{\text{graph}}(\mathcal{G}, \mathcal{G}^*)}{\Upsilon^{\text{graph}}(\mathcal{G}, \mathcal{G}^*) + \Omega^{\text{graph}}(\mathcal{G}, \mathcal{G}^*)} \tag{7.26}$$

In case of the sparsely labeled datasets (VRD and Genome), we leverage the recall@k as in related work. Note that since we do not predict any bounding boxes we do not use the overlap over bounding boxes in our formulation of the recall@k. Let ϑ be the triplet confidence defined as:

$$\vartheta : \mathcal{I} \times \mathcal{I} \rightarrow [0, 1], \quad \vartheta(i, j) \coloneqq \boldsymbol{\nu}_{i,s} \cdot \boldsymbol{\nu}_{j,t} \cdot \varepsilon_{i,j,e} \tag{7.27}$$

where $s = \arg\max_{h \in \mathcal{C}}(\boldsymbol{\nu}_{i,h})$, $t = \arg\max_{h \in \mathcal{C}}(\boldsymbol{\nu}_{j,h})$, and $e = \arg\max_{h \in \mathcal{R}}(\varepsilon_{i,j,h})$.

Based on the triplet confidence, we build 3 matrices carving p triplets with the source and target nodes $S, T \in \mathcal{C}^p$ and relations $R \in \mathcal{R}^p$ as: $S_i = \boldsymbol{\nu}_{\iota_s(i)}, T_i = \boldsymbol{\nu}_{\iota_t(i)}, R_i = E_{\iota_s(i),\iota_s(i)}, \forall i \in \{1, \ldots, p\}$. We define $\iota_s, \iota_t : \{1, \ldots, p\} \rightarrow \{1, \ldots, p\}$ as bijective functions having the following property:

$$\vartheta(\iota_s(i), \iota_t(i)) \geq \vartheta(\iota_s(j), \iota_t(j)), \forall i, j \in \{1, \ldots, p\}, i \leq j \tag{7.28}$$

Thus, we re-order the triplets in descending order based on the triplet confidence given by ϑ. We then select the top k triplets by directly leveraging the previously defined ordering of the edges as well as the source and target nodes:

$$S_i^k \coloneqq \boldsymbol{\nu}_{\iota_s(i)}, T_i^k \coloneqq \boldsymbol{\nu}_{\iota_t(i)}, P_i^k \coloneqq E_{\iota_s(i),\iota_s(i)}, \forall i \in \{1, \ldots, \min(k, p)\} \tag{7.29}$$

Then, we compute a counting function for our previously defined matrices:

$$\chi^{\text{triplet}}(S, T, R, s, t, e) \coloneqq \sum_i ([S_i \doteq s] \cdot [T_i \doteq t] \cdot [R_i \doteq e]) \tag{7.30}$$

where i is iterating over all triplets defined by the matrices S, T, and R depicting the source, target, and relation. As for the predicted triplets, we build p^* ground truth triplet representations using the matrices $S^*, T^* \in \mathcal{C}^{p^*}$ and $R^* \in \mathcal{R}^{p^*}$.
For estimating the recall@k, we leverage both ground truth and predicted triplets via the three matrices shaping the triplets (S, T, and R):

$$\text{recall@}\, k \coloneqq \frac{1}{p^*} \sum_i \min(\chi^{\text{graph}}((\boldsymbol{v}^*, E^*), S_i^*, T_i^*, R_i^*), \chi^{\text{triplet}}(S^k, T^k, R^k, S_i^*, T_i^*, R_i^*)) \tag{7.31}$$

where i iterates over $i \in \varphi(S^*, T^*, R^*)$. Therein, we define φ as a function that maps the three triplet matrices to the indices that point to a *single* occurrence of each triplet. More formally, we design φ such that $\forall i, j \in \varphi(S^*, T^*, R^*)$ the condition holds that the function only points on unique triplets, *i.e.*, $[(S_i^* \doteq S_j^*, T_i^* \doteq T_j^*, R_i^* \doteq R_j^*) \Rightarrow i \doteq j]$. Thus, $\varphi(S^*, T^*, R^*) \subseteq \{1, \ldots, p^*\}$ and is equal to $\{1, \ldots, p^*\}$ if and only if all triplets are *unique*. Note that we further report all results in percentages.

7.4.5 Evaluated methods

Additionally, to our proposed NAP model, we provide the results of two baseline methods (*static graph* and *multi-class network*). Moreover, we establish an upper bound for our approach and evaluate methods that require additional location supervision. We provide further implementation details of the baselines in Appendix E.2.

Static graph. First, we employ a *static graph* baseline that always infers the *same graph* comprising the most frequent nodes and edges. For example, the most frequent node is the `person` category in Visual Genome, while the most common predicate (except for he `no-relation` class) is the `on` relation.

Annotation: {toothbrush, toothbrush, person, sink, person}
Caption 1: "Two children stand at the sink and brush their teeth"
NAP: {toothbrush, toothbrush, sink, person, person}
Caption 2: "Children standing in the bathroom brushing their teeth"
NAP: {toothbrush, toothbrush, toothbrush, person, person, sink}
Image
Multi-cls: {toothbrush, person, sink, toothbrush, person}
MRCNN: {person, person, table, sink, toothbrush, toothbrush}
NAP: {toothbrush, toothbrush, person, person, sink}

Figure 7.4: Example node estimates for visual and textual input data. NAP generated the correct number of nodes for all classes in caption 1. In comparison, caption 2 does not contain any number specifications of the objects, leading to an incorrect estimate of NAP for the toothbrush object. For image-only, NAP recognized the correct number of people and toothbrushes but missed the sink.

Multi-class network. We further employ a *multi-class* network, which uses the same backbone architecture as NAP, but is trained in a multi-class setup (*i.e.*, leveraging the sigmoid activation). More precisely, the network predicts at most one object instance (*i.e.*, one node) per semantic class and does not predict any edges. We show an example of a prediction of the *multi-class* network in Figure 7.4 where the model recognized two categories: `tootbrush` and `person`.

Networks with additional supervision. When further supplementary annotations are available in the dataset (*i.e.*, bounding boxes and instance masks), we implement methods that leverage this additional information, therefore, defining an upper bound for our NAP scheme.

7.4.6 Node estimation in textual and visual data

Setting. NAP can also be used for node estimation only, *e.g.*, if the graph representation contains the nodes but not their relationships (*i.e.*, the number of objects present in an image and their respective class). In such cases, the scoring function ν associates individual vertices to their respective class, while dismissing the predicted edges. We evaluate NAP for this task in context of multi-set object prediction from images and textual descriptions obtained from the COCO [Lin et al., 2014]

Method	Mask-Super.	COCO Jacc.	COCO F1	[#caps=1] Jacc.	[#caps=1] F1	[#caps=5] Jacc.	[#caps=5] F1	COCO+Cap Jacc.	COCO+Cap F1
MRCNN	YES	81.23	86.98	–	–	–	–	–	–
Static Graph	NO	15.41	21.98	15.41	21.98	15.41	21.98	15.41	21.98
Multi-class	NO	35.04	43.55	36.77	44.44	43.49	51.94	43.91	52.74
Ours (NAP)	NO	47.72	57.11	47.72	56.14	54.38	63.08	57.95	67.05

Table 7.4: Results on the COCO dataset for different input modality types: *images* (left), *captions* (center), and *both* modalities (right).

dataset. In case of visual data, we embed the images into the global representation vector γ using the penultimate layer of a ResNet50 [He et al., 2016]. For the textual descriptions, we compute γ by stacking multiple 1D-convolution layers from the one-hot representations of the words. As the captions and images in COCO cover the same examples (but different modalities), we also generate a combined descriptor by combining both visual and textual descriptors via concatenation.

Results. In Table 7.4, we report the node recognition results using the Jaccard index and F1 score between the ground truth $\boldsymbol{v}^*$ and inferred node classes $\boldsymbol{v}$ as our metrics. To derive an upper bound for our approach, we use the instance-segmentation network Mask-RCNN [He et al., 2017] and generate a node for each predicted object instance. Mask-RCNN reaches a Jaccard index of around 85.05%, but requires pixel-wise segmentation masks as labels during training. NAP obtains a better performance than the multi-class network that also does not require location supervision for any input modalities (example predictions in Figure 7.4 of our caption and image only networks). Since the COCO dataset contains multiple captions per graph, we analyze the impact on the results when multiple image descriptions are passed to NAP as input. Using all five captions improves the performance considerably (54.38%), while when leveraging both the five captions and the image the Jaccard index rises to 57.95%.

Method	Box-Superv	Nodes Jacc.	Edges Acc.	Triplets R100	Triplets R150	Nodes Jacc.	Edges Acc.	Triplets R100	Triplets R150
[Lu et al., 2016a]	YES	42.35	33.10	20.89	20.89	–	–	–	–
Static Graph	NO	10.61	12.56	0.00	0.00	2.53	3.50	0.00	0.00
Multi-class	NO	23.39	–	–	–	10.86	–	–	–
Ours (NAP)	NO	25.17	56.38	6.15	8.22	30.83	44.90	8.66	11.33

Table 7.5: Semantic graph generation on *VRD* (left) and *Visual Genome* (right)

Method	Box-Superv	Nodes Jacc.	Edges Acc.	Triplets Jacc.	Triplets F1	Nodes Jacc.	Edges Acc.	Triplets Jacc.	Triplets F1
SSD	YES	99.53	99.99	99.19	99.43	99.81	99.02	98.61	99.24
Static Graph	NO	5.53	25.42	0.13	0.24	21.41	24.95	1.39	2.64
Multi-class	NO	71.18	–	–	–	31.35	–	–	–
Ours (NAP)	NO	99.23	99.22	97.56	98.45	99.59	91.63	91.20	95.05

Table 7.6: Semantic graph generation on *MNIST-Graphs* (left) and *CLEVR* (right)

7.4.7 Learning graphs from natural and synthetic images

Setup. We now evaluate our method for complete graph generation using both natural and synthetic visual data. While the VRD [Lu et al., 2016a] and Visual Genome [Krishna et al., 2016] datasets comprise natural images with manually labeled graphs of object instances (*e.g.*, `cat`) and predicates (*e.g.*, `sleeping-on`), MNIST-Graphs and CLEVR are synthetically generated datasets with location-based edge labels (*e.g.*, `left-behind`). While we generate MNIST-Graphs ourselves by placing digits from MNIST randomly on a blank 128×128 image, CLEVR [Johnson et al., 2017a] is a popular dataset comprising up to ten nodes (3D geometric shapes) per graph. As mentioned previously, to evaluate the quality of node generation, we adopt the Jaccard index. For the edges, we leverage the accuracy between $\bar{E}$ and $\bar{E}^A$ after matching (as the edges cannot be shaped into sets as in case of the nodes). In the completely labeled datasets (CLEVR and MNIST-Graphs), we also report the Jaccard index and F1 score over the triplets (relation, source and target node), while for the sparsely annotated datasets (VRD and Visual Genome), we use recall@k as in [Lu et al., 2016a]. However, in contrast to other works, our definition of the recall does not take into consideration location information.

Dataset	Graph2Graph	C-Img	C-Cap	MNIST	CLEVR	Genome	VRD
Inference [ms]	0.58	24.54	0.03	0.46	2.50	24.66	24.92
Matching [ms]	1.95	1.42	1.61	1.54	7.28	5.17	7.66

Table 7.7: Average inference and graph matching time of our proposed method.

Results. In Table 7.5, we show the results of our method for the manually annotated datasets VRD and Visual Genome. Thereby, we use V+L+K [Lu et al., 2016a] as an upper bound, which requires both bounding box annotations and semantic embeddings for training. Without using such additional labels, NAP clearly surpasses the multi-class network, with largest gains on the Visual Genome dataset (with a node Jaccard index of 30.83% and edge accuracy of 44.90%). Next, we evaluate our model on synthetic datasets (MNIST-Graphs and CLEVR) in Table 7.6, where in contrast to the previous benchmarks, the graphs are *balanced* and *completely labeled*, *i.e.*, there are no missing or noisy graph annotations. We create an upper bound for our approach using the SSD [Liu et al., 2016] detector (in this case there are no labeled masks as in COCO). Since the relations are location-based, the edges are computed directly from the bounding box coordinates as we did in Chapter 4 for our architecture generating soft paths. While SSD achieves a near perfect performance on the synthetic datasets, NAP achieves a triplet Jaccard index of 97.56% on MNIST-Graphs and 91.2% on CLEVR.

7.5 Learning and inference speed

While there are many ways of addressing our node assignment problem, we leverage a matching algorithm that runs in $O(\bar{n}^3)$. Computing the node assignment cost κ^V is constant for each pair (i, i^*). Thus, this results in an overall time complexity of $O(\bar{n}^2)$. In case of the edges, for computing the assignment cost κ^E, we require $O(\bar{n}^2)$ for each node pair (i, i^*) in the supergraphs.

Our NAP scheme requires two modifications to the conventional image classification training scheme: (1) two additional fully-connected layers that generate

the node and edge set and (2) a matching step between the predicted and ground truth nodes. Next, we analyze the extra time (in seconds) our network takes on different datasets and settings (Table 7.7). For the inference step, our network generates the node and edge set in around 1ms for Graph2Graph, COCO-Captions, MNIST-Graphs, and CLEVR. In case of COCO-Images, Genome and VRD we have an increased inference time, which is caused mostly by generating the feature vectors themselves, *i.e.*, passing the image through a pre-trained CNN (around 23ms). The overhead of the matching scheme is sustainable as it needs on average at most 8ms per graph instance, which is around one third of the time of the 2D-CNNs.

7.6 Summary and discussion

While previously we focused on reasoning on given graph representations of visual and textual data, in this chapter, we addressed the problem of constructing such semantic graphs automatically. To that end, we presented a method for inferring semantic graphs from input data sources that can be encoded in vector form. The core of our approach is the Node Association Procedure (NAP), a novel learning scheme that leverages a matching step between predicted and ground truth vertices. NAP does not assume a given order of the nodes, does not require location supervision, and allows us to train our model end-to-end. We demonstrate how our approach can be applied for graph auto-encoding, node recognition in images and captions, and visual graph generation on synthetic and real-world data. As NAP does not require any additional supervision besides graphs, our method has the potential to facilitate progress in high-level reasoning by detaching structured representations from costly task-specific processing steps.

8 Conclusion

While AI has made large strides in semantic analysis of natural images (*e.g.*, image captioning, visual reasoning), high-level understanding of documents has been scarce. In this thesis, we address the problem of question answering on learning materials where we target diverse types of graphical content. Our emphasis is on textbook content and presentation slides that have a strong variety of page structure and are rich in variations of figure types. To that end, we address the *question answering task on three different levels of page granularity*:

- We first address the visual reasoning task and introduce a novel architecture for answering compositional questions based on different types of images. Our model achieves state-of-the-art results on several popular benchmarks and one can infer the reasoning of the model for answering different questions.
- We propose a novel network for multi-modal QA based on edge refinement, which achieves strong results on textbook question answering.
- We dive into the page analysis task and collect a novel large scale dataset of 4M questions associated to 400k pages.

Moreover, in order to employ graph neural networks on graphical and textual content extracted from pages, we need to both *pre-process the page* and *bring the extracted information into graph format*:

- For extracting relevant document components from pages, we use multi-class page segmentation approaches which we performed on our own collected datasets.
- Since VQA methods based on graph networks require a graph representation of the input image, we propose a novel technique for generating semantic graphs directly from feature vectors (without requiring bounding box supervision).

8.1 Contributions

Next, we summarize the key contributions made in this thesis:

Page analysis on presentation slides
Related publications: [Haurilet et al., 2019b,c]
We introduce the task of semantic analysis of presentation slides. To the best of our knowledge, this problem was not addressed before and we therefore collect and publicly release two large scale datasets with accompanying presentation slides. To that end, we manually annotate 2000 and 1330 pages, respectively, with pixel-level annotations of 25 semantic classes. While previous page segmentation datasets constrain that a *single* class is assigned to each pixel, we allow the mapping of several categories in a multi-class setup. We provide a detailed analysis of our collected data where we show the class distribution of our labels, the class overlap per pixel showing the importance of the multi-class property of our datasets, and the location dependence of different document classes. Finally, we establish a lower bound by implementing several baselines on our data and adapt complex deep learning models previously used for semantic segmentation of natural images to our tasks.

Answering questions by following paths
Related publications: [Haurilet et al., 2019d]
We introduce a novel architecture for answering questions from graphical content. As visual data can be highly compositional (*i.e.*, several objects interacting with each other), we leverage a *graph representation* of the scene. Our model traverses this graph structure in search of information relevant for answering the current

question. While previous traversal schemes are either *non-differentiable* or can only focus on *single nodes* at once, our approach is trained with regular back-propagation techniques and can attend to several nodes simultaneously. Thus, our model can easily infer answers for counting and exist-type questions, where it is necessary to focus on several nodes at once. Due to the traversal characteristic of our method, we can directly infer the *reasoning* of our model for answering different questions. Finally, our model achieves state-of-the-art results on several datasets for visual reasoning improving the performance of other complex VQA methods.

Multi-modal question answering on textbook content

Related publications: [Bender*, Haurilet* et al., 2019; Haurilet et al., 2018]

We address the problem of answering questions on multi-modal data (text and images). To that end, we introduce an architecture for automatically extracting text from visual representations of pages which constitutes a pre-processing step for our multi-modal QA technique. We evaluate our text generation method on the large scale IM2LATEX dataset comprising 100k images of typed text content.

We move forward with the multi-modal question answering task where we focus on textbook content extracted from the sixth grade curricula. Due to the vast amount of input content associated to each query, we propose a pre-processing approach for selecting *supporting* entities that are relevant for generating an answer. We propose a neural architecture based on edge refinement that operates on node-pairs (sentences and scientific terms) for capturing the compositional input content. Finally, we evaluate our approach for textbook question answering, where we obtain compatible results with other VQA methods on the validation set of the TQA benchmark and the first place in the TQA challenge encompassing the test set of TQA.

Answering questions on entire pages

Related publications: [Haurilet et al., 2021b]

We introduce the novel task of answering questions on *entire* pages. To that end, we collect SlideQA – a large scale dataset of 4 million page-related queries and 400k pages. While we include different types of document components to our generated

pages (*e.g.*, natural images, tables, enumerations), we also provide a high diversity of questions which focus on different parts of the page. We perform a thorough analysis of our collected data where we show the unique properties of our questions and slides, *e.g.*, compositionality of the pages, diversity of document components, variety in query tasks, and a high number of unique answers.

On our dataset, we evaluate our novel architecture specialized on answering questions on page content. Our network fuses the input image and question into the same representation space and, then, via several attention modules we infer an answer at *character-level simultaneously*. The proposed attention-based method improves the performance of other complex VQA approaches by a significant margin.

Graph generation from single feature vectors

Related publications: [Haurilet et al., 2021a]

We introduce a novel approach for generating graph structures from global data representations in form of vectors. Our neural architecture generates the nodes and edges *directly* from the feature vector leveraging conventional fully-connected layers. Due to the unordered nature of the node set, we propose a matching scheme between the predicted and ground truth nodes, enabling end-to-end training of our network. In contrast to other graph generation techniques, our method does not require bounding box supervision and can operate on both images and text. We show the strength of our network on several datasets for node recognition from images (COCO-Images) and text (COCO-Captions) as well as graph generation from synthetic (MNIST-Graphs and CLEVR) and natural images (VRD and Genome).

Collected datasets and benchmarks

Throughout this thesis, we introduce multiple tasks for document understanding which were not addressed before and, thus, we collect several accompanying datasets. Most prominently, the SlideQA dataset encompasses 4M questions associated to 400k pages. While the queries are related to the structure and semantic information related to different entities in documents, we ensure that the pages comprise a high variety of graphical and structural types.

Moreover, we collect two datasets for page segmentation encompassing in total over 3000 presentation slides with fine-grained labels of 25 classes. While previous methods include a single label per pixel, we enable multi-label classification where regions can belong to several semantic categories (*e.g.*, text and diagrams). To foster further research on slide segmentation, we made the datasets publicly available.

Finally, we propose a novel learning scheme for training graph generation networks. To show its strength on a wide variety of datasets, we analyze our approach on two novel datasets in addition to existing benchmarks. Namely, we introduce the MNIST-Graphs dataset comprising over 210k images of digits with location-based relations and Graph2Graph encompassing synthetically generated nodes and edges which we use for graph autoencoding.

8.2 Future work

While models for high-level understanding and especially for question answering have made extraordinary strides in recent years, there is still a wide gap to human performance. Due to the nature of our proposed approaches, we can directly infer the reasoning of our models and, thus, we can easily deduct weak spots in the data and overall architecture. Thus, a possible future work is leveraging this feature for improving and balancing the data and re-structuring its under-performing modules.

A reason for such a strong improvement of models for high-level reasoning tasks lies in the strong development of visual and textual embeddings. Thus, a possible extension of our work is to employ different novel embeddings in our graph and visual data structure (*e.g.*, features extracted from transformers). Finally, our question answering methods can be utilized in a wide variety of applications, such as: (1) for a more convenient interaction with a mobile assistant/robot, (2) in assistive technologies conveying relevant information from learning materials to people with visual impairments, and (3) automatic indexing of document content based on information carved by the input query.

Short Curriculum Vitae

Name	Monica Laura Zündorf

Education and Professional Experience

since 01/2016	PhD Student and Research Assistant Computer Vision for Human-Computer Interaction, KIT
09/2015	M.Sc. in Computer Science at KIT Graduation with distinction Thesis: "*Completely Unsupervised Person Identification in TV-Series*" at the Computer Vision for HCI Lab

Scholarships and Awards

06/2020	Outstanding reviewer award (CVPR)
10/2009 – 09/2014	Full-Scholarship from DAAD

Own Publications

1. **Uncertainty-sensitive Activity Recognition: A Reliability Benchmark, and the CARING Models**
   Alina Roitberg, Monica Haurilet, Manuel Martinez, and Rainer Stiefelhagen
   International Conference on Pattern Recognition (ICPR), 2020.

2. **Detective: An Attentive Recurrent Model for Sparse Object Detection**
   Amine Kechaou, Manuel Martinez, Monica Haurilet, and Rainer Stiefelhagen
   International Conference on Pattern Recognition (ICPR), 2020.

3. **Multi-task Calorie Prediction on a Large-Scale Recipe Dataset Enriched with Nutritional Information**
   Robin Ruede, Verena Heusser, Lukas Frank, Alina Roitberg, Monica Haurilet, and Rainer Stiefelhagen
   International Conference on Pattern Recognition (ICPR), 2020.

4. **Bring the Environment to Life: Sonifying Fine-Grained Localized Objects for Persons with Visual Impairments**
   Angela Constantinescu, Karin Mueller, Monica Haurilet,Vanessa Petrausch, and Rainer Stiefelhagen
   International Conference on Multimodal Interaction (**ICMI**), 2020.

5. **CNN-based Driver Activity Understanding: Shedding Light on Deep Spatiotemporal Representations**
   Alina Roitberg, Monica Haurilet, Simon Reiss, and Rainer Stiefelhagen
   International Conference on Intelligent Transportation Systems (**ITSC**), 2020.

6. **Open Set Driver Activity Recognition**
   Alina Roitberg, Chaoxiang Ma, Monica Haurilet, and Rainer Stiefelhagen
   Intelligent Vehicles (**IV**), 2020.

7. **Deep Classification-driven Domain Adaptation for Cross-Modal Behavior Recognition**
   Simon Reiss, Alina Roitberg*, Monica Haurilet, and Rainer Stiefelhagen*
   Intelligent Vehicles (**IV**), 2020.

8. **Activity-aware Attributes for Zero-Shot Driver Behavior Recognition**
   Simon Reiss, Alina Roitberg*, Monica Haurilet, and Rainer Stiefelhagen*
   CVPR Workshop on Learning with Limited Labels (**VL3**), 2020.

9. **Drive&Act: A Multi-modal Dataset for Fine-grained Driver Behavior Recognition in Autonomous Vehicles**
   Manuel Martin, Alina Roitberg*, Monica Haurilet, Matthias Horne, Simon Reiss, Michael Voit, and Rainer Stiefelhagen*
   In International Conference on Computer Vision (**ICCV**), 2019.

10. **It's not about the Journey; It's about the Destination: Following Soft Paths under Question-Guidance for Visual Reasoning**
    Monica Haurilet, Alina Roitberg, and Rainer Stiefelhagen
    In Conference on Computer Vision and Pattern Recognition (**CVPR**), 2019.

11. **WiSe – Slide Segmentation in the Wild**
    Monica Haurilet, Alina Roitberg, Manuel Martinez, and Rainer Stiefelhagen
    In International Conf. on Document Analysis and Recognition (**ICDAR**), 2019.

12. **Learning Fine-Grained Image Representations for Mathematical Expression Recognition**
    Sidney Bender, Monica Haurilet*, Alina Roitberg, and Rainer Stiefelhagen*
    In ICDARW on Graphics Recognition (**GREC**), 2019.

13. **End-to-end Prediction of Driver Intention using 3D Convolutional Neural Networks**
    Patrick Gebert, Alina Roitberg*, Monica Haurilet, and Rainer Stiefelhagen*
    In Intelligent Vehicles Symposium (**IV**), 2019.

14. **DynGraph – A Dynamic Graph Architecture for VQA**
    Monica Haurilet, Ziad Al-Halah, and Rainer Stiefelhagen
    In German Conference on Pattern Recognition (**GCPR**), 2019.

15. **Analysis of Deep Fusion Strategies for Multi-modal Gesture Recog.**
    Alina Roitberg, Tim Pollert*, M. Haurilet, Manuel Martin, and Rainer Stiefelhagen*
    In CVPR Workshop on Analysis, and Modeling of Faces and Gestures (**AMFG**), 2019.

16. **SPaSe – Multi-Label Page Segmentation for Presentation Slides**
    Monica Haurilet, Ziad Al-Halah, and Rainer Stiefelhagen
    In Winter Conference on Applications of Computer Vision (**WACV**), 2019.

17. **MoQA - A Multi-Modal Question Answering Architecture**
Monica Haurilet, Ziad Al-Halah, and Rainer Stiefelhagen
In ECCV Workshop on Shortcomings in Vision and Language (SiVL), 2018.
🏆 Winner of the Textbook Question Answering Challenge (TQA) held in conjunction with CVPR, 2017.

18. **Towards a Fair Evaluation of Zero-Shot Action Recognition using External Data**
Alina Roitberg, Manuel Martinez, Monica Haurilet, and Rainer Stiefelhagen
In ECCV Workshop on Shortcomings in Vision and Language (SiVL), 2018.

19. **DriveAHead – A Large-Scale Driver Head Pose Dataset**
Anke Schwarz, Monica Haurilet*, Manuel Martinez, and Rainer Stiefelhagen*
In CVPR Workshop on Computer Vision in Vehicle Technology (CVVT), 2017.

20. **Marlin: A High Throughput Variable-to-Fixed Codec using Plurally Parsable Dictionaries**
Manuel Martinez, Monica Haurilet, R. Stiefelhagen, and J. Serra-Sagrista
In Data Compression Conference (DCC), 2017.

21. **Naming TV Characters by Watching, and Analyzing Dialogs**
Monica Haurilet, Makarand Tapaswi, Ziad Al-Halah, and Rainer Stiefelhagen
In Winter Conference on Applications of Computer Vision (WACV), 2016.

Bibliography

S. Abdou and M. S. Scordilis. Beam search pruning in speech recognition using a posterior probability-based confidence measure. In *Speech Communication*, 2004.

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. In *Transactions on Pattern Analysis and Machine Intelligence*, 2012.

M. Acharya, K. Kafle, and C. Kanan. Tallyqa: Answering complex counting questions. In *Association for the Advancement of Artificial Intelligence*, 2019.

A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. In *International Journal of Computer Vision*, 2017.

A. Amin and R. Shiu. Page segmentation and classification utilizing bottom-up approach. In *International Journal of Image and Graphics*, 2001.

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Deep compositional question answering with neural module networks. In *Conference on Computer Vision and Pattern Recognition*, 2016.

A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. Inter-

national conference on document analysis and recognition2015 competition on recognition of documents with complex layouts-rdcl2015. In *International Conference on Document Analysis and Recognition*, 2015.

A. Araujo, J. Chaves, H. Lakshman, R. Angst, and B. Girod. Large-scale query-by-image video retrieval using bloom filters. In *arXiv preprint arXiv:1604.07939*, 2015.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. In *arXiv preprint arXiv:1803.01271*, 2018.

P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems*, 2016.

H. Ben-Younes, R. Cadene, M. Cord, and N. Thome. Mutan: Multimodal tucker fusion for visual question answering. In *International Conference on Computer Vision*, 2017.

Bender*, Haurilet*, A. Roitberg, and R. Stiefelhagen. Learning fine-grained image representations for mathematical expression recognition. In *International Conference on Document Analysis and Recognition Workshops*, 2019.

A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *International Conference on Image Processing*, 2016.

G. Bouchard, S. Singh, and T. Trouillon. On approximate reasoning capabilities of low-rank vector spaces. In *Association for the Advancement of Artificial Intelligence Spring Syposium*, 2015.

T. M. Breuel. Robust, simple page segmentation using hybrid convolutional mdlstm networks. In *International Conference on Document Analysis and Recognition*, 2017.

H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In *Conference on Computer Vision and Pattern Recognition*, 2018.

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Conference on Computer Vision and Pattern Recognition*, 2017.

K. Chen, M. Seuret, M. Liwicki, J. Hennebert, and R. Ingold. Page segmentation of historical document images with convolutional autoencoders. In *International Conference on Document Analysis and Recognition*, 2015a.

K. Chen, M. Seuret, J. Hennebert, and R. Ingold. Convolutional neural networks for page segmentation of historical document images. In *International Conference on Document Analysis and Recognition*, 2017.

K. Chen, J. Wang, L.-C. Chen, H. Gao, W. Xu, and R. Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. In *arXiv preprint arXiv:1511.05960*, 2015b.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *Transactions on Pattern Analysis and Machine Intelligence*, 2018.

M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Conference on Computer Vision and Pattern Recognition*, 2014.

C. Clark and S. Divvala. Pdffigures 2.0: Mining figures from research papers. In *Joint Conference on Digital Libraries*, 2016.

C. Clausner, A. Antonacopoulos, and S. Pletschacher. International conference on document analysis and recognition2017 competition on recognition of documents with complex layouts-rdcl2017. In *International Conference on Document Analysis and Recognition*, 2017.

A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld. Document-level representation learning using citation-informed transformers. In *arXiv preprint arXiv:2004.07180*, 2020.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Empirical Methods in Natural Language Processing*, 2017.

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition*, 2016.

R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*, 2018a.

R. Das, T. Munkhdalai, X. Yuan, A. Trischler, and A. McCallum. Building dynamic knowledge graphs from text using machine reading comprehension. In *International Conference on Learning Representations*, 2018b.

Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, 2016.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, 2017.

K. Do, T. Tran, T. Nguyen, and S. Venkatesh. Attentional multilabel learning over graphs-a message passing approach. In *Machine Learning*, 2019.

D. Drivas and A. Amin. Page segmentation and classification utilising a bottom-up approach. In *International Conference on Document Analysis and Recognition*, 1995.

M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. In *International Journal of Computer Vision*, 2015.

S. Feng and M. F. Duarte. Graph autoencoder-based unsupervised feature selection with broad and local data structure preservation. In *Neurocomputing*, 2018.

A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Empirical Methods in Natural Language Processing*, 2016.

P. Gao, P. Lu, H. Li, S. Li, Y. Li, S. Hoi, and X. Wang. Question-guided hybrid convolution for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2018.

V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A convolutional encoder model for neural machine translation. In *arXiv preprint arXiv:1611.02344*, 2016.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, 2017.

S. Gidaris and N. Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Conference on Computer Vision and Pattern Recognition*, 2019.

R. Girshick. Fast r-cnn. In *International Conference on Computer Vision*, 2015.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2017.

A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning*, 2006.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. In *arXiv preprint arXiv:1410.5401*, 2014.

J. Gu, H. Zhao, Z. Lin, S. Li, J. Cai, and M. Ling. Scene graph generation with external knowledge and image reconstruction. In *Conference on Computer Vision and Pattern Recognition*, 2019a.

W. Gu, F. Gao, X. Lou, and J. Zhang. Link prediction via deep learning. In *arXiv preprint arXiv:1910.04807*, 2019b.

L. Guo, J. Liu, P. Yao, J. Li, and H. Lu. Mscap: Multi-style image captioning with unpaired stylized text. In *Conference on Computer Vision and Pattern Recognition*, 2019.

N. Gupta, K. Lin, D. Roth, S. Singh, and M. Gardner. Neural module networks for reasoning over text. In *International Conference on Learning Representations*, 2020.

D. Gurari, Q. Li, A. J. Stangl, A. Guo, C. Lin, K. Grauman, J. Luo, and J. P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Conference on Computer Vision and Pattern Recognition*, 2018.

J. Ha, R. M. Haralick, and I. T. Phillips. Document page decomposition by the bounding-box project. In *International Conference on Document Analysis and Recognition*, 1995a.

J. Ha, R. M. Haralick, and I. T. Phillips. Recursive xy cut using bounding boxes of connected components. In *International Conference on Document Analysis and Recognition*, 1995b.

E. Hamuda, B. Mc Ginley, M. Glavin, and E. Jones. Improved image processing-based crop detection using kalman filtering and the hungarian algorithm. In *Computers and Electronics in Agriculture*, 2018.

M. Haurilet, Z. Al-halah, and R. Stiefelhagen. Moqa - a multi-modal question answering model. In *European Conference on Computer Vision Workshop*, 2018.

M. Haurilet, Z. Al-Halah, and R. Stiefelhagen. Dyngraph: Visual question answering via dynamic scene graphs. In *German Conference on Pattern Recognition*, 2019a.

M. Haurilet, Z. Al-Halah, and R. Stiefelhagen. Spase - multi-label page segmentation for presentation slides. In *Winter Conference on Applications of Computer Vision*, 2019b.

M. Haurilet, A. Roitberg, M. Martinez, and R. Stiefelhagen. Wise - slide segmentation in the wild. In *International Conference on Document Analysis and Recognition*, 2019c.

M. Haurilet, A. Roitberg, and R. Stiefelhagen. It's not about the journey; it's about the destination: Following soft paths under question-guidance for visual reasoning. In *Conference on Computer Vision and Pattern Recognition*, 2019d.

M. Haurilet, A. Roitberg, S. Reiss, C. Seibold, M. Martinez, and R. Stiefelhagen. Nap: Semantic graph generation through the node association procedure. In *Submission*, 2021a.

M. Haurilet, A. Roitberg, and R. Stiefelhagen. Towards end-to-end visual question answering on entire pages. In *Submission*, 2021b.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016.

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *International Conference on Computer Vision*, 2017.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. In *Neural computation*, 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, 1997.

M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, 1990.

R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *International Conference on Computer Vision*, 2017.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, 2017.

D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Conference on Computer Vision and Pattern Recognition*, 2019.

A. Jabri, A. Joulin, and L. Van Der Maaten. Revisiting visual question answering baselines. In *European Conference on Computer Vision*, 2016.

S. K. Jauhar, P. Turney, and E. Hovy. Tabmcq: A dataset of general knowledge tables and multiple-choice questions. In *arXiv preprint arXiv:1602.03960*, 2016.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Conference on Computer Vision and Pattern Recognition*, 2017a.

J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick. Inferring and executing programs for visual reasoning. In *International Conference on Computer Vision*, 2017b.

K. Kafle, B. Price, S. Cohen, and C. Kanan. Dvqa: Understanding data visualizations via question answering. In *Conference on Computer Vision and Pattern Recognition*, 2018.

S. E. Kahou, V. Michalski, A. Atkinson, Á. Kádár, A. Trischler, and Y. Bengio. Figureqa: An annotated figure dataset for visual reasoning. In *International Conference on Learning Representations Workshop*, 2018.

A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision*, 2016.

A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Conference on Computer Vision and Pattern Recognition*, 2017.

D. Kim, Y. Yoo, J. Kim, S. Lee, and N. Kwak. Dynamic graph generation network: Generating relational knowledge from diagrams. In *Conference on Computer Vision and Pattern Recognition*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Neural Information Processing Systems*, 2015.

S. Kok and P. Domingos. Statistical predicate invention. In *International Conference on Machine Learning*, 2007.

R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi. Text generation from knowledge graphs with graph transformers. In *North American Chapter of the Association for Computational Linguistics*, 2019.

R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *International Journal of Computer Vision*, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*. 2012.

M. Kuhlmann and S. Oepen. Towards a catalogue of linguistic graph banks. In *Computational Linguistics*, 2016.

H. W. Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistics Quarterly*, 1955.

S. Kullback. Letter to the editor: The kullback-leibler distance. 1987.

F. Lebourgeois, Z. Bublinski, and H. Emptoz. A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents. In *International Conference on Pattern Recognition*, 1992.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, 1966.

H. Li, P. Wang, C. Shen, and G. Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. In *Association for the Advancement of Artificial Intelligence*, 2019.

J. Li, H. Su, J. Zhu, S. Wang, and B. Zhang. Textbook question answering under instructor guidance with memory networks. In *Conference on Computer Vision and Pattern Recognition*, 2018a.

Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang. Scene graph generation from objects, phrases and region captions. In *Conference on Computer Vision and Pattern Recognition*, 2017.

Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning deep generative models of graphs. In *arXiv preprint arXiv:1803.03324*, 2018b.

C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representations*, 2015.

M. Liao, J. Zhang, Z. Wan, F. Xie, J. Liang, P. Lyu, C. Yao, and X. Bai. Scene text recognition from two-dimensional perspective. In *Association for the Advancement of Artificial Intelligence*, 2019.

G. Lin, C. Shen, A. Van Den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2016.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2015.

C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016a.

J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *Neural Information Processing Systems*, 2016b.

M.-T. Luong, T. D. Nguyen, and M.-Y. Kan. Logical structure recovery in scholarly articles with rich document features. In *Multimedia Storage and Retrieval Innovations for Digital Library Systems*, 2012.

C. Ma, C. Shen, A. Dick, Q. Wu, P. Wang, A. van den Hengel, and I. Reid. Visual question answering with memory-augmented networks. In *Conference on Computer Vision and Pattern Recognition*, 2018.

L. Ma, Z. Lu, and H. Li. Learning to answer questions from image using convolutional neural network. In *Association for the Advancement of Artificial Intelligence*, 2016.

T. Ma and A. Zhang. Affinitynet: semi-supervised few-shot learning for disease type prediction. In *Association for the Advancement of Artificial Intelligence*, 2019.

F. Mahdisoltani, J. Biega, and F. M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *Conference on Innovative Data Systems Research*, 2013.

M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Neural Information Processing Systems*, 2014.

M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A deep learning approach to visual question answering. In *International Journal of Computer Vision*, 2017.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics: System Demonstrations*, 2014.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*, 2013.

M. Narasimhan, S. Lazebnik, and A. Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *Neural Information Processing Systems*, 2018.

G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision*, 2017.

A. Newell and J. Deng. Pixels to graphs by associative embedding. In *Neural Information Processing Systems*, 2017.

H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2015.

S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence*, 2018.

R. Parasuraman. Designing automation for human use: empirical studies and quantitative models. In *Ergonomics*, 2000.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, 2014.

E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Association for the Advancement of Artificial Intelligence*, 2017.

T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Conference on Computer Vision and Pattern Recognition*, 2017.

R. Qiao, L. Liu, C. Shen, and A. Van Den Hengel. Less is more: zero-shot learning from online textual documents with noise suppression. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Y. Qin, J. Du, Y. Zhang, and H. Lu. Look back and predict forward in image captioning. In *Conference on Computer Vision and Pattern Recognition*, 2019.

P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, et al. A large-scale evaluation of computational protein function prediction. In *Nature Methods*, 2013.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition*, 2016.

M. Ren, R. Kiros, and R. Zemel. Exploring models and data for image question answering. In *Neural Information Processing Systems*, 2015a.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems*, 2015b.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision*, 2015.

B. Sahbani and W. Adiprawita. Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system. In *International Conference on System Engineering and Technology*, 2016.

A. Sankar, X. Zhang, and K. C.-C. Chang. Meta-gnn: metagraph neural network for semi-supervised learning in attributed heterogeneous information networks. In *International Conference on Advances in Social Networks Analysis and Mining*, 2019.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Neural Information Processing Systems*, 2017.

A. G. Schwing and R. Urtasun. Fully connected deep structured networks. In *arXiv preprint arXiv:1503.02351*, 2015.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. In *AI Magazine*, 2008.

T. Shen, G. Lin, C. Shen, and I. Reid. Learning multi-level region consistency with dense multi-label networks for semantic segmentation. In *International Joint Conferences on Artificial Intelligence*, 2017.

K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2016.

M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Conference on Computer Vision and Pattern Recognition*, 2017.

M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, 2018.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards vqa models that can read. In *Conference on Computer Vision and Pattern Recognition*, 2019.

R. Smith. An overview of the tesseract ocr engine. In *International Conference on Document Analysis and Recognition*, 2007.

Z. Su, C. Zhu, Y. Dong, D. Cai, Y. Chen, and J. Li. Learning visual knowledge memory networks for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2018.

M. Suzuki, T. Kanahori, N. Ohtake, and K. Yamaguchi. An integrated ocr software for mathematical documents and its output with accessibility. In *Computers Helping People with Special Needs*, 2004.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, 2016.

X. Tao, Z. Tang, C. Xu, and Y. Wang. Logical labeling of fixed layout pdf documents using multiple contexts. In *International Workshop on Document Analysis Systems*, 2014.

D. Teney, L. Liu, and A. v. d. Hengel. Graph-structured representations for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2016.

C. Tensmeyer and T. Martinez. Document image binarization with fully convolutional neural networks. In *International Conference on Document Analysis and Recognition*, 2017.

K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing*, 2015.

L. R. Tucker. Some mathematical notes on three-mode factor analysis. In *Psychometrika*, 1966.

J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. In *International Journal of Computer Vision*, 2013.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Conference on Neural Information Processing Systems*, 2017.

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2017.

K. Wang and S. Belongie. Word spotting in the wild. In *European Conference on Computer Vision*, 2010.

Q. Wang, Z. Zhou, L. Huang, S. Whitehead, B. Zhang, H. Ji, and K. Knight. Paper abstract writing through editing mechanism. In *Conference of the Association for Computational Linguistics*, 2018.

T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *International Conference on Pattern Recognition*, 2012.

T. Wang, Y. Zhu, L. Jin, C. Luo, X. Chen, Y. Wu, Q. Wang, and M. Cai. Decoupled attention network for text recognition. In *arXiv preprint arXiv:1912.10205*, 2019.

T. C. Wei, U. Sheikh, and A. A.-H. Ab Rahman. Improved optical character recognition with deep neural network. In *International Colloquium on Signal Processing & Its Applications*, 2018.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, 1992.

Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Visual question answering: A survey of methods and datasets. In *Computer Vision and Image Understanding*, 2017.

Y. Xian, B. Schiele, and Z. Akata. Zero-shot learning-the good, the bad and the ugly. In *Conference on Computer Vision and Pattern Recognition*, 2017.

H. Xie, S. Fang, Z.-J. Zha, Y. Yang, Y. Li, and Y. Zhang. Convolutional attention networks for scene text recognition. In *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2019.

C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*, 2016.

W. Xiong, T. Hoang, and W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Empirical Methods in Natural Language Processing*, 2017.

D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *Conference on Computer Vision and Pattern Recognition*, 2017.

H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, 2016.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, 2015.

K. Xu, L. Song, Y. Feng, Y. Song, and D. Yu. Coordinated reasoning for cross-lingual knowledge graph alignment. In *Association for the Advancement of Artificial Intelligence*, 2020.

G. R. Yang, I. Ganichev, X.-J. Wang, J. Shlens, and D. Sussillo. A dataset and architecture for visual reasoning with a working memory. In *European Conference on Computer Vision*, 2018a.

J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph r-cnn for scene graph generation. In *European Conference on Computer Vision*, 2018b.

S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *International Conference on World Wide Web*, 2011.

X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, and C. L. Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2017.

X. Yang, K. Tang, H. Zhang, and J. Cai. Auto-encoding scene graphs for image captioning. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *Conference on Computer Vision and Pattern Recognition*, 2016.

T. Yao, Y. Pan, Y. Li, and T. Mei. Exploring visual relationship for image captioning. In *European Conference on Computer Vision*, 2018.

D. Yu, J. Fu, T. Mei, and Y. Rui. Multi-level attention networks for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2017.

F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *Conference on Computer Vision and Pattern Recognition*, 2015.

Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian. Deep modular co-attention networks for visual question answering. In *Conference on Computer Vision and Pattern Recognition*, 2019.

F. Zhan and S. Lu. Esir: End-to-end scene text recognition via iterative image rectification. In *Conference on Computer Vision and Pattern Recognition*, 2019.

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *arXiv preprint arXiv:1805.08318*, 2018.

S. Zhang, Y. Qin, K. Sun, and Y. Lin. Few-shot audio classification with attentional graph neural networks. In *Interspeech*, 2019.

Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *International Conference on Computer Vision*, 2015.

Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *Conference on Computer Vision and Pattern Recognition*, 2016.

A Examples - Soft Paths

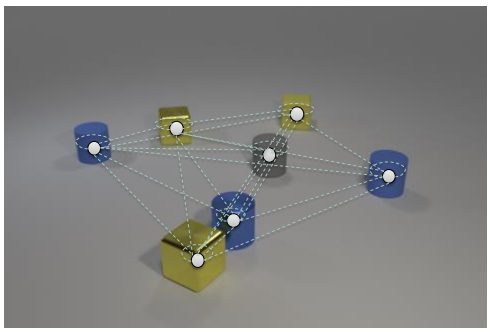
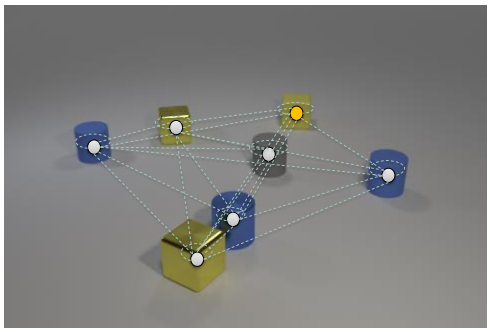
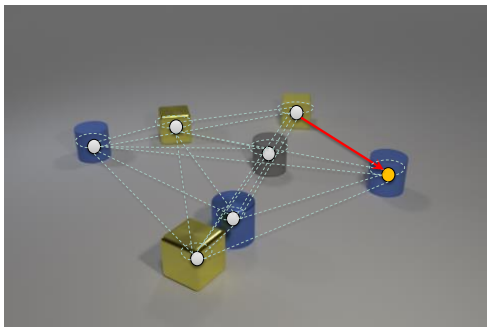


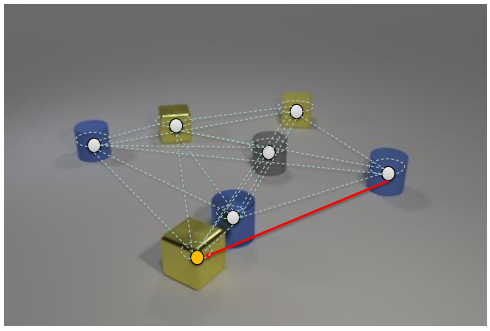
Image There is a tiny rubber thing that is right of the matte cube; are there any yellow cubes in front of it?



Step 1 There is **a** tiny rubber thing **that is right of the matte cube**; are there any yellow cubes in front of it?



Step 2 There **is** a **tiny rubber thing** that is **right of the** matte cube; are there any yellow cubes in front of it**?**



Step 3 There is a tiny rubber thing that is right of the matte cube; are there any yellow **cubes in front of it**? *Answer: yes* ✓

Figure A.1: Existence question with a necessary reasoning chain length of three.

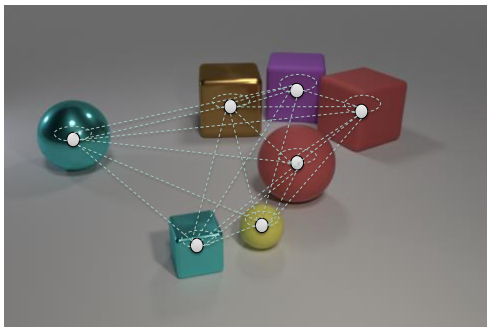
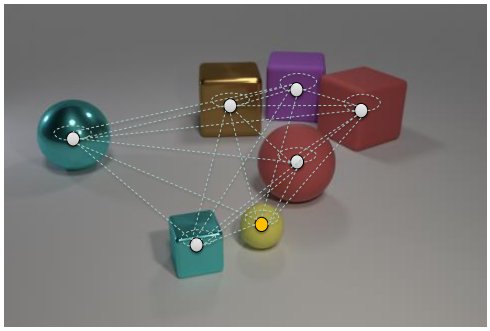
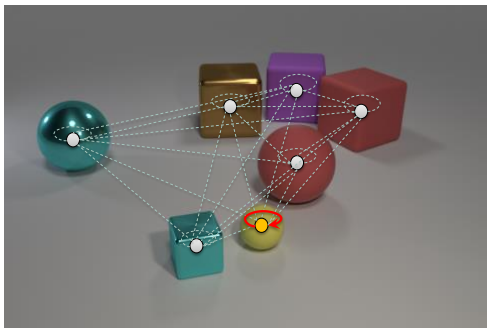


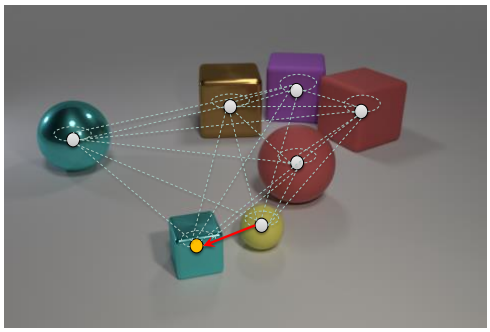
Image Are there any small cyan objects on the left side of the yellow object?



Step 1 Are there any small cyan objects on the **left** side **of the yellow object?**



Step 2 Are there any small cyan objects on the left side of **the** yellow object?



Step 3 Are there **any small cyan objects** on the **left** side **of** the yellow object?

Figure A.2: Existence question with a necessary reasoning chain length of two and with a single found destination. *Answer: yes* ✓

In this section of the appendix, we analyze the soft paths generated by our model for both correct and incorrect answers. Figures A.1 to A.8 provide examples for maximum path length $r = 3$, with each column illustrating the state at step i. White circle markings depict the nodes with *low* confidence of being visited by the traveler in the current step i. Orange markings depict *high* probability nodes and the arrows mark a high transition confidence (> 0.5). Underneath each image, we highlight the words in the question that received high attention values (> 0.5) in the visual guide for the current path section.

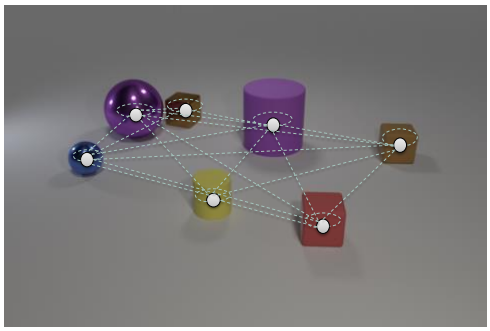
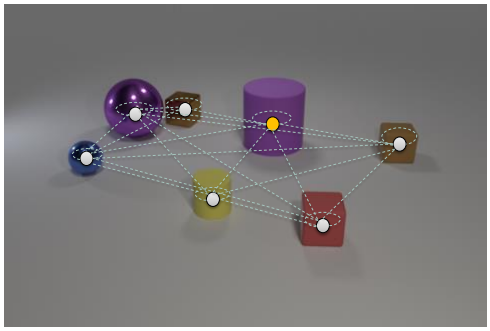
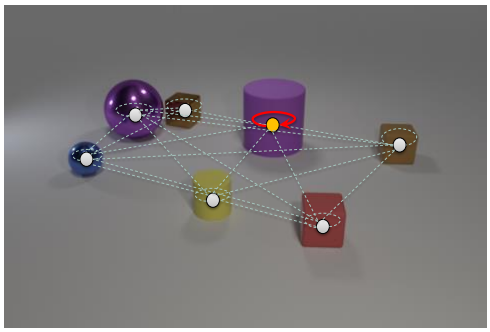


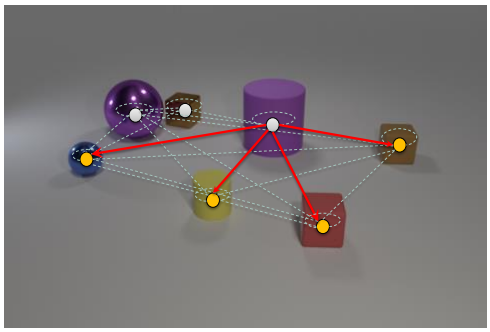
Image Are there any tiny objects in front of the purple rubber object?



Step 1 Are there any tiny objects in front **of the purple rubber object?**



Step 2 Are there any tiny objects in front of **the purple** rubber object?



Step 3 Are there **any tiny objects in front of** the purple rubber object? *Answer: yes* ✓

Figure A.3: Existence-type question with multiple found destinations. All the destination nodes identified by our model are correct.

Correct predictions. Figures A.1 to A.4 show a variety of the *existence* task queries, which were correctly handled by our network. In the first example (Figure A.1), the number of required reasoning steps corresponds to the maximum path length r and our model easily finds the path leading to the correct destination. The next example (Figure A.2), on the other hand, only requires two reasoning steps while our model has to traverse exactly $r = 3$ steps. This is, however, not an issue in our approach, as we allow self-loops. The self-loop is present in the second time step on only a single node (the *yellow* sphere). This confirms, that the model leverages object attributes for traversal, as only edges with a *yellow* target have high transition confidence. Figure A.3 correctly found four destinations that are in front of the purple rubber object as multiple objects fit the query.

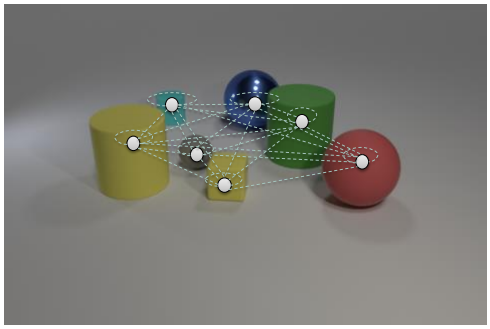
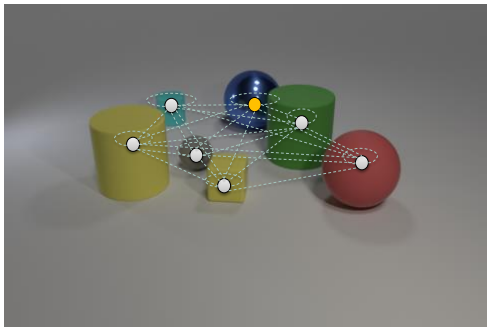
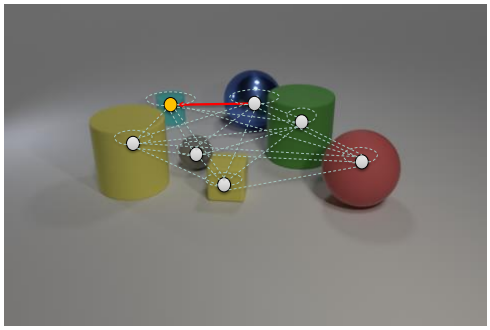


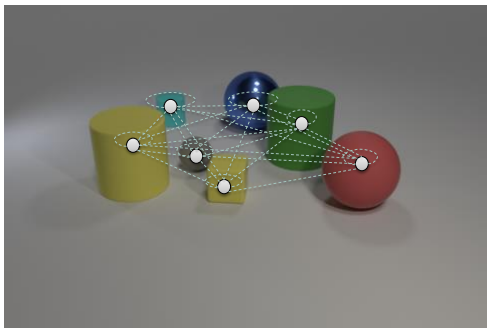
Image Is there a big blue metal thing that is behind the rubber object behind the blue shiny object?



Step 1 Is there a big blue metal thing that is **behind the rubber object behind the blue shiny object?**



Step 2 Is there a big blue metal thing that is behind **the rubber object** behind the blue shiny object?



Step 3 Is there **a big blue metal thing** that is **behind** the rubber object behind the blue shiny object? *Answer: no* ✓

Figure A.4: Example of question about the existence of an object where the correct answer is 'no' (*i.e.*, there are no such destinations).

In case that the query addresses a non-existent object (*i.e.*, the correct answer is 'no'), our model does not have any high confidence destination nodes, as illustrated in the Figure A.4. The traveler followed the correct path from the *blue shiny object* to the *rubber object behind* it. However, there are no other objects behind this rubber object and, thus, there are no final high confidence destination nodes.

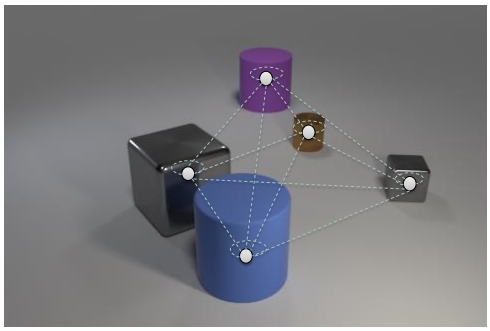
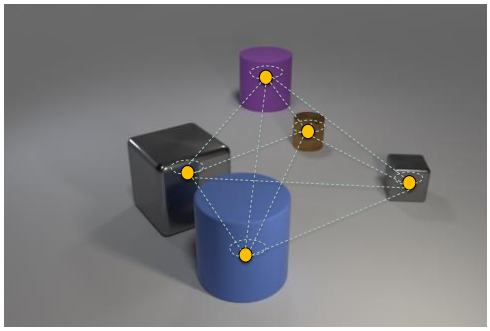
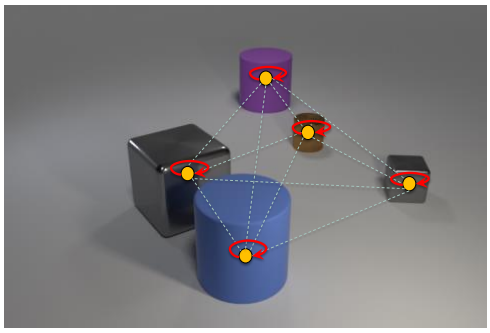


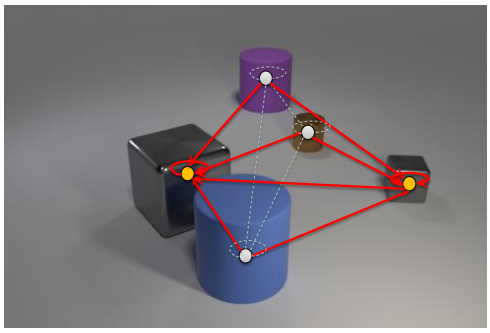
Image How many cubes are there?



Step 1 How many cubes are there?



Step 2 How many **cubes are** there?



Step 3 How many **cubes are there**?

Figure A.5: Counting task with a reasoning chain length of one. *Answer: 2* ✓

Figures A.5 and A.6 visualize the paths produced for *counting*. Both examples necessitate shorter reasoning chains than our maximal path length (*i.e.* 'How many cubes are there?' only requires a single reasoning step). The traveler handles this successfully by visiting some nodes more than once. Surprisingly, we observe different strategies for different lengths of the required reasoning chain. In case of the path of length 1 (Figure A.5) the traveler visits all nodes twice and chooses the nodes of the type cube in the last step.

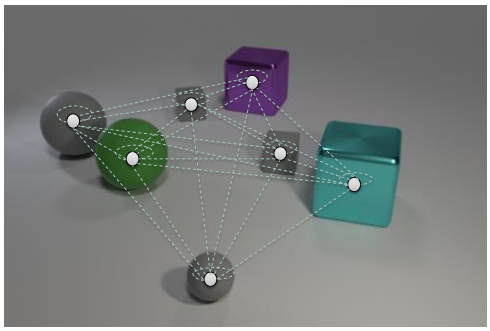
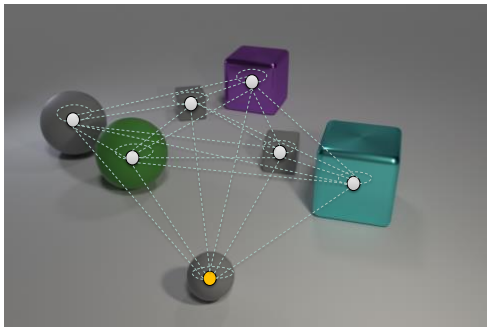
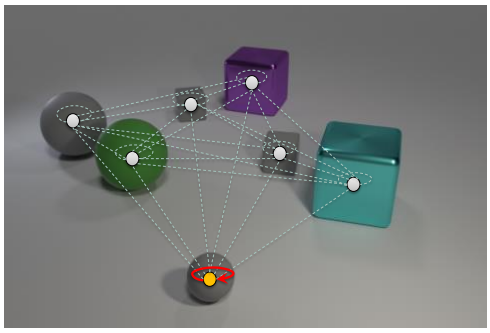


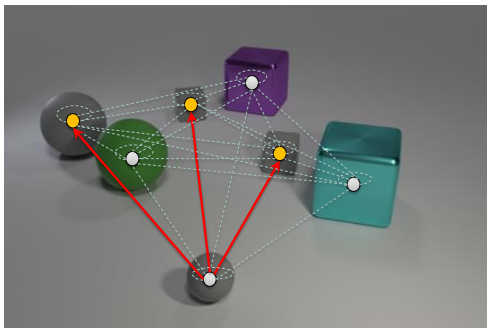
Image What number of other things are the same color as the small matte ball?



Step 1 What number of other things are the same color **as the small matte ball**?



Step 2 What number of other **things are** the same **color as** the small matte ball?



Step 3 What number of **other things are** the **same color as** the small matte ball?

Figure A.6: Counting question with a necessary path length of two. The model was able to find all three destinations. *Answer: 3* ✓

Figure A.6 shows an example for a reasoning chain with two steps. To address this difficulty, our soft paths network selects a single source node (the same small matte ball), *i.e.*, the model leveraged the self loop.

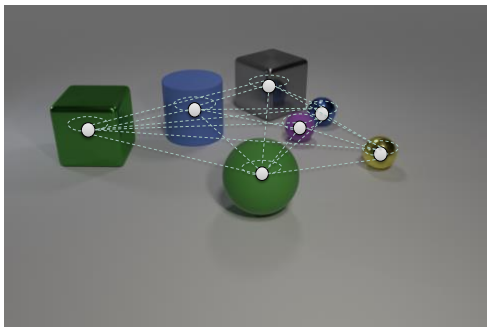
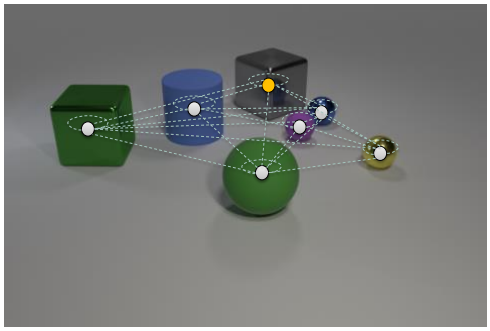
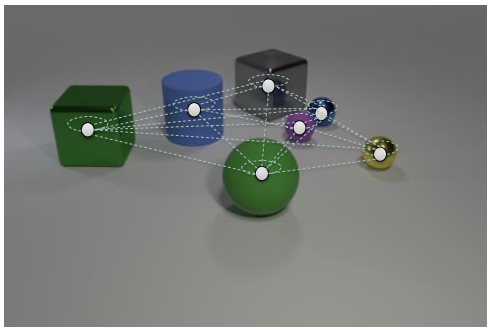


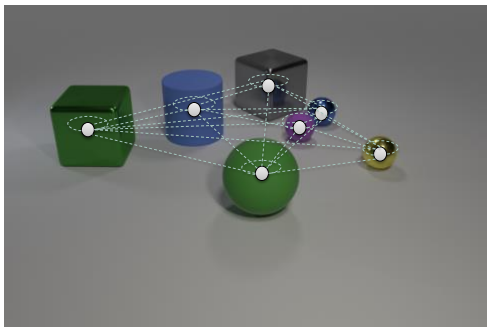
Image Are there any big green things that are in front of the large metal object that is on the left side of the gray thing?



Step 1 Are there any big green things that are in front **of the** large metal **object that is on the left side of the gray thing?**



Step 2 Are there any big green things that are in front **of the large metal object** that is on the **left** side of the gray thing?



Step 3 Are there **any big green things** that are **in front of** the large metal object that is on the left side of the gray thing? Answer: no ✗

Figure A.7: Example of an incorrectly answered question. Since the large metal object left of the starting node was not found in the second step (computed confidence of 0.3 below our threshold of 0.5), the model also did not find any destination nodes.

Analysis of incorrect predictions. In Figures A.7 and A.8, we analyze queries which were not answered correctly. In the *existence* task example (Figure A.7), the traveler was not able to find the large metal object on the left side of the source node, as it produced a confidence of 0.3 (our threshold was 0.5). Due to this deficit in an intermediate step, the traveler did not reach the correct destination node (the big green object) and is predicting the incorrect answer 'no'.

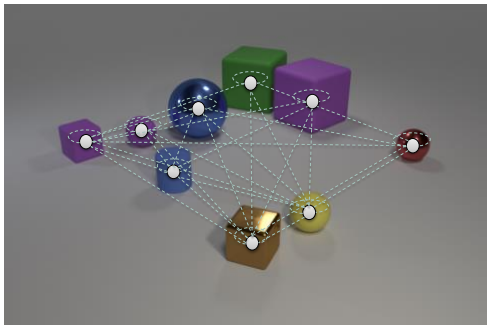
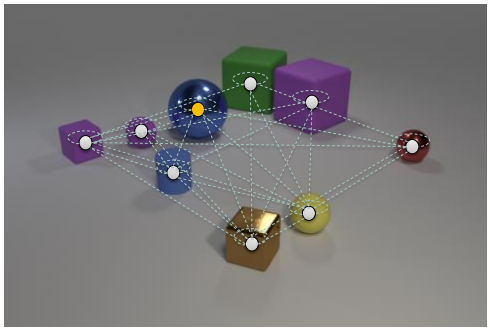
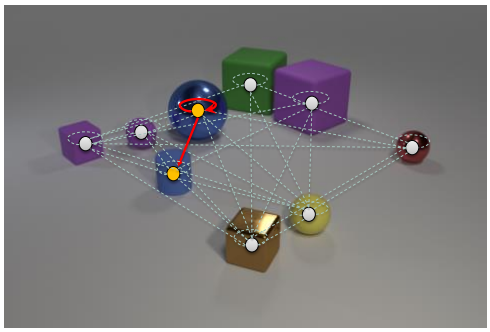


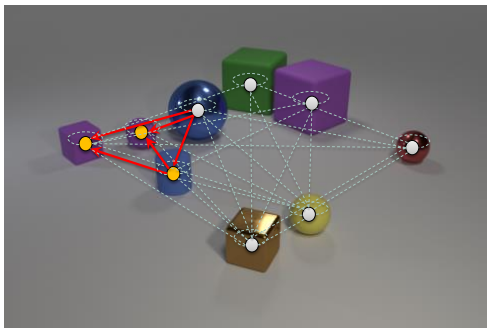
Image How many matte things are on the left side of the blue thing on the left side of the large blue metal thing?



Step 1 How many matte things are on the **left side of the blue thing on the left** side of the large blue metal thing?



Step 2 How many **matte things** are on the **left** side of the blue thing on the left side of the large blue metal thing?



Step 3 How many **matte things** are **on** the **left** side of **the** blue thing on the left side of the large blue metal thing? Answer: 3 ✗

Figure A.8: Due to an incorrect self-loop in the second step, presumably retained because the object satisfies one of the conditions of the next step (it is blue), the answer for the counting question was higher than required (correct answer is 2).

Figure A.8 is an example of *counting*, where our model found one object more than necessary. First, the traveler correctly chose the blue sphere as its starting point. Then, the traveler should have moved to the blue object left of it, as stated in the query. While our model did this transition, it also wrongfully retained a self-loop to an object from the previous step. We suppose, this relation was kept, as this source object met one of the two conditions for the next step: it is blue (but *not* left of itself, which is the second condition). Finally, our model considers destinations left of both, the small cylinder and the incorrectly visited blue sphere, resulting in three counted objects, instead of two.

B Text Recognition

B.1 FGFE network

In this chapter, we address the text recognition task and introduce a neural architecture for text recognition in form of mathematical formulas (overview in Figure B.1). While we evaluate our approach for mathematical expression recognition, our network can be directly applied for text recognition in natural language. Our model first embeds the input image containing a mathematical formula using the *fine-grained* feature extractor FGFE (Figure B.2). The produced features are then passed to the encoder, which maps them to (1) local features and to (2) a global representation of the image. Based on both the global and the local features, the decoder constructs the predicted formula via an LSTM with an attention mechanism step-by-step. The formula thereby is represented as LaTeX tokens of atomic size, *e.g.*, 'a', '\begin{array}' or '\frac'. We choose this token-wise format over a character-based representation, as this avoids irregular results and reduces the formula length. Next, we define the modules of our proposed architecture.

Publications. In this chapter, we present a text recognition approach previously published in [Bender*, Haurilet* et al., 2019].

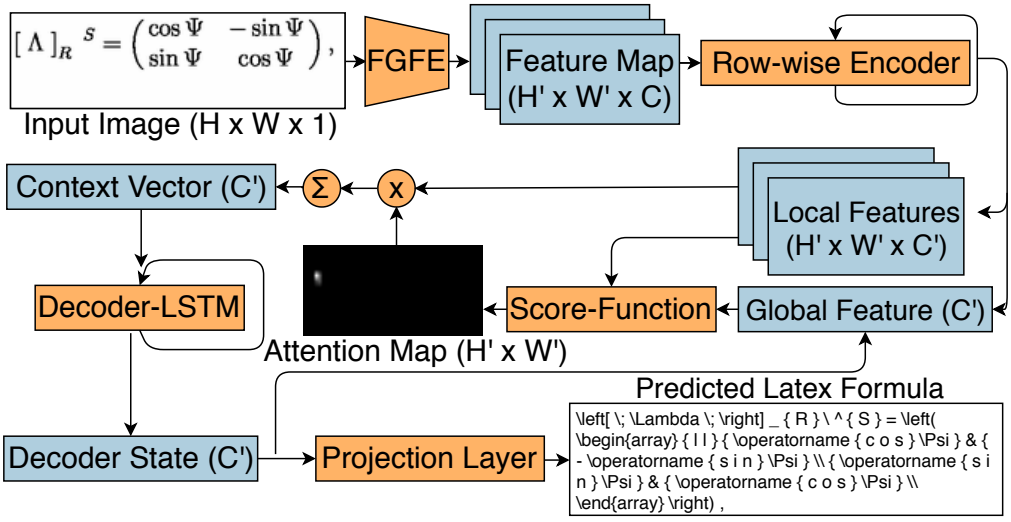


Figure B.1: Our neural architecture for generating mathematical expressions based on an input image. The core modules of our model are: (1) a feature extractor based on a CNN for obtaining the feature maps; (2) a relational encoder that embeds the feature cells into a new representation; (3) a decoder that generates the LaTeX markup using the embeddings produced by the relational encoder.

Fine-grained feature extractor. Our fine-grained feature extractor consists of a Convolutional Neural Network (CNN) that is able to compute useful high-level features from the pre-processed input gray-scale image of size $h \times w$. Through the number of layers and especially the size of the stride of each convolutional or pooling layer, the receptive field and, thereby, the granularity of the features can be adjusted. While in case of image captioning on natural images, large receptive fields are conventional used [Xu et al., 2015; Yao et al., 2018], we argue that for mathematical expression recognition as we have very small symbols, a fine-grained representation benefits the neural network.

As we see in Figure B.2, FGFE comprises three CNN blocks which consist of two to four convolution layers, a max-pooling layer, and finally batch-normalization is applied. In order not to reduce the size of the image too fast and, thus, keep the fine-grained structure of the mathematical expression, 2×1 and 1×2 max-pooling

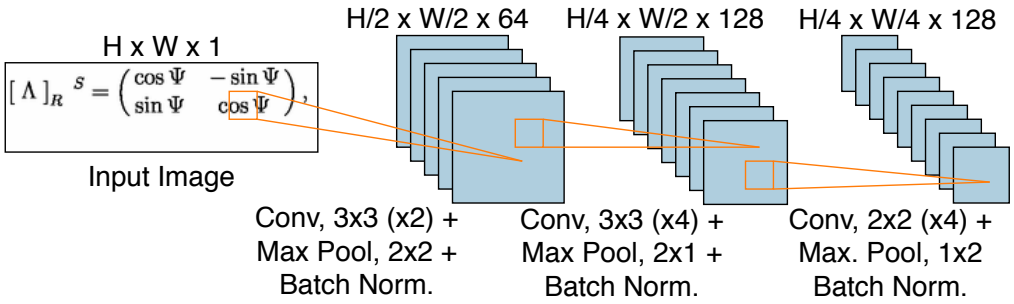


Figure B.2: The Fine-Grained Feature Extractor (FGFE) consists of three blocks: the first consists of two convolutional layers, while the latter ones contain four convolutions each. We apply max pooling for increasing the receptive field and normalize the feature maps using batch norm.

layers are applied with a stride of 2×1 and 1×2 in the second and the third block, respectively. For an input image of dimensions $h \times w$, our FGFE produces an output of the shape $h/4 \times w/4 \times 128$, *i.e.*, only slightly reducing the image size.

Encoder. To translate a variable-sized three dimensional feature map into a formula, we leverage an encoder-decoder module. The encode operation maps the visual feature map to a global feature and a refined local representation. An LSTM is used row-wise on the feature map and then by concatenating the outputs of the rows, we obtain the final local representation. For the global feature, we average all final states of each row processed by the encoder LSTM.

Attention-based decoder. Finally, we use the local and global visual representations to generate the underlying markup by applying an LSTM with attention mechanism. In every step, we get a score value for every local feature using an additive scoring function, which compares the local feature with the current state of the decoder LSTM (see Figure B.1). We model a probability distribution over the local features by normalizing these scores using softmax and, then, we use them as weights for the local features. Next, we average these weighted local embeddings and obtain a single fixed-dimensional context vector that is fed together with the

current decoder state into the decoder LSTM cell. The projection layer then maps the new decoder state to a probability distribution over the token classes. During prediction, we select the token class with the highest activation in each decoding step.

Learning setting. We minimize the cross entropy loss using stochastic gradient decent on the IM2LATEX-100K dataset for 15 epochs. As [Deng et al., 2017], we start with an initial learning rate of 0.1 and decay it by 10 every time the validation loss does not improve. For a stable training procedure, we feed the embedding of the correct token from the last decoding step into the decoder LSTM. During test time, we use beam search [Abdou and Scordilis, 2004] with a width of 5 for decoding in order to reduce search errors.

B.2 Evaluation

In this section, we compare our model to related work based on both token- and image-based evaluation metrics. Then, we discuss the performance of our models for different formula lengths and show that even though the accuracy drops for very large expressions, we are still able to produce frequently the correct prediction. We analyze the performance for different formulas based on their structure types, *e.g.*, matrices and fractions, and the impact of rare token classes on accuracy. Finally, we provide some example predictions with the corresponding attention maps where we show that our model can generate highly-structured formulas.

Evaluation metrics. To evaluate our network, we leverage the same token- and image-based metrics as related work [Deng et al., 2017]. While *token-based metrics* compare the predicted token sequence *pred* with the ground truth markup *gt*, *image-based metrics* use images *predImg* and *gtImg*, images that were generated by the LaTeX compiler from the predicted and ground truth markup.

1. Absolute Token Accuracy (ABS_T**)** is a metric that calculates the accuracy of *exact matches* between the ground truth and prediction in LaTeX format.

2. Token Edit Distance Accuracy (EDA_T) softens the ABS_T by rewarding predictions that are similar to the ground truth:

$$EDA_T(\text{pred}, \text{gt}) = 1 - \frac{\delta(\text{pred}, \text{gt})}{\max(|\text{pred}|, |\text{gt}|)} \tag{B.1}$$

where δ denotes the Levenshtein distance [Levenshtein, 1966] between $pred$ and gt, and $|x|$ denotes the number of tokens of the sequence x. The Levenshtein distance is defined as the number of edit operations (comprising remove, swap, add) that are necessary to map pred to gt.

3. Absolute Image Accuracy (ABS_I) is similar to the token-based absolute accuracy where we have a correct prediction if $predimg$ and $gtimg$ match exactly. As in related work [Deng et al., 2017], since white spaces have no semantic meaning in mathematical expressions, for the comparison we remove columns consisting only of whitespace in predimg and gtimg.

4. Image Edit Distance Accuracy (EDA_I) is calculated by first converting the images $predimg$ and $gtimg$ to arrays of booleans based on the threshold 0.5. The final EDA_I is defined as the average row-wise Levenshtein distance as depicted in Eq. B.1.

5. BLEU is a popular metric used to evaluate distances between sentence pairs in natural language (*e.g.*, in image captioning). BLEU is defined as the geometric mean of BLEU-1 to BLEU-4 times a brevity penalty. BLEU-n, thereby, calculates the n-gram overlap between both sentences (*i.e.*, precision), while the brevity penalty penalizes short sentences (*i.e.*, recall).

Overall results on IM2LATEX-100K. Next, we provide the final results of our model on the test set of the large scale IM2LATEX-100K dataset (see Table B.1). Since the LaTeX compiler is not an injective function (*i.e.*, different LaTeX markups can generate identical images), a text-only metric can have difficulties to provide a good comparison between mathematical expressions. Thus, we provide results for both token- as well as image-based metrics.

Approach	Attention	*BLEU*	EDA_T	ABS_T	EDA_I	ABS_I
Classical Methods and Baselines						
Prior	–	0.0	20.0	0.0	85.0	0.0
INFTY [Suzuki et al., 2004]	–	66.7	–	–	–	26.7
Deep Learning Architectures						
CTC [Graves et al., 2006]	–	30.4	–	–	–	9.2
Caption [Xu et al., 2015]	softmax	75.0	–	–	–	55.7
Im2Tex [Deng et al., 2017]	hierarch.	86.2	–	–	–	79.6
	hard	87.1	–	–	–	77.1
	sparsemax	87.0	–	–	–	78.1
	softmax	87.7	92.1	41.2	88.6	79.9
Ours	softmax	**90.3**	**92.8**	**46.8**	**93.1**	**84.3**

Table B.1: Performance on the IM2LATEX-100K dataset based on five metrics. We group the methods into: deep learning and non-deep learning methods.

Table B.1 shows the results of our prior baseline which always produces the most frequent tokens in the training data. More precisely, in case of the image-based metrics, the prior model produces an image containing the most frequent pixel, *i.e.*, only white pixels. For the token-based metrics, the prior classifier always predicts a formula of the average formula length from the training data, *i.e.*, consisting of 52 tokens. The first half contains only left curly bracket tokens, as these are at the first positions the most frequent ones, while in the second half the most frequent token is the right curly bracket. Not surprisingly, as we have a very large number of different classes the baseline shows 0% for *BLEU*, ABS_I, and ABS_T, while in case of the easier EDA_T and EDA_I we achieve a significantly better performance of about 20% and 85%, respectively.

The INFTY Reader applied to mathematical expression recognition easily beats our prior baseline by a large margin (*i.e.*, from 0% to 26.7% ABS_I), while CTC improves the performance of the baseline by 9%. In comparison, the Caption method which does not use an encoder module, shows a further improvement compared to INFTY. The Im2Tex deep learning method that uses both an encoder and an attention-based decoder, is able to show very promising results improving the Caption network

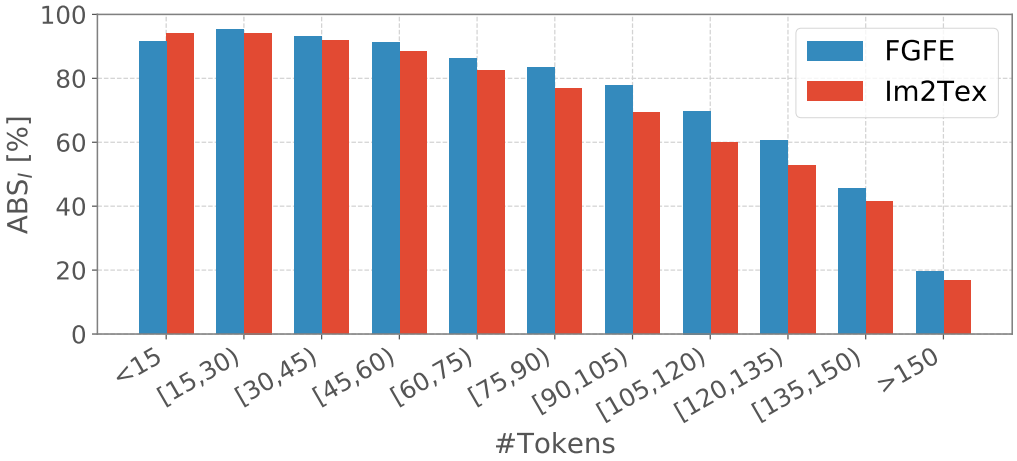


Figure B.3: Overview of the impact of formula length on performance. While short formulas almost achieve perfect accuracy (over 95%), long formulas still follow short with a performance of around 20%.

by over 20% ABS_I. As we see in Table B.1, different attention mechanisms are evaluated on the Im2Tex architecture: (1) a hierarchical attention that combines different layer outputs of the encoder, (2) hard, and (3) sparse max which both use discretized values and, thus, have to be trained using reinforcement learning. Nonetheless, the popular (4) softmax normalization shows the best performance in comparison the other proposed attention modules, having an overall improvement of over 50% over the widely used INFTY model in Image Absolute Accuracy.
Finally, our model achieves state-of-the-art results of 84.3% in Image Absolute Accuracy outperforming Im2Tex by over 4% and the Caption model by almost 30%. We want to note that the gap between our model and related work is much greater in absolute accuracies than in the edit distance metrics. This is not surprising, as the edit distance metrics are by far easier, as we also see the small gap in edit distance of our prior baseline to the other approaches. Nonetheless, we were able to improve the results in the edit distances as well as the popular text-based BLEU metric.

Impact of the formula length on performance. In Figure B.3, we analyze the impact of formula length on the image absolute accuracy of our neural architecture. As expected, we experience a drop in performance for longer formulas, *e.g.*, for

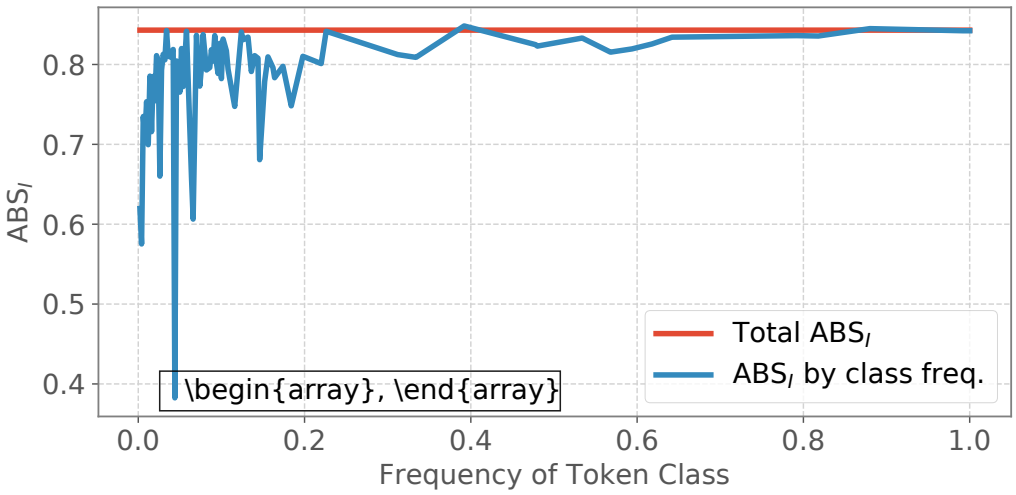
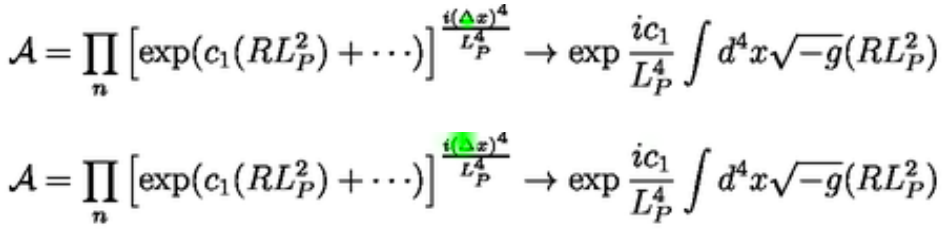


Figure B.4: Impact of rare token classes on the absolute image accuracy.

formulas under 100 tokens we achieve an accuracy of 80%, while for mathematical expressions over 150 tokens we obtain around 20%. We also note that very short formulas (*i.e.*, shorter than 15 tokens) have a worse performance than medium-sized ones (*i.e.*, between 15 and 30 characters). The reason for these results is presumably due to the imbalance in the training data, since short formulas are by far less frequent (under 3% of mathematical expressions) than the longer ones (over 13% of formulas have a length between 15 and 30 tokens).

Impact of rare token classes. In Figure B.4, we show the relation between the average ABS_I of images containing a token and the frequency of the particular token classes. Thus, the Y-axis shows the absolute image accuracy of all images containing the particular token class at least once. The X-axis corresponds to its frequency, *i.e.*, percentage of images that contain at least one appearance of the particular token class in the ground truth data.

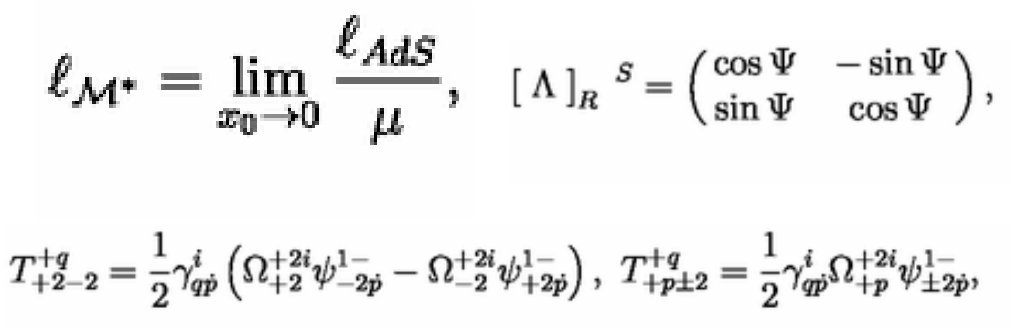
We notice that there is a strong correlation between the frequency of the token classes and the prediction performance of our model: the fewer training samples we have at our disposal, the lower is our accuracy. Moreover, formulas which contain at least one of the 61 worst recognized token classes, are never predicted correctly and have a frequency of less than 1% in the training set. The reason that ABS_I does



$$\mathcal{A} = \prod_{n} \left[\exp(c_1(RL_P^2) + \cdots) \right]^{\frac{i(\Delta x)^4}{L_P^4}} \rightarrow \exp \frac{ic_1}{L_P^4} \int d^4x \sqrt{-g}(RL_P^2)$$

$$\mathcal{A} = \prod_{n} \left[\exp(c_1(RL_P^2) + \cdots) \right]^{\frac{i(\Delta x)^4}{L_P^4}} \rightarrow \exp \frac{ic_1}{L_P^4} \int d^4x \sqrt{-g}(RL_P^2)$$

Figure B.5: Comparison of the attention maps of FGFE when generating the token '\Delta' (top) and of Im2Tex when incorrectly producing the token '\alpha' (bottom).



$$\ell_{\mathcal{M}^*} = \lim_{x_0 \to 0} \frac{\ell_{AdS}}{\mu}, \quad [\,\Lambda\,]_R{}^S = \begin{pmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{pmatrix},$$

$$T^{+q}_{+2-2} = \frac{1}{2}\gamma^i_{q\dot{p}} \left(\Omega^{+2i}_{+2} \psi^{1-}_{-2\dot{p}} - \Omega^{+2i}_{-2} \psi^{1-}_{+2\dot{p}} \right), \; T^{+q}_{+p\pm 2} = \frac{1}{2}\gamma^i_{q\dot{p}} \Omega^{+2i}_{+p} \psi^{1-}_{\pm 2\dot{p}},$$

Figure B.6: Examples of correctly predicted mathematical expressions containing arrays, subscripts, superscripts and fractions.

not drop to 0% when reaching rare tokens with a frequency of nearly 0 is that many symbols have exact synonyms (*i.e.*, predicting the exact synonym instead of a rare token does not impact ABS_I).

Importance of a fine-grained visual representation. In Figure B.5, we show an example attention map that was produced by our model (top) and by the architecture in related work [Deng et al., 2017] (bottom). Since the formulas contain sub- and superscripts, many characters are smaller than the default token size. Nonetheless, the decoder using the attention maps of FGFE is able to concentrate its weights to the small delta character, leading to a correct prediction of the '\Delta' token. In comparison, the decoder applied on the coarser features of related work, was not able to focus on a single character, but on multiple characters simultaneously. Thus, this unwanted fusion of features of multiple different characters lead the model to produce erroneously an '\alpha' token.

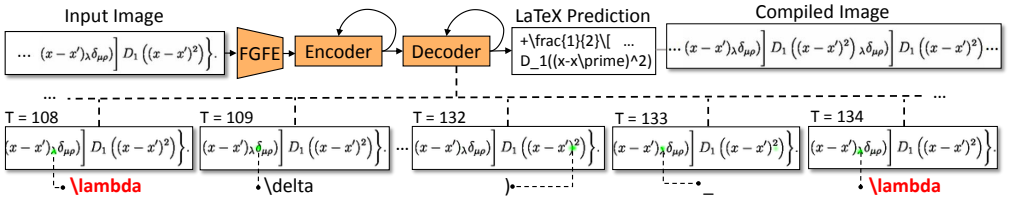


Figure B.7: Recursive behavior of the model for long expressions. This example shows correctly predicted tokens up to time step $T = 132$. For $T = 133$, the model loops back to the previously generated lambda causing a recursion in the prediction.

Analysis of highly-structured mathematical expressions. Highly-structured formulas still constitute a problem for the model, as we show in Figure B.4 where we experience a strong drop in performance due to expressions of the matrix type (*i.e.*, arrays). Even though array tokens have around 5k sample images, the network has difficulties in learning these complex structures. Nonetheless, the model is often able to correctly predict such complex structures (see examples in Figure B.6).

Discussion on typical errors. Additionally to difficulties due to highly-structured mathematical formulas, we encounter other two types of errors: (1) simple errors caused by brackets and (2) a subsequence of the formulas are generated multiple times. The former error type is not caused by the model forgetting to close an open bracket (which occurs less than 1% of the time), but *when* it is closed.

Recursive behavior in long formulas. The former error type revolves around long formulas. In some long expressions, the network produces infinite many times a subsequence of the formula, *i.e.*, the model stops only when the maximum number of time steps is reached. In Figure B.7, we show an example of a formulas on which the model generates an infinite loop of the following subsequence: '$_\lambda\sigma_{\mu\rho})]D_1((x - x\prime)^2)$'. After predicting the final round bracket, the model jumps back to generating the lambda from previous steps. When visualizing this behavior, we see that even the attention module focuses on the previously predicted token (*i.e.*, the lambda in our example). The probable culprit of these loops is the decoder,

which is not able to keep track on the already generated tokens. Nonetheless, these errors occur only on very long formulas, where the decoder LSTM has to memorize a large number of previous steps.

B.3 Summary and discussion

In this chapter, we developed a deep learning architecture for text sequence generation. We proposed a fine-grained feature extractor that is trained in an end-to-end manner with a relational encoder and a decoder with an attention mechanism using softmax normalization. Since the symbols have a large range of different sizes, a small receptive field in the feature extractor benefits our model, especially as we use an encoder on the produced feature maps that is able to link regions of larger characters. We show the strength of our network on a large scale dataset for mathematical formula recognition, where our model is able to learn nontrivial spatial relations like fractions and arrays (see Figure B.6). Furthermore, the model is able to predict some remarkably complex formulas, which are longer than formulas contained in the training set. Our model shows strong results on all metrics outperforming current state-of-the-art by over 4% in absolute image accuracy.

C FUSE Network

Table C.1 shows the structure of the FUSE architecture. We group the network into four parts: (1) The visual embedding module encodes the image using convolution layers where the receptive field is strongly enlarged by increasing the stride. (2) In a similar manner, the question is encoded via convolutions. In contrast to images, we pass the words of the query as input (*i.e.*, encompassing a matrix) and, thus, we leverage 1D convolutions over the word vectors. To obtain a final global representation, we average the encodings into a single vector. (3) We then combine the image and question features via concatenation, where we append the query on each visual slice over the location dimensions. We re-encode this multi-modal encoding using 1×1 convolutions followed by sum pooling. (4) Finally, as in other VQA models, we use several fully-connected layers to encode the feature vector from the previous step. The output of the last fully-connected layer models a probability distribution over the answers seen during training.

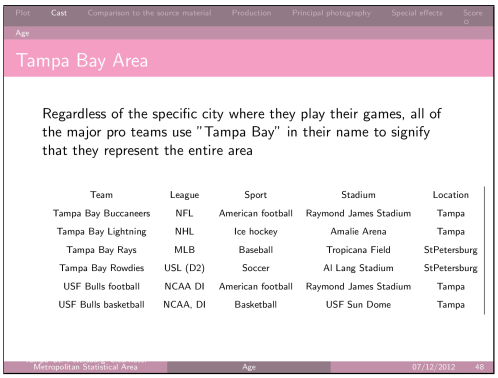
Module	Output size	Layers
(1) Visual embedding	134×112×32	*conv*(3×3, 32, ReLU, str=1) *conv*(3×3, 32, ReLU, str=1) *conv*(3×3, 32, ReLU, str=2)
	67×56×64	*conv*(3×3, 64, ReLU, str=1) *conv*(3×3, 64, ReLU, str=1) *conv*(3×3, 64, ReLU, str=2)
	33×28×128	*conv*(3×3, 128, ReLU, str=1) *conv*(3×3, 128, ReLU, str=1) *conv*(3×3, 128, ReLU, str=2)
	16×14×128	*conv*(3×3, 128, ReLU, str=1) *conv*(3×3, 128, ReLU, str=1) *conv*(3×3, 128, ReLU, str=2)
(2) Question encoder	12×32	*conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=2)
	6×32	*conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=2)
	3×32	*conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=2)
	3×32	*conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=1) *conv1D*(3, 32, ReLU, str=1)
	32	*sum* over variable dim.
(3) Multi-modal module	16×14×160	*concat*(question, image) = y
	16×14×200	*conv*(1×1, 64, ReLU, str=1) *conv*(1×1, 64, ReLU, str=1) *conv*(1×1, 200, softmax) = A
	200×160	global represent.: $\sum$ (A $*$ y)
(4) Prediction network	200×150	*fc*(512, ReLU) *fc*(512, ReLU) *fc*(150, softmax) = prediction

Table C.1: Overview of the FUSE network

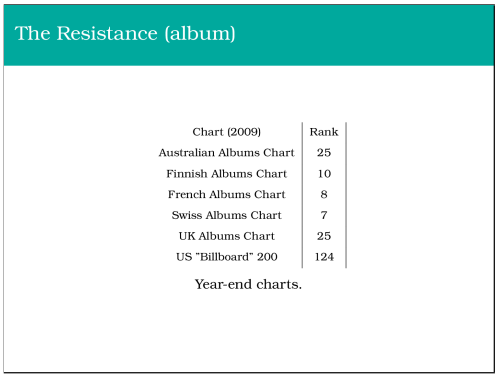
D Examples - SlideQA

In this chapter, we show example pages with accompanying questions from our collected SlideQA dataset. Moreover, we include the predictions of our proposed FUSE model together with our baselines (Q+IMG, SAN, and RN).

Figure D.1 shows questions aiming to analyze the *reading comprehension* of the VQA models. In both examples, our FUSE architecture was able to infer the correct title and caption, respectively. In Figure D.2, we include example *relational questions* where the models need to identify several page components and use these as a reference to localize, count, or recognize other document entities. As we see, our model inferred the correct number of birds in the natural image as well as recognized the table category left of the figure. Figures D.3 and D.4 show pages with associated questions on the global structure and overall appearance of the page. Finally, Figure D.6 and Figure D.5 visualize different variations of color and layout for the same document content.

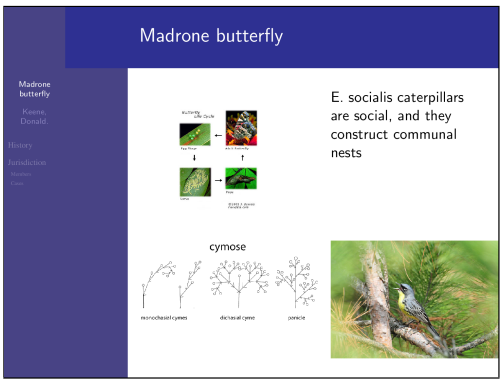


Question: What is the title of the slide?
Q+IMG: 'Melanie C' ✗ SAN: 'Man on the Moon: The End of Day' ✗ RN: 'Man on the Moon: The End of Day' ✗ FUSE: 'Tampa Bay Area' ✓

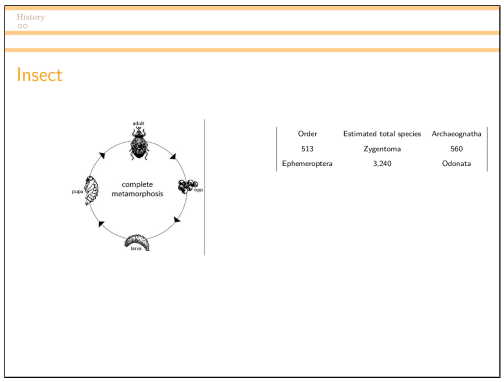


Question: What does the caption of the table entail?
Q+IMG: 'The natural image' ✗ SAN: 'Weekly charts.' ✗ RN: 'Weekly charts.' ✗ FUSE: 'Year-end charts.' ✓

Figure D.1: Example answers of the proposed networks for structural and component-based questions.

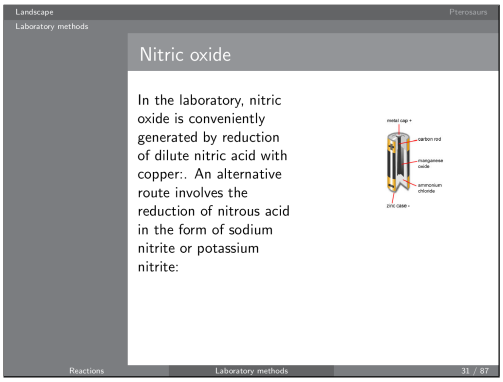


Question: How many birds are pictured in the bottom-right image?
Q+IMG: '2' ✗ SAN: '2' ✗ RN: '1' ✓
FUSE: '1' ✓

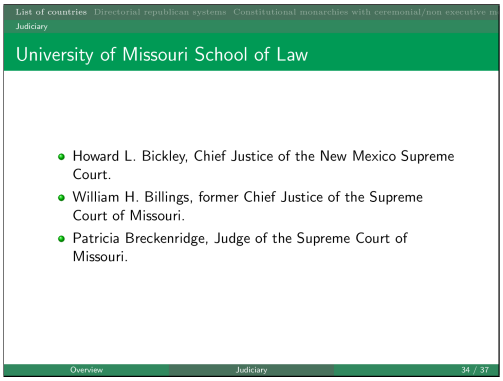


Question: What component type is to the right of the figure?
Q+IMG: 'enumeration' ✗ SAN: 'enumeration' ✗ RN: 'table' ✓ FUSE: 'table' ✓

Figure D.2: Example of predictions for relational questions.

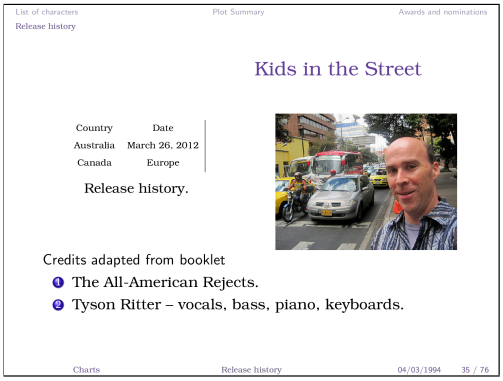


Question: What page number does the slide have?
Q+IMG: '72' ✗ SAN: '4' ✗ RN: '3' ✗
FUSE: '31' ✓

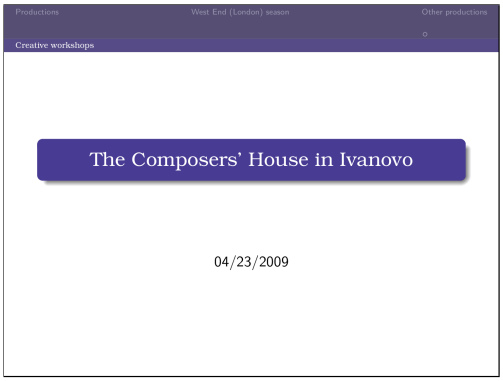


Question: What color does the layout of the slide have?
Q+IMG: 'green' ✓ SAN: 'blue' ✗
RN: 'green' ✓FUSE: 'green' ✓

Figure D.3: Example answers of the proposed networks for structural and component-based questions.



Question: How many components does the page have?
Q+IMG: '1' ✗ SAN: '2' ✗
RN: '2' ✗ FUSE: '3' ✓



Question: What slide type is this page?
Q+IMG: 'Content-slide' ✗ SAN: 'Content-slide' ✗ RN: 'Title-slide' ✓
FUSE: 'Title-slide' ✓

Figure D.4: Example answers of the proposed networks for structural and component-based questions.

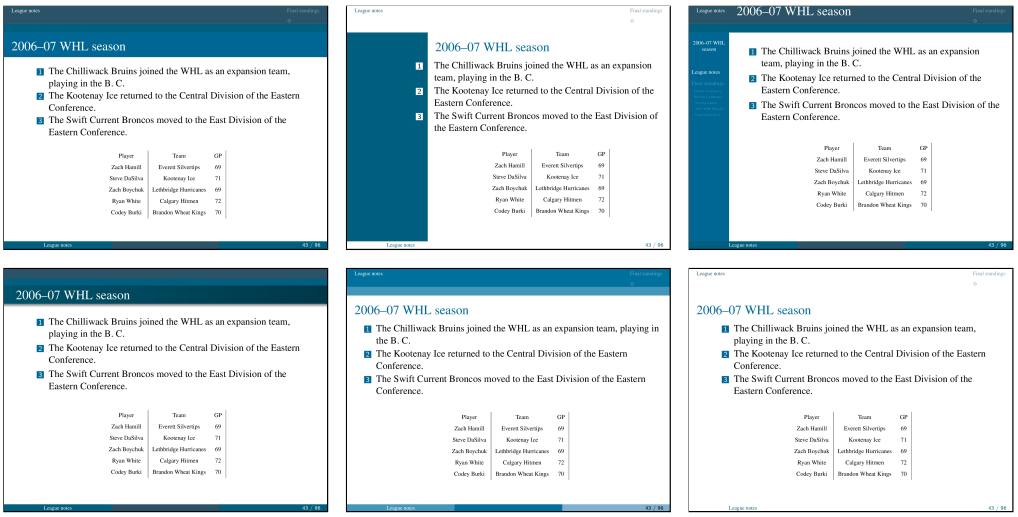


Figure D.5: Example of different layouts for the same document content (enumeration and table) and identical color (midnight blue).

Figure D.6: Example of different colors for the same document content (two enumerations, a table, and a natural image) and identical layout.

E Node Matching

E.1 Qualitative results

COCO-Images and COCO-Captions [#caps=1]

Annotation: {horse, horse, person, person}

Caption 1: "Two equestrians riding their horses on the beach together"

NAP: {horse, horse, person, person}

Caption 2: "Two people riding horses along the ocean beach"

NAP: {horse, horse, person, person}

Img (NAP): {horse, horse, person, person}

Annotation: {sheep, sheep, sheep, sheep, sheep}

Caption 1: "Pack of five sheep sleeping on a snowy day'

NAP: {sheep, sheep, sheep, sheep, sheep}

Caption 2: "Sheep are laying down together in the snow"

NAP: {sheep, sheep, sheep}

Img (NAP): {sheep, sheep, sheep, sheep, sheep}

COCO-Captions [#caps=5] and COCO-Images+COCO-Captions [#caps=5]

Caption 1: "A man and woman sitting on a stone bench under umbrellas"
Caption 2: "Man and woman with umbrellas seated on a stone bench"
Caption 3: "Two people sitting on a rock while holding onto umbrellas"
Caption 4: "A beautiful woman sitting next to a man"
Caption 5: "Two people sitting on a stone bench holding umbrellas"
NAP [#caps=5]: {`person`, `umbrella`, `person`, `chair`, `handbag`, `umbrella`}
NAP [#caps=5]+Img: {`handbag`, `person`, `person`, `bench`, `umbrella`, `umbrella`}
GT: {`handbag`, `person`, `person`, `bench`, `umbrella`, `umbrella`}

Caption 1: "Two little girls holding up chocolate and vanilla donuts"
Caption 2: "Two young girls peeking through the holes in their donuts"
Caption 3: "Two little girls looking the through the holes in doughnuts"
Caption 4: "Two young girls holding doughnuts over their eyes"
Caption 5: "Two children hold up doughnuts to their eyeballs"
NAP [#caps=5]: {`person`, `donut`, `person`, `donut`}
NAP [#caps=5]+Img: {`person`, `person`, `chair`, `donut`, `donut`}
GT:{`person`, `person`, `chair`, `donut`, `donut`}

VRD

GT Nodes: {wheel, car, bus, wheel, sky, street}
GT Triplets: {(wheel, on, bus); (car, behind, bus); (car, under, sky); (bus, has, wheel); (bus, near, car); (bus, above, wheel); (bus, under, sky); (bus, on, street); (wheel, under, bus); (sky, above, bus)}

NAP Nodes: {wheel, building, bus, person, sky, street, car, wheel}
NAP Triplets: {(sky, above, wheel); (sky, above, street); (wheel, on, street); (bus, on, street); (bus, has, wheel); (sky, above, bus); (street, under, sky); (street, under, bus); (wheel, under, sky) }

Visual Genome

GT Nodes: {racket, man, shirt, face, man, head, hand, hair, shirt}
GT Triplets: {(man, wear, shirt); (man, have, face); (man, have, hair); (man, wear, shirt); (man, have, head); (man, wear, shirt); (hand, hold, racket)}

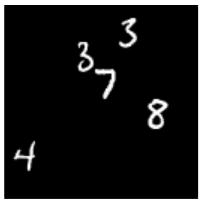
NAP Nodes: {racket, man, shirt, short, man, head, hand, hair, shirt, face}
NAP Triplets: {(man, wear, shirt); (man, have, hair); (head, on, shirt); (shirt, on, man); (man, hold, racket); (hair, on, shirt); (hair, on, racket); (hair, on, hand); (man, hold, racket); (man, have, hand)}

GT Nodes: {grass, giraffe, tail, branch, giraffe, tree, head, dirt, giraffe, leg}
GT Triplets: {(giraffe, have, head); (giraffe, have, leg); (tree, have, branch); (dirt, on-top-of, grass); (giraffe, have, tail)}

NAP Nodes: {grass, giraffe, tail, giraffe, tree, head, neck, giraffe, leg, branch, dirt}
NAP Triplets: {(giraffe, have, leg); (tree, behind, giraffe); (giraffe, have, tail); (giraffe, have, head); (giraffe, have, giraffe); (tree, behind, giraffe); (head, of, giraffe); (head, on, grass); (giraffe, have, tree); (tail, on, leg)}

MNIST-Graphs[1,2]

	1	9	6	9	1
1	–	←↓	←↓	→↑	→↓
9	→↑	–	←↑	→↑	→↑
6	→↑	→↓	–	→↑	→↓
9	←↓	←↓	←↓	–	→↓
1	←↑	←↓	←↑	←↑	–

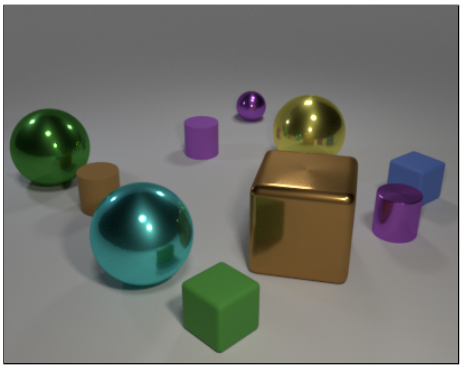


	4	7	3	3	8
4	–	←↓	←↓	←↓	←↓
7	→↑	–	←↓	→↓	←↑
3	→↑	→↑	–	→↑	←↑
3	→↑	←↑	←↓	–	←↑
8	→↑	→↓	→↓	→↓	–

	4	7	6	9	9
4	–	→↑	←↑	←↑	→↑
7	←↓	–	←↓	←↓	←↓
6	→↓	→↑	–	←↓	→↓
9	→↓	→↑	→↑	–	→↑
9	←↓	→↑	←↑	←↓	–

	4	7	8	9	9
4	–	←↑	→↑	←↑	→↑
7	→↓	–	→↑	→↓	→↑
8	←↓	←↓	–	←↓	→↓
9	→↓	←↑	→↑	–	→↑
9	←↓	←↑	←↑	←↓	–

CLEVR[3,4]



	○	▯	□	○	○	▯	□	○	□	▯
○	–	←↓	←↑	←↑	←↓	←↑	←↑	←↓	←↑	←↑
▯	→↑	–	←↑	→↑	←↑	←↑	←↑	←↓	←↑	→↑
□	→↓	→↓	–	→↓	←↓	←↓	←↓	←↓	←↓	→↓
○	→↓	←↓	←↑	–	←↓	←↓	←↓	←↓	←↓	→↓
○	→↑	→↓	→↑	→↑	–	←↑	→↑	→↓	←↑	→↑
▯	→↓	→↓	→↑	→↑	→↓	–	→↑	→↓	←↓	→↓
□	→↓	→↓	→↑	→↑	←↓	←↓	–	→↓	←↓	→↓
○	→↑	→↑	→↑	→↑	←↑	←↑	←↑	–	←↑	→↑
□	→↓	→↓	→↑	→↑	→↓	→↑	→↑	→↓	–	→↑
▯	→↓	←↓	←↑	←↑	←↓	←↑	←↑	←↓	←↓	–

[1] We represent with ←, →, ↑, and ↓ the `left`, `right`, `top`, and `bottom` relationship between the digits
[2] We mark with red errors in the prediction
[3] ○, ▯, and □ stand for a `sphere`, `cylinder`, and `cube` shape, respectively
[4] We denote with ←, →, ↑, and ↓ the `left`, `right`, `behind`, and `in front` relation between the shapes

In the previous pages, we visualize the nodes and graphs inferred by our approach for several datasets: COCO-Images, COCO-Captions (with a single caption), COCO-Captions (with five captions), COCO-Images+COCO-Captions (with images and the five captions), VRD, Visual Genome, MNIST-Graphs, and CLEVR. For MNIST-Graphs, we mark with red the incorrect predictions.

E.2 Parameter and learning setting

Data splits. We split all our data into training, validation, and testing in an identical manner as related work. In the case that no available labels are provided for testing (CLEVR and COCO), we use the provided validation set for testing. Thus, we show the performance of the baselines, the networks requiring additional supervision, and NAP on the validation set provided in the datasets. We extract our own data samples for validation from the official training set.

Learning procedure. While our approach can be combined with any conventional backpropagation algorithms, we train all our methods using Adam [Kingma and Ba, 2015] as an **optimization scheme** with an initial **learning rate** of 0.00025 and the exponential decay rate for the first moment of 0.9 and the second moment of 0.999. In case of the node set estimation, we empirically choose a lower learning rate of 0.0001. Note that we have not evaluated other optimizers in combination with our networks and we analyze learning rates in the range 0.00001 and 0.001. In the loss, we set the node weight to 1 and for the edges to 0.5. Finally, we use early stopping based on the sum of the accuracy over the nodes and edges.

Graph2Graph. We generate random graphs comprising c semantic classes associated to the nodes, which are connected through binary edges. Since we can generate a large amount of different graphs, the models are not prone to overfitting. We represent each sample as a set of nodes encoded using a one-hot representation, while the edges are depicted by a binary matrix. Moreover, we set the number of

nodes fixed throughout our evaluation to five. Since for each model we have fixed graph sizes, we do not have any `empty` nodes and, thus, $\bar{n}$ is equal to n (*i.e.*, we do not require a supergraph expansion).

In the first step of our approach, we flatten the entire input graph sample $d \in \mathcal{D}$ into a global vector representation γ. In case of Graph2Graph, we leverage graph convolutions, which we apply multiple times to refine the data point d. Thereby, we choose the number of graph convolution layers based on the longest path in the input graphs, *i.e.*, we use $n - 1$ layers. While using less layers would still achieve a reasonable accuracy, in most cases, we would not achieve a near perfect performance as each node would need information from all other nodes in the graph. All the graph convolutions leverage a ReLU activation function and have 4096 hidden units. Thus, the last graph convolution layer outputs a matrix of the form $n \times 4096$, which we flatten over the nodes by averaging, we obtain a 4096-dimensional vector. Then, in our experiments, we decrease this vector into a smaller global encoding γ ranging from only 3 to 30 numbers using a fully-connected layer with ReLU activation and $3 - 30$ units (*i.e.*, comprising the bottleneck).

Then, we inflate the global vector γ to our graph using our previously described NAP scheme. We use the global vector γ and map it to the node-specific features ϕ_i through a fully-connected layer with ReLU activation using 4096 units each. The probability distribution ν_i for each node i over the classes $\mathcal{C}$ is obtained by passing ϕ_i through a fully-connected layer followed by softmax normalization. In a similar manner, we compute the edge embeddings $\varepsilon_{i,j}$ using a fully-connected over the relation classes, which in our case are binary, *i.e.*, we use a fully-connected with softmax normalization over two classes (for consistency, we use two units and softmax).

COCO. COCO is a popular large scale dataset with a high number of annotations for instance segmentation and image captioning. The annotation process comprised several rounds of check ups, where incorrect labels were discarded and resulted in very clean instance segmentations and image captions. We leverage the images, instance labels, and captions to analyze our model for generating nodes from visual and textual data. Thereby, the instance-wise annotations of the 91 object classes

will represent the ground truth node set V^*. Since for each image the number of nodes is variable, we expand V^* with `empty` nodes resulting to $\bar{\boldsymbol{v}}^*$ with ten nodes per supergraph. Note that we do not have any edges between these nodes and, thus, we do not use any edge information for the loss or matching procedure.

For the visual representation, we leverage the penultimate layer after the global average pooling of a ResNet-50 [He et al., 2016] pre-trained on ImageNet. The CNN embeddings are represented by a 2048-dimensional vector, which we re-encode using two fully-connected layers of 1024 units with ReLU activation generating our global vector γ. Note that we process the images in the same way as required for the ResNet-50 model.

In case of the COCO-Captions variant, we represent each word separately using one-hot encodings. We lower-case all words and remove all punctuations from the sentence. Thereby, we restrict the vocabulary to comprise only words occurring at least 100 times in our training set, resulting in a vocabulary size and one-hot encoding dimension of 1926. To enable batch training, we necessitate that *all* captions have an equal length and, therefore, we set all sentences to a common size of 20 words through padding or by discarding excess words at the end of the sequence.

While we fuse the captions by averaging their representations to obtain a single vector embodying all five captions, we combine the two vectors representing the visual and textual content, respectively, via concatenation.

We inflate γ to the nodes by using a fully-connected layer to the probability distribution of the 91 semantic classes plus the `empty` category. Thereby, we set the maximum number of possible nodes to 15 as most of the images fall into this set (86%) and discard all graphs that have more nodes. In this setup, since we only address the task of node recognition, we set the edge weights to $\lambda^E = 0$ and $\ell^E = 0$ and train the network as mentioned in Section E.2.

VRD. While COCO comprises over 123k images, in case of VRD, we only have 5k data points from which 1k we leave for testing as defined in the standard split. Due to the small amount of data in VRD, we restrict our graphs to have at most

ten nodes associated to 100 semantic categories and 70 edge classes. We have not included any additional pre-processing than required by the neural networks that we considered for embedding the images.

We have analyzed several pre-trained networks (ResNet, Inception, and VGG) for extracting a visual representation and leverage the architecture that obtained the highest accuracy on the validation set, *i.e.*, Inception-v3 [Szegedy et al., 2016]. We obtain a 2048-dimensional embedding, which we re-encode with a fully-connected layer with ReLU activation representing our 1024-dimensional global vector γ.

We then inflate the global vector γ into the node-specific embeddings ϕ_i through a fully-connected layer with 128 units for each node index i. We map each vector ϕ_i to the node distributions ν which have the shape 10×101 (as we have 100 categories plus the `empty` class). In a similar manner, we extract the edge distributions ε that comprise 70 classes. We update the parameters in an end-to-end manner.

Visual Genome. To keep the analysis consistent with the model for VRD, we use an identical setup for both the backbone and data pre-processing. We also keep the same supergraph sizes, while the main difference lies in the number of node and edge categories. In case of Visual Genome, we predict 100 node classes plus the `empty` class, and 50 relation types.

We re-embed the 2048-dimensional visual vector by employing two fully-connected layers with ReLU activation of sizes 1024 each resulting in a global vector γ of size 1024. Then, we inflate the 1024-dimensional vector γ through a fully-connected layer resulting in 128-dimensional vectors ϕ_i for each node i. We map each vector ϕ_i to the node distributions ν (comprising 101 categories) and edge distributions ε (with 50 classes).

MNIST-Graphs. We generate MNIST-Graphs ourselves by leveraging the MNIST dataset comprising grayscale images associated to ten handwritten digit categories. First, for each data sample, we create a blank 128×128 image and, then, we include a digit at a time by replacing the black pixels of the input image with the randomly selected digits totaling to five per graph. Thereby, we make sure that the digits do

not highly-overlap and make sure that only digits from the training set are included in our training images. We create in total 210k such images, where the same digit can occur more than once. We generate the labels by directly setting the class of the selected digits as the nodes, while the relations (`left`, `right`, `in front`, `behind`) are computed based on the center of the digits in the 128×128 image.

As input to our network, we use the 128×128 (grayscale) image, which we re-encode using a CNN, which we train with the rest of the network via the NAP scheme in an end-to-end manner. The number of node classes is equal to ten, while we have four relation types. Note that the supergraph $\bar{\mathcal{G}}$ is equal to the graph $\mathcal{G}$ as we have a constant number of nodes for each image (thus, the `empty` nodes are not required in this setup).

We encode the image using a CNN with five blocks comprising two convolution layers with ReLU activation and 3×3 kernels each. Therein, the first layer has a stride of one keeping the input dimensionality, while the second one employs a stride of two. Each block contains a different number of units for the convolutions: 32, 32, 64, 64, and 128. Finally, we use a final convolution layer with 128 units and with a stride of one, and obtain our global vector γ by first averaging the resulting tensor and, then, using a second fully-connected layer with 1024 units.

From the 1024-dimensional vector γ, we inflate the node-specific features ϕ_i via a fully-connected layer with 256 units. We then get the distributions for the nodes through another linear mapping with 151 units followed by softmax and the same for the edges, where we use a fully-connected layer with 50 units.

CLEVR. We resize all images in CLEVR to a common size of $120 \times 160 \times 3$ and generate the graphs directly from the annotations provided in the dataset. Since we do not have any labels in the test set for comparing the quality of the graphs, we leverage the validation set provided in the dataset for testing. We extract from the training set 10k graphs for validation.

We embed the images using a CNN which is trained together with the graph generation module. The CNN comprises five blocks of three convolution layers each with ReLU activation and 3×3 kernels. The first two layers comprise a stride

of two, while the final one in each block contains a stride of two for decreasing the dimensionality. All the layers in one block have an identical number of units: 128, 256, 256, 256, and 512, respectively. Then, we use another convolution layer with a size of 512, ReLU activation, and a stride of one. The global vector is obtained through average pooling followed by another fully-connected layer with 1024 units as for MNIST-Graphs. Using the global vector γ, we generate the node-specific features ϕ_i with a fully-connected layer with 256 units. We then map them on the three node classes and four edge categories. Since the number of nodes per graph varies for each image, we leverage `empty` nodes (*i.e.*, the supergraphs are not necessarily equal to the subgraphs). Finally, we train the backbone CNN together with the graph generation module end-to-end using only the graphs as supervision.

F Deutsche Zusammenfassung

Seit dem Aufkommen von Deep Learning wurden außerordentliche Fortschritte in Bereichen erzielt, in denen es um die semantische Analyse und des Verstehens von Daten geht. Durch diese Entwicklung wurde die Kluft zwischen den Fähigkeiten von Menschen und der KI deutlich verringert. Im Zentrum der semantischen Analyse und des Verstehens von Daten steht die Aufgabe des *Question Answering*, die darauf abzielt, Systeme zu implementieren, welche Fragen auf der Grundlage einer gegebenen Wissensbasis beantworten. Das Beantworten von Fragen hat eine Vielzahl von Anwendungen, die von der automatischen Indizierung und Anwendungsfällen in Systemen für assistive Technologien bis hin zur Grundlagenforschung zu menschenähnlicher KI reichen. Aufgrund ihrer Bedeutung in derart weitreichenden Anwendungsbereichen, hat das Beantworten von Fragen in den letzten Jahren in der Forschung enorme Aufmerksamkeit bekommen, was zu beträchtlichen Fortschritten sowohl in der Struktur solcher Systeme als auch in den zugehörigen Datensätzen führte.

Beliebte Ansätze zur Beantwortung von Fragen umfassen neuronale Netze, die die textuellen Daten (Fragen) und die eingegebenen Informationen (z.B. Bilder) in den gleichen Repräsentationsraum einbetten und dann auf der Grundlage ihrer

gemeinsamen Kodierung eine Antwort generieren. Die meisten dieser Verfahren verarbeiten jedoch nur einen einzigen Datentyp (z.B. natürliche Bilder) und haben Schwierigkeiten bei der Darstellung hochstrukturierter Daten (z.B. Beziehungen von Objekten in der Szene). Darüber hinaus haben die vorgeschlagenen Modelle eine Black-Box-Struktur, da man ihre Argumentation zur Generierung der Antwort nicht direkt aus dem Netzwerk ableiten kann.

In dieser Arbeit gehen wir auf diese Unzulänglichkeiten ein und schlagen menschlich interpretierbare Ansätze für die Beantwortung von Fragen vor, bei denen die Argumentation der Netzwerke direkt aus ihrer Architektur abgeleitet werden kann. Darüber hinaus sind unsere Ansätze in der Lage, hoch strukturierte Daten aus einer Wissensbasis, die sowohl textuelle als auch visuelle Daten umfassen kann, zu verstehen und aus ihnen zu schlussfolgern. Wir streben Netzwerke an, die in der Lage sind, über mehrere Modalitätstypen und -strukturen hinweg zu verallgemeinern, und die Schlussfolgerungen auf der Grundlage von längeren Argumentationsketten ziehen können. Lernmaterialien sind perfekte Kandidaten für die Analyse solcher Systeme, da sie reichhaltige und vielfältige Inhalte, verschiedene Figurentypen und verschiedene Interaktionsmöglichkeiten bieten. Darüber hinaus hat die visuelle Argumentation auf Lernmaterialien weitere wichtige Anwendungen, z.B. für die automatische Beantwortung von Fragen sehbehinderter Studenten zum Vorlesungsinhalt sowie für die schnelle Indexierung von Dokumentdaten.

Um den Netzwerken das Verstehen von strukturierten Inhalten zu ermöglichen, nutzen wir eine graph basierte Darstellung des Wissensinhalts für unsere neuronalen Netze. Im Falle von Bildern stellen die Knoten Objekte in der Szene dar, während die Kanten ihre Beziehungen repräsentieren. Im Vergleich dazu sind die Knoten bei Textinhalten Satzeinbettungen, bei denen die Kanten ihre Platzierung modellieren. Wir stützen unsere Methoden auf Graph-Neuronalen Netzen, die von Natur aus mit hochstrukturierten Eingabedaten umgehen und eine natürliche Darstellung der Wissensbasis liefern können. In dieser Arbeit konzentrieren und

untersuchen wir zwei Arten von Graphnetzwerken: einen beziehungsorientierten Ansatz, der auf Kantenpooling basiert, und eine pfadbasierte Methode, die den visuellen Eingabegraphen anhand der Fragestellung durchsucht.
Schließlich demonstrieren wir die Argumentationsfähigkeiten unserer Modelle in einer umfangreichen experimentellen Analyse. Dabei zeigen wir, dass unsere Modelle in den Bereichen Fragenbeantwortung, multimodales Argumentieren und Beantwortung von Seitenfragen Ergebnisse auf dem neuesten Stand der Technik erzielen. Wir evaluieren darüber hinaus verschiedene Konfigurationen unserer Architekturen, Eingabemodalitäten und der Datenrepräsentation (z.B. rohe visuelle 3D-Tensoren oder grafische Darstellungen). Unsere Analyse zeigt, dass unsere Modelle in der Lage sind, Transparenz und menschliche Interpretierbarkeit zu bieten, d.h. es ist möglich, direkt Rückschlüsse über das bearbeitete Wissen und die Argumentation zur Beantwortung jeder Frage zu ziehen.