Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Visual Question Answering on Scientific Charts Using Fine-Tuned Vision-Language Models

Florian Schleid

Field of Study: Computer Science
Matriculation No.: 7306796
1st Examiner: Prof. Dr. Chris Biemann, Universität Hamburg
2nd Examiner: Dr. Martin Semmann, Universität Hamburg

Faculty of Mathematics, Informatics and Natural Sciences

Universität Hamburg
Hamburg, Germany

A thesis submitted for the degree of

Master of Science (M. Sc.)

Supervisor:
Jan Strich, Universität Hamburg

Visual Question Answering on Scientific Charts Using Fine-Tuned Vision-Language Models

Masters's Thesis submitted by: Florian Schleid

Date of Submission: 01.11.2025

Supervisor:
Jan Strich, Universität Hamburg

Committee:
1st Examiner: Prof. Dr. Chris Biemann, Universität Hamburg
2nd Examiner: Dr. Martin Semmann, Universität Hamburg

Universität Hamburg, Hamburg, Germany
Faculty of Mathematics, Informatics and Natural Sciences

Abstract

Scientific charts often encapsulate the core findings of research papers, making the ability to answer questions about these charts highly valuable. This is especially true for readers with vision impairments or in scenarios where a large amount of papers need to be scanned for their research results, e.g., in meta-analyses. This thesis explores the task of visual question answering (VQA) on scientific charts using large Vision-Language Models (VLMs) and newly developed datasets. In the context of the SciVQA shared task from the 5th Workshop on Scholarly Document Processing, we design and evaluate multimodal systems capable of answering diverse question types, such as multiple-choice, yes/no, unanswerable, and open-ended questions, based on chart images extracted from scientific literature. We investigate the effects of zero-shot and one-shot prompting, as well as supervised fine-tuning (SFT), on the performance of Qwen2.5-VL models (7B and 32B variants). We also experimented with including more training data from domain-specific datasets (SpiQA and ArXivQA) and manually improved the training data provided by the shared task. Our experiments show that fine-tuning leads to great performance increases and underline that the prompt has a significant influence on the results. However, we also found that a one-shot example could not improve the generated answers. Furthermore, the fine-tuned models were sensitive to the domain of the training data, as including additional data from similar datasets did not lead to better results. Our fine-tuned Qwen2.5-VL 32B model achieves a substantial improvement over the GPT-4o mini baseline and reaches 4th place in the shared task, showing the effectiveness of domain-specific fine-tuning. We published the code for the experiments[1].

1. https://github.com/Flo0620/SciVQA_Repo

Contents

List of Figures

List of Tables

1 Introduction

Figures are often the first thing that readers of scientific papers look at (Rolandi et al., 2011). In addition, they frequently communicate the main results of a paper in a condensed form[1]. Therefore, figures can be a valuable resource for automatic information extraction systems. In recent years, the number of papers published has grown rapidly (Hanson et al., 2024). Manually reading increasing amounts of research papers to keep updated on the current state of the art (SOTA) does not scale and, therefore, motivates automated systems to harvest the results of scientific papers. This is especially important for meta-analyses, which are common in medical and psychological research and combine the results of different studies that are focused on the same disease or phenomenon to determine the effectiveness of a treatment. Such studies are labor-intensive and require domain experts to extract data from hundreds of research articles. The ability to automatically identify and extract the relevant data from a scientific paper and its charts would significantly simplify the creation and updating of such meta-analyses (Mutinda et al., 2022; Marshall et al., 2020).

A further issue in the present landscape of scientific publishing is that only a small fraction of papers include meaningful alt-text for their charts. As a result, visually impaired readers are often unable to access the information these charts convey (Kumar and Wang, 2024; Crane et al., 2023). Advances in Vision-Language Models (VLMs) for visual question answering (VQA) on scientific charts offer a promising way to improve the accessibility of research papers for blind and low-vision audiences (Yan et al., 2025).

However, automatically interpreting charts poses challenges due to their detailed visual components and the complex spatial arrangements of elements. The process requires spatial reasoning and numerical understanding (Meng et al., 2024). In the domain of scientific papers, automatic VQA is further complicated due to domain-specific terminology and unique chart types (Huang et al., 2025). Furthermore, there is a research gap regarding the ability of these models to recognize visual cues (e.g., color or shape) and to connect and integrate them with textual elements such as the legend, axis annotations, or the caption (Borisova et al., 2025).

1. https://www.springer.com/gp/authors-editors/authorandreviewertutorials/writing-a-journal-manuscript/figures-and-tables/10285530

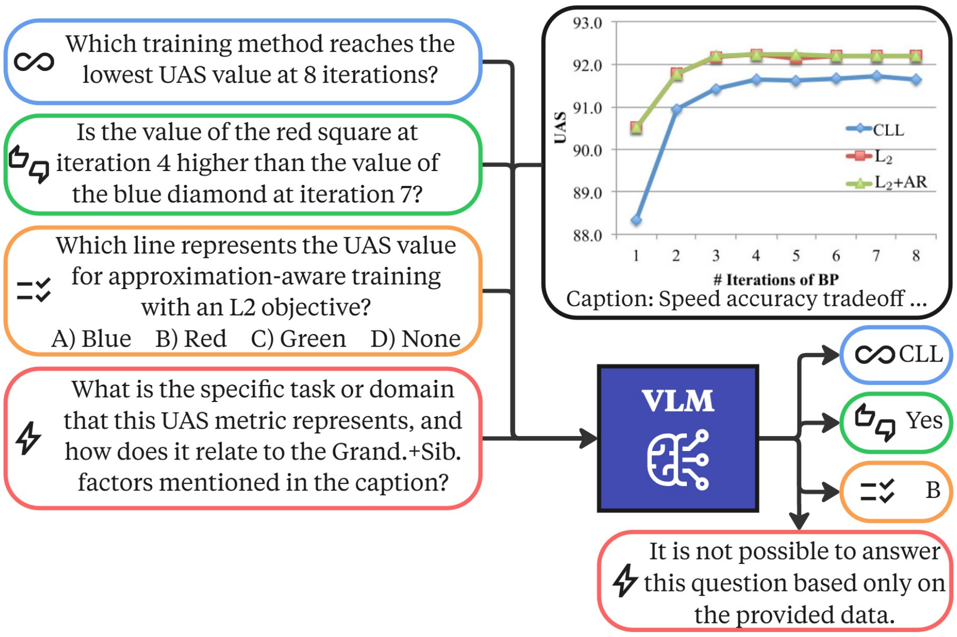

Which training method reaches the lowest UAS value at 8 iterations?
Is the value of the red square at iteration 4 higher than the value of the blue diamond at iteration 7?
Which line represents the UAS value for approximation-aware training with an L2 objective?
A) Blue B) Red C) Green D) None
What is the specific task or domain that this UAS metric represents, and how does it relate to the Grand.+Sib. factors mentioned in the caption?
UAS
# Iterations of BP
CLL
L2
L2+AR
Caption: Speed accuracy tradeoff ...
VLM
CLL
Yes
B
It is not possible to answer this question based only on the provided data.

Figure 1.1: Overview of the system with the four question types: infinite answer set, yes/no, multiple-choice, and unanswerable.

In recent years, the development of Large Language Models (LLMs) like GPT-4 (OpenAI et al., 2024), Gemini 2.5 (Comanici et al., 2025), and Llama 3 (Grattafiori et al., 2024) enabled text-based question answering. The adaptation of LLMs to be able to work on visual input introduced VLMs, which can be trained to perform VQA on charts. New VLMs like Qwen2.5-VL enable state-of-the-art results in the domain of chart VQA (Bai et al., 2025). Furthermore, recent datasets, like SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024), provide large amounts of question-answer pairs for images of scientific charts.

This thesis intends to explore the topic of VQA on scientific charts in the context of the SciVQA shared task (Borisova et al., 2025). It comes with a dataset and challenges participants to answer questions about scientific charts. An exemplary image with four corresponding questions from this dataset can be seen in Figure 1.1. The questions are divided into four different question types, namely infinite answer set questions, yes/no questions, multiple-choice questions where multiple of the four options can be correct, and unanswerable questions. An unanswerable question cannot be answered based only on the information from the image and the caption. Furthermore, the questions can either be visual questions, meaning that they refer to one of the visual attributes shape, size, position, height, direction, or color, or they are non-visual questions.

Such data enables the fine-tuning of VLMs that specifically excel on the task of VQA on scientific charts. This can be done by using Parameter-Efficient Fine-Tuning (PEFT), which refers to a spectrum of approaches that reduce the memory and computational requirements of fine-tuning by only updating a subset of the parameters of the original model (Prottasha et al., 2025). Besides the fine-tuning, the formulation of the prompt is very important. It combines the question and a task description, together with additional information like the caption or metadata about the figure type, into a natural language query. This prompt, along with the image of the chart, is given to the VLM as input. Even slight changes in the formulation can lead to substantial differences in performance

(Zhou et al., 2022). Furthermore, the prompt can contain a demonstration of the task with the correct solution, a so-called one-shot example. Brown et al. (2020) found that such demonstrations can have a positive influence on the results. Therefore, prompt engineering, the process of finding a suitable prompt template, is an important step to increase the performance of the model in the SciVQA task (Zhou et al., 2022).

1.1 Research Questions

The objective of this thesis is to perform as well as possible on the SciVQA shared task, and therefore, to improve the abilities of open source VLMs on VQA on scientific charts. For this task, the open-source model Qwen2.5-VL (Bai et al., 2025) is used in its variants with 7 billion and 32 billion parameters. Qwen2.5-VL reaches SOTA performance on chart VQA benchmarks like ChartQA (Masry et al., 2022) and CharXiv (Wang, Xia, et al., 2024; Bai et al., 2025) and is therefore suited for the proposed challenge. As a first measure to improve the performance, prompt engineering is employed to create a prompt template specifically designed for the SciVQA task. Furthermore, the influence of a one-shot example is tested. For additional performance gains, the SciVQA dataset is used to fine-tune the Qwen2.5-VL base models, leveraging Low Rank Adaptation (LoRA) and quantization for resource-effective fine-tuning. Hyperparameter tuning determines the best-performing fine-tuning setup. In an attempt to reach even higher performance, the training data was expanded using the SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024) datasets and by manually improving the SciVQA data.

This setup should solve the following research questions:

- **RQ1**: What is a good prompt template for VQA on scientific charts in the context of the SciVQA shared task, and can a one-shot example improve the performance?
- **RQ2**: How does fine-tuning a VLM impact the performance of VQA on scientific charts, and what hyperparameter configurations yield the best results?
- **RQ3**: Does fine-tuning on a combination of the SciVQA dataset with similar datasets, or on a manually improved version of the SciVQA dataset, enhance the answer quality of the model?

1.2 Structure of the thesis

This section gives an overview of the structure and the contents of this thesis. In the Introduction in Chapter 1, the problem of VQA is motivated, and the research questions are formulated. Chapter 2 covers the theoretical foundations needed in this thesis, starting with a short introduction into neural networks for sequence modeling tasks in Section 2.1. It continues in Section 2.2 with the historic development that led to LLMs and covers the adaptation of LLMs to VLMs in Section 2.3. This section also discusses the fine-tuning and prompt engineering for VLMs. Chapter 3 provides related work on the task of VQA on scientific charts, including a detailed explanation of the SciVQA dataset (Section 3.1), and the baselines of the SciVQA shared task (Section 3.2). Furthermore, other related datasets and models from the domain of VQA on charts are introduced in Section 3.3 and Section 3.4. The Qwen2.5-VL model architecture and its pretraining are presented in Section 3.5, and an overview of the models proposed by the other participants of the SciVQA competition is given in Section 3.6. The explicit experiments that were conducted are then explained in Chapter 4. These experiments are zero-shot prompting in Section 4.1 and one-shot prompting in Section 4.2. After that, the fine-tuning of Qwen2.5-VL models only on the SciVQA dataset is described in Section 4.3. Expanding the SciVQA dataset with data from the ArXivQA and SpiQA datasets is detailed in Subsection 4.4.1. Subsequently, the fine-tuning on this expanded dataset is explained in Subsection 4.4.2. Improving the SciVQA data and fine-tuning on it is the topic of Section 4.5, and finally, the experiment on the influence of fine-tuning different modules of the model is discussed in Section 4.6. These experiments are then evaluated in Chapter 5 in Section 5.2 after the evaluation metrics are explained in Section 5.1. Section 5.3 then contains ablation studies with a cross-validation between the SciVQA, ArXivQA, and SpiQA datasets, as well as an experiment on the influence of the training dataset size. It also provides an error analysis per figure type. In the conclusion in Chapter 6, the findings for each research question are then summarized, and the limitations and possible future work for each research question are provided.

2
Fundamentals

In this chapter, we discuss the theoretical foundations and developments that led to the emergence of VLMs. Within Section 2.1, we begin with an introduction to neural networks, starting with the classic Multi-Layer Perceptron (MLP) in Subsection 2.1.1. Then Recurrent Neural Networks (RNNs) (Subsection 2.1.2) and Long Short-Term Memory Networks (LSTMs) (Subsection 2.1.3) are explained as methods to tackle sequence modeling tasks. In Section 2.2, a brief history of language models is presented, and Transformers, which laid the foundation for LLMs, are introduced in Subsection 2.2.5. The subsequent Section 2.3 explains the adaptation of LLMs to process visual inputs as a second modality and details the pretraining and fine-tuning of VLMs.

2.1 Neural Networks

This section starts by introducing MLPs as a classical neural network architecture (Subsection 2.1.1). It continues by explaining how RNNs (Subsection 2.1.2) and LSTMs (Subsection 2.1.3) enable predictions for time-dependent sequence data.

2.1.1 Multi-Layer Perceptron (MLP)

MLPs are the most basic neural networks (Zhang, 2023). Figure 2.1 shows the structure of an MLP that is divided into multiple layers of perceptrons. The perceptron (also called a neuron) was first introduced by Rosenblatt (1958), and it works by computing a weighted sum of the input signals. A bias is added to this sum, and the result is the input into a non-linear activation function, such as the sigmoid function. The resulting value is the output of the perceptron (Alom et al., 2018). In an MLP, the first layer, called the input layer, receives the input signal. Each input node is connected to each perceptron in the second layer, which is a hidden layer. Through these connections, the input signals are forwarded to each perceptron in the hidden layer, and the computed outputs of each perceptron represent the output of the hidden layer. This output is then again passed to all neurons in the next layer. In the last layer, the so-called output layer, the result is the output signal of the MLP (Haykin, 2009). To train the network, an

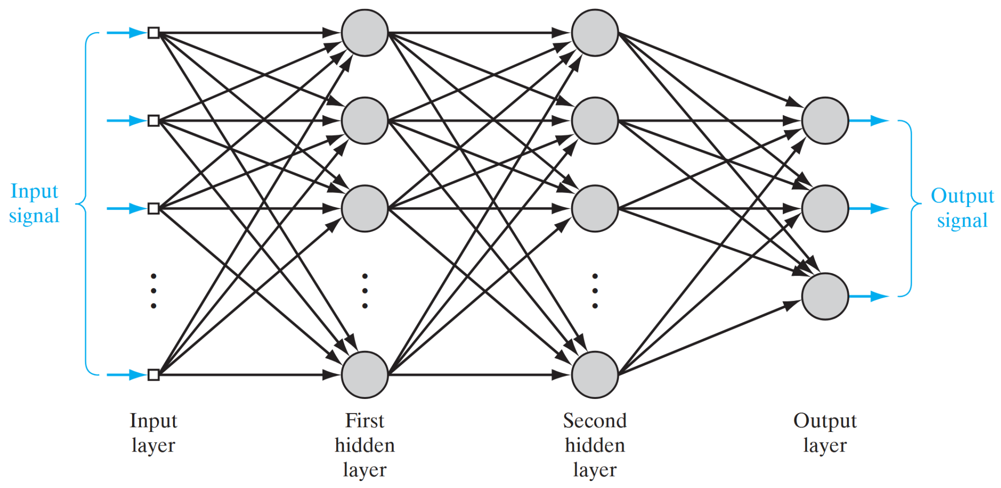


Figure 2.1: Schematic of an MLP. Extracted from Haykin (2009).

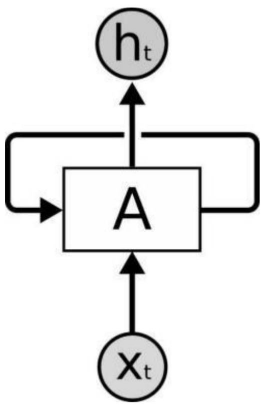


Figure 2.2: Schematic of an RNN. Extracted from Zhang (2023).

error signal is computed at the output layer and recursively back-propagated through the network. In the backpropagation process, the weights and biases are updated to reduce the error of the network (Haykin, 2009).

2.1.2 Recurrent Neural Networks (RNNs)

Traditional neural networks like MLPs cannot directly deal with time-dependent sequence data such as natural language text, since they treat each input and output independently. This is solved by RNNs, which were first introduced in 1986 by Rumelhart et al. (1986). Figure 2.2 depicts the basic structure of an RNN, which is similar to that of an MLP with an input layer X_t, a hidden layer A, and an output layer h_t. Opposed to MLPs, RNNs process sequence data by including information from predictions of previous time steps. This is done by adding the output of the hidden layer A from the previous time step to the input of the hidden layer A at the current time step. This is done through a recurrent connection as depicted in Figure 2.2. Through this reuse of the hidden layer state, RNNs have the ability to deal with the time dependence of sequence data (Zhang, 2023).

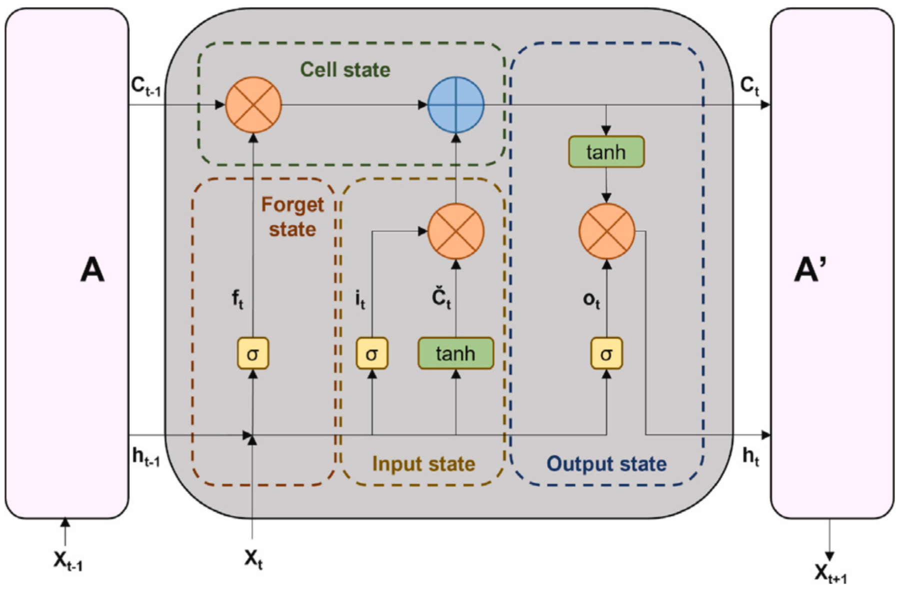


Figure 2.3: Schematic of an LSTM. Extracted from Picornell Rodríguez et al. (2023).

2.1.3 Long Short-Term Memory Networks (LSTMs)

A problem of RNNs is that during training, vanishing and exploding gradients can occur, which restricts their capacity to model long-term dependencies (Zhang, 2023). Hochreiter and Schmidhuber (1997) proposed LSTMs to tackle this problem by introducing trainable gates that control the flow of information across the different steps of processing a sequence. The structure of an LSTM cell is shown in Figure 2.3. The cell gets the input of the current time step X_t, the hidden state from the previous time step h_{t-1}, and the cell state of the previous time step C_{t-1}. The forget gate decides how much of the previous state is forgotten. The input gate controls which information from the current input is added to the cell state. Lastly, the output gate controls the creation of the hidden state h_t for the current time step, based on the current input X_t, the previous hidden state h_{t-1}, and the current cell state C_t that was created by the forget and input gates and the previous cell state. The new hidden state h_t is then handed over to the next time step and can also be used to compute the prediction for the current time step (Picornell Rodríguez et al., 2023). Each of the gates learns how to selectively forget and include information from previous states, to model long-term dependencies in sequences. This enables LSTMs to perform well in sequence modeling tasks (Zhang, 2023).

2.2 Language Models

This section gives an overview of the architectures and techniques that led to the Transformer architecture and subsequently to LLMs (Subsection 2.2.6), covering early statistical models (Subsection 2.2.1), the Encoder-Decoder architecture that builds upon the previously introduced RNNs and LSTMs (Subsection 2.2.3), the attention mechanism (Subsection 2.2.4), and an introduction into the functionality of Transformers (Subsection 2.2.5).

2.2.1 The n-Gram Model

First language models in the late 1980s often used the n-gram model (Rosenfeld, 2000), which predicts the next word based on the previous $n - 1$ words. The probability of the n-th word given the $n - 1$ last words is calculated by collecting a large corpus of text and dividing the number of occurrences where the $n - 1$ words are followed by the n-th word by the number of occurrences of the sequence of the $n - 1$ last words. However, the corpus needs to be very large to support larger numbers of n since otherwise there is a high likelihood that many possible n-grams are not even contained in the training corpus. This is due to the exponentially growing number of possible n-grams for increasing n, where n is the exponent and the vocabulary size is the base. Usually n is 2 or 3 (Wang, Chu, et al., 2024; Rosenfeld, 2000). A specific example for a model based on the n-gram model is the back-off model by Katz (1987), which expands the n-gram model by "backing off" to use the $(n - 1)$-gram model if the underlying training corpus does not contain the n-gram to create a prediction for the next word.

2.2.2 Word2Vec

To be able to use neural networks in natural language tasks, a way to translate human language into a numeric representation, while preserving the semantic relationships in the language, is needed (Wang, Chu, et al., 2024). The idea of projecting words into a vector representation is already quite old and was, for example, performed in 2000 by Bengio et al. (2000). However, in 2013, the release of Word2Vec (Mikolov et al., 2013) made it possible to efficiently convert words into vector representations for huge vocabulary corpora. These vector representations capture syntactic and semantic word relationships. This was achieved by introducing the Continuous Bag-of-Words (CBOW) and Skip-gram models. The CBOW model is a neural network that is trained to predict a target word in a sentence by its surrounding words. The Skip-gram model does the opposite by giving a target word as input and predicting the surrounding words of the target word. Both models have only a single hidden layer, and then the weights of the hidden layer are the embedding of the target word. The Skip-gram model works better on semantic tasks and the CBOW model slightly better on syntactic tasks, while also being faster to train than the Skip-gram model (Mikolov et al., 2013). The vectors resulting from Word2Vec can be used to perform mathematical operations on the words and, therefore, enable neural networks to achieve better performance in natural language processing tasks. Words that have a semantic relationship, such as a country and its capital, are mapped closer to each other in the vector space. This even enables vector calculations on the embedding vectors, e.g. $\text{vec}(\text{"Madrid"}) - \text{vec}(\text{"Spain"}) + \text{vec}(\text{"France"})$ is close to vec("Paris") (Wang, Chu, et al., 2024; Mikolov et al., 2013).

2.2.3 Encoder-Decoder Architecture

Sutskever et al. (2014) attracted significant attention by using an LSTM in the Encoder-Decoder architecture for machine translation. The architecture solved the problem that an RNN or LSTM returns one output for each input element in the sequence. In many tasks, an input sequence of a given length needs to be mapped to an output sequence of a different length. Such tasks can be machine translation or speech recognition, but also question answering, where the given question and the returned answer are word sequences of different lengths. Such sequence-to-sequence tasks can be solved using the Encoder-Decoder architecture (Sutskever et al., 2014).

The architecture consists of two main components: an Encoder that processes an input sequence over K time steps and condenses it into a fixed-length context vector, and a Decoder that unfolds over L time steps to generate the output sequence based on the context vector that represents the input sequence. In this setup, L can be any number of steps and therefore supports any sequence-to-sequence task. In a basic setup, only the final context vector generated by the Encoder is provided to the Decoder in the first time step. For the remaining steps, the Decoder uses its internal state. Although effective, this approach faced limitations in retaining information from long input sequences, as the Decoder relied solely on the final Encoder state (Tsirmpas et al., 2024).

To address this, Bahdanau et al. (2015) proposed an extension that allows the Decoder to automatically search the input for elements that are relevant to predict an output at the current time step. This is done by using the Encoder to generate an annotation h_i for each element of the input sequence that contains information about the whole input sequence with a strong focus on the parts around the i-th element. A feedforward network is trained to predict how well the state of the Decoder at a specific time step t matches with each of the generated annotations h_i. For an input sequence of length K, this results in K matching scores $e_{tj}, \quad j = 1, \dots, K$ for a time step t. After applying softmax on these matching scores, the weights $\alpha_{tj}, \quad j = 1, \dots, K$ are obtained. These are then used to compute a weighted sum of the annotation vectors:

$$c_i = \sum_{j=1}^{K} \alpha_{ij} h_j \tag{2.1}$$

c_i now contains information from the input sequence that is specifically relevant to the current state of the Decoder and is used by the Decoder to generate the next output. This mechanism improves the ability of the model to handle long sequences. (Bahdanau et al., 2015). This mechanism is also known as additive attention, which is one of the most commonly used attention mechanisms. It is also very similar to the attention function used in the Transformer originally proposed by Vaswani et al. (2017). Opposed to using a feedforward network to compute the matching of the Decoder's state with the annotations of the input elements, Vaswani et al. (2017) use the dot product as a compatibility function.

2.2.4 Attention in Transformers

Attention is the key mechanism in the Transformer architecture, which was introduced by Vaswani et al. (2017). It is used to replace recurrent connections while maintaining the ability to relate information from every part of the sequence to compute a representation of the whole sequence. The attention function is the mapping of a query and a set of key-value pairs to an output. One specific attention layer in the Transformer architecture allows the decoder of a Transformer to attend to every part of the input sequence. In that case, the query is the state of the decoder, and the keys and values come from the encoder output. This would be an Encoder-Decoder attention layer. There are also self-attention layers where all keys, values, and queries originate from one place. If no optimizations or simplifications are applied to the attention function, the output, as well as the query, the keys, and the values, are vectors. The query and the keys are

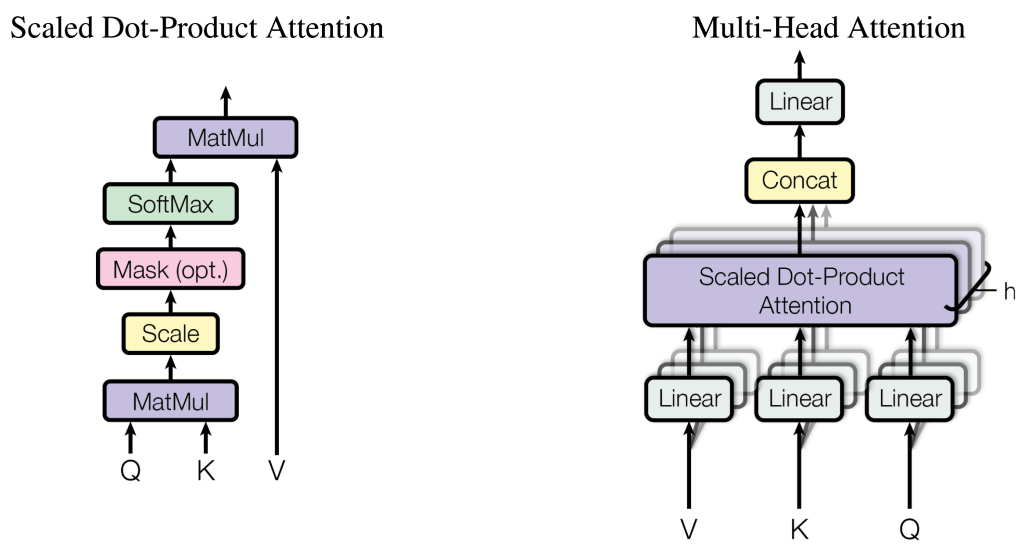


Figure 2.4: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention that consists of multiple attention layers in parallel. Extracted from Vaswani et al. (2017).

used in a compatibility function to get one weight for each value vector. These weights are used to compute the weighted sum of the value vectors, which is the output of the attention function for that query (Vaswani et al., 2017). Although there are multiple implementations to compute attention, in the original proposal of the Transformer architecture by Vaswani et al. (2017), the Scaled Dot-Product Attention was used. An overview can be seen on the left of Figure 2.4, and this is the mathematical equation:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

Since in practice the attention values for multiple queries are computed at once, instead of a single query vector, multiple queries are condensed into a matrix Q. The key and value vectors are also condensed in the matrices K and V. As visible in Figure 2.4 and Equation 2.2, the first step is to compute the dot products of the queries with all keys through a matrix multiplication. The matrix is then divided element-wise by $\sqrt{d_k}$ where d_k is the dimension of the keys. After this scaling, the softmax function is applied to obtain the final weights. These weights are then multiplied by the value vectors V through another matrix multiplication that leads to the output. The division by $\sqrt{d_k}$ is done to counteract the effect that for larger values of d_k the dot product grows larger, which pushes the softmax function into regions where it has very small gradients (Vaswani et al., 2017).

The right side of Figure 2.4 shows the Multi-Head Attention Mechanism. Instead of using one large dimension for the query, key, and value vectors, they are linearly projected h times with different learned parameters to obtain representations with h times lower dimensionality. On each of the projected sets of query, keys, and values, the Scaled Dot-Product Attention is then computed in parallel. The h outputs are then concatenated and again linearly projected, resulting in the final output. This process enables the model to attend to information from different representation subspaces at different positions while having a similar computational complexity as single-head attention with full dimensionality (Vaswani et al., 2017).

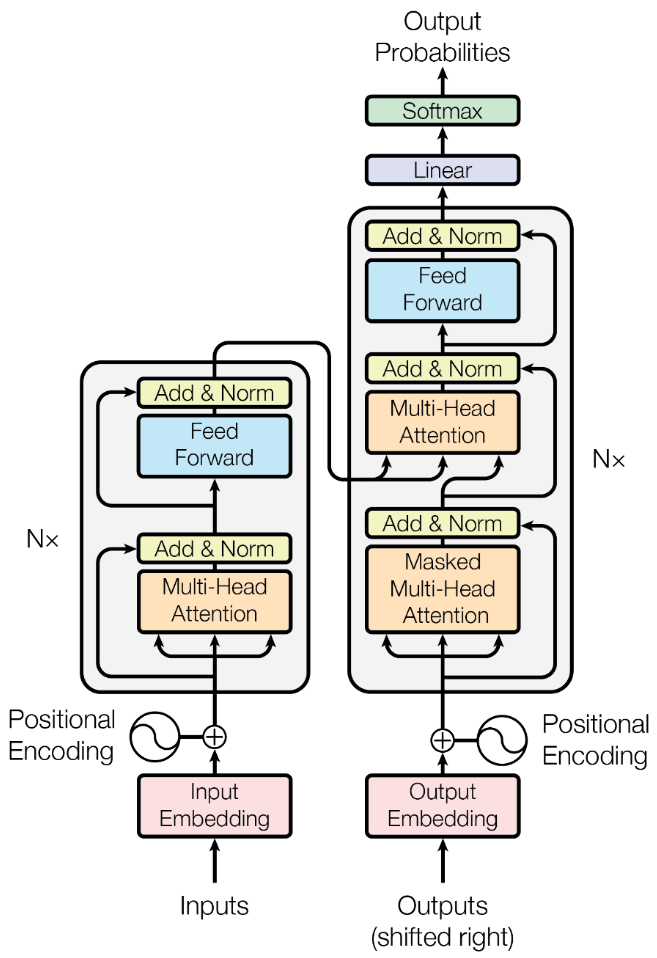


Figure 2.5: The Transformer architecture. Extracted from Vaswani et al. (2017).

2.2.5 Transformers

The Transformer architecture resulted from the attempt to replace RNNs for sequence-to-sequence tasks while retaining the Encoder-Decoder architecture and adding self-attention (Tsirmpas et al., 2024). As visible in Figure 2.5, in Transformers, the Encoder and Decoder consist of N layers. The Decoder is auto-regressive, which means that the output of the last layer is the input of the first Decoder layer in the next timestep. The Encoder processes the entire input sequence at once. Its first layer receives as input a list of K vectors, where each vector corresponds to one token of the input sequence. These vectors are the combination of a representation of the token (for example, a word2vec embedding) and the positional encoding of the token in the input sequence. This positional encoding is a dense vector representation of the index of the token in the input sequence. This is necessary since the information about the order of the sequence would otherwise be lost in the attention function. Therefore, information about the position of each token in the sequence needs to be injected into the embedding of the token. The final output of each layer is the input of the next layer (Tsirmpas et al., 2024; Vaswani et al., 2017).

Each of the N Encoder layers is divided into two sub-layers. Firstly, there is a Multi-Head Self-Attention Mechanism, and secondly, an MLP. In one layer, the MLP is applied to each position of the sequence with the same weights. However, across the layers, each layer has its own set of parameters for the MLP. After each sub-layer, a residual connection (He et al., 2016) adds the input of the sub-layer to the output of the sub-layer, and normalization is applied (Vaswani et al., 2017). This is a mechanism to stabilize the training of deep networks and to prevent vanishing or exploding gradients (Lin et al., 2022). To facilitate this application of residual connections, all sub-layers, as well as the embedding process, produce outputs of the same dimension (Vaswani et al., 2017).

The Decoder takes the previous outputs, which are again embedded and combined with a positional encoding, as the input into the first sub-layer (Vaswani et al., 2017). To create an autoregressive model that can generate an output one step at a time, the output of the decoder should only depend on tokens generated up to the current time step (Zhang and Shafiq, 2024). Subsequently, in the training setting, the ground-truth output tokens are shifted by one position to the right, and the first sub-layer uses Masked Multi-Head Attention instead of normal Multi-Head Attention. Masking prevents a position i from attending to a position greater than i, making sure that only already generated tokens are attended to. Opposed to the Encoder, in the Decoder, a second Multi-Head Attention sub-layer is added after the first attention sub-layer. This is an Encoder-Decoder attention sublayer and receives the output of the Encoder as the keys and values, and the output of the Masked Multi-Head Attention sub-layer as the query. Therefore, every position in the Decoder can attend to the entire input sequence. The last sub-layer is again an MLP, and also the residual connections around each sub-layer, together with the layer normalization, are the same as in the Encoder.

The output of the last Decoder layer is fed into a linear transformation with learned weights, and finally, the softmax function is applied to the output of the transformation to obtain the final output probabilities for the next token (Vaswani et al., 2017).

2.2.6 Large Language Models (LLMs)

LLMs constitute a family of deep learning models that excel in understanding and generating natural language. They usually contain billions of parameters and are typically trained on huge amounts of training data. The foundation of LLMs is the Transformer architecture. Depending on the task that should be solved by the model, either only the Decoder, the Encoder, or both are retained (Shao et al., 2024).

As illustrated in Figure 2.6, various LLM models with diverse model architectures were recently developed. Encoder-only models, such as BERT (Devlin et al., 2019), ERNIE (Zhang et al., 2019), and ALBERT (Lan et al., 2020), appear less prominently in recent years but remain fundamental for contextual understanding tasks. Decoder-only models, shown throughout the timeline, include OpenAI's GPT series (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020; OpenAI et al., 2024), Meta's open-source LLaMA models (Touvron, Lavril, et al., 2023; Touvron, Martin, et al., 2023; Grattafiori et al., 2024) as well as the recent Qwen2.5 model (Yang et al., 2025) (Shao et al., 2024). Such decoder-only models rely solely on a Decoder with its auto-regressive character to generate one token based on the previous tokens and thrive in generation tasks. Encoder–Decoder architectures integrate both components, leveraging the language understanding of the Encoder and the generation capabilities of the Decoder. Such models are typically used for conditional generation tasks like summarization or machine translation. Examples for an Encoder-Decoder architecture shown in Figure 2.6 would be T5 (Raffel et al., 2019) and its successors mT5 (Xue et al., 2021) and FLAN-T5 (Chung et al., 2024).

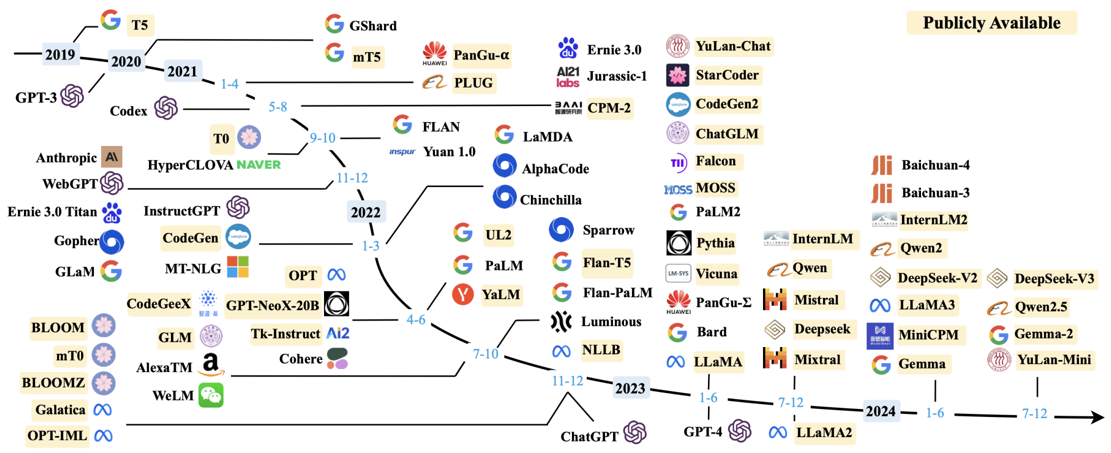


Figure 2.6: A timeline of LLMs with over 10 billion parameters developed in recent years. The timeline is primarily based on the release date of the technical paper for a model, such as its submission date to arXiv. For models without an associated paper, the date corresponds to the earliest known public release or announcement. Extracted from Zhao et al. (2023).

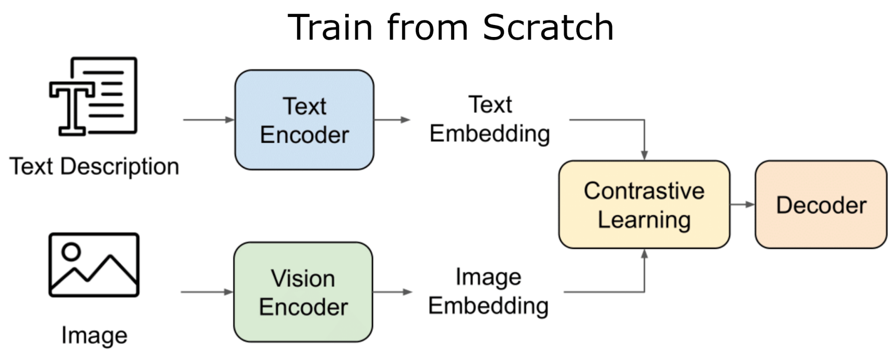
Modern LLMs are trained on huge amounts of text data and exhibit strong capabilities to understand language and to solve complex tasks through text generation. With sufficiently large parameter counts, LLMs can gain abilities that are not present in smaller models. Such abilities include in-context learning, where an LLM is provided with a natural language instruction and possibly some example solutions of the task, and is able to solve the task without extra training. Instruction following is another valuable ability of LLMs, where the LLM can follow a task description of a new task without explicit examples. This is enabled by performing instruction tuning, which means fine-tuning the LLM on multi-task datasets with natural language task descriptions (Zhao et al., 2023).

Figure 2.6 highlights the competitive and collaborative nature of LLM development. Major technology companies, like Google, Microsoft, Meta, and others, have been active across the entire timeline, alongside contributions from research institutions and open-source communities. This collaborative push has enabled a wide range of applications. LLMs can, for instance, be used for Information Retrieval, where the traditional search engines are challenged by chatbots like ChatGPT. Furthermore, Microsoft 365 includes Copilot[1] to automate office work, and GitHub Copilot can support Software Developers to write code[2] (Zhao et al., 2023). LLMs are also applied in the original task explored by Vaswani et al. (2017), namely machine translation. Google, for instance, uses their LLM PaLM 2 (Anil et al., 2023) to add 110 new languages to Google Translate[3].

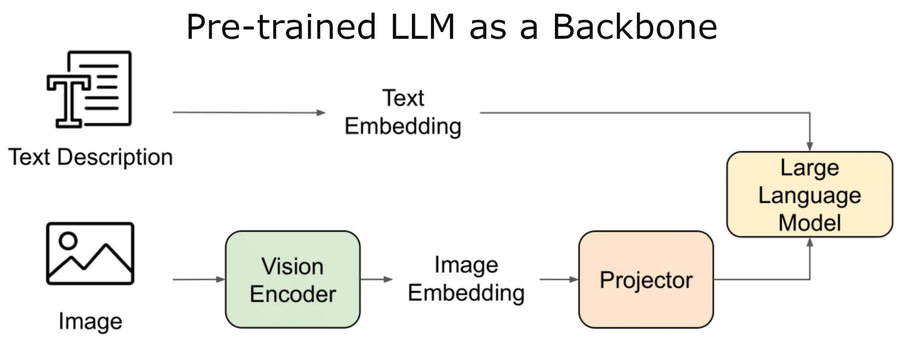
2.3 Vision-Language Models (VLMs)

To expand the capabilities of LLMs even further, VLMs were developed. The first VLMs were adaptations of the BERT model (Devlin et al., 2019), like ViLBERT (Lu et al., 2019). VLMs can process visual inputs such as images and even videos along with classic text prompts. Subsection 2.3.1 explains their general architecture. After

1. https://adoption.microsoft.com/en-us/copilot/
2. https://github.com/features/copilot
3. https://blog.google/products/translate/google-translate-new-languages-2024/



(a) VLM architecture variant that trains the whole model from scratch.



(b) VLM architecture variant that uses a pretrained LLM as a base.

Figure 2.7: Two common variants of VLM architectures. Based on Li et al. (2025).

that, Subsection 2.3.2 covers the pretraining of a VLM which enables the model to capture vision-language knowledge and to perform on tasks without specific fine-tuning (Zhang, Huang, et al., 2024). Subsection 2.3.3 discusses prompt engineering as well as zero-shot and few-shot prompting that enables improved performance on new tasks through task-specific prompts. Lastly, Subsection 2.3.4 explains fine-tuning, where the parameters of a VLM are updated or new parameters are added to specialize the model for a new task or domain (Prottasha et al., 2025).

2.3.1 VLM Architecture

In order to extract information from images and to be able to work with them, a Vision Encoder is needed. The idea is to project visual components into embeddings that align with the embeddings from LLMs. This is done by training the Vision Encoder on image-text pairs so that it captures visual and language relationships effectively. The Vision Encoders can be Convolutional Neural Networks such as ResNet (He et al., 2016) or Vision Transformers (ViTs) (Dosovitskiy et al., 2021). Through the pretraining on large amounts of labeled visual data or through Self-Supervised training, they can capture domain-specific visual knowledge from their training domains (Li et al., 2025).

Among others, there are two popular variants for building a VLM as shown in Figure 2.7. One option is to train a completely new model from scratch. The second option, which is used in many recent VLMs, uses a pretrained LLM as a base for the VLM, since it already has powerful language understanding and makes a dedicated text encoder obsolete. Also, the original text decoder of the LLM can often be used to generate the text output. In the first variant, which is depicted in Figure 2.7a, in addition to the Vision Encoder, a Text Encoder is needed, which embeds the text input sequence into an embedding space. Then, the text embeddings and visual embeddings are aligned in a shared latent space. This can be done, for instance, by contrastive learning (Oord et al., 2019), which captures cross-modal relationships by minimizing the distance between the visual and text embeddings in the shared space for related image-text pairs and maximizing the distance for unrelated pairs. The resulting embeddings are then passed to a Decoder (Li et al., 2025).

The second variant, as visible in Figure 2.7b, often does not include a dedicated Text Encoder and instead relies on LLMs with their original Text Decoder for the text understanding. In comparison to the first variant, this is more efficient, since the pretrained LLM already contains rich world knowledge, which does not need to be learned again (Yin et al., 2024). The visual information is incorporated through projection layers or cross-attention mechanisms. The projector maps the visual features of the Vision Encoder into compact embedding tokens that are aligned with the text embeddings of the LLM. This is normally done by an MLP that can be trained jointly with the rest of the model, including the LLM, to optimize cross-modal objectives. Instead of complete end-to-end training, other parts of the model, such as the LLM or the Vision Encoder, can also be frozen to retain pretrained knowledge (Li et al., 2025).

To improve the performance on the downstream tasks, as well as the safety and reliability of the VLM, an alignment algorithm can be applied to the pretrained VLM. The goal of such algorithms is to align the generated answers of the VLM closer to the preferences and values of humans. The general approach is to use Reinforcement Learning from Human Feedback (RLHF) where a reward model is designed and applied on the VLM with collected human feedback. This technique has shown great success for LLMs and has also become one of the most popular and effective methods for aligning VLMs, although the second visual modality increases complexity (Li et al., 2025).

2.3.2 Pretraining of VLMs

The goal of pretraining a VLM is that the model learns image-text correlation and is able to perform predictions on visual recognition tasks in a zero-shot setting. There exist different pretraining objectives which broadly fall into three categories: contrastive objectives, generative objectives, and alignment objectives (Zhang, Huang, et al., 2024).

Contrastive objectives train the VLM to embed image and text features of the same sample closely together in the feature space, while maximizing the distance between image and text features of different samples. This also works only for the image modality, where the distance of augmentations of one image should be minimized and the distance from the other samples maximized (Zhang, Huang, et al., 2024).

Generative objectives learn the semantics of images and text by training the model to generate missing text or image data. It is done by masking out parts of texts or images and training the model to reconstruct the missing parts (He et al., 2022; Devlin et al., 2019). This can be done with only the text, only the image, or in a cross-modal fashion where parts of the text, as well as of the image, are masked and then reconstructed by the model (Zhang, Huang, et al., 2024).

Alignment objectives introduce a score function that measures the alignment probability between the text embedding and the image embedding. This function is used in the loss function of the training, such that the model is optimized to generate embeddings that result in the score function being high for embeddings of an image-text pair, and low for embeddings of unpaired images and texts. This can also be done only for regions of the image and parts of the text to model local cross-modal correlations in image-text pairs. Compared to the contrastive objective, the score function of the alignment objective only models whether two embeddings match or not. It is therefore much easier to optimize for than to optimize for the contrastive objective. However, the alignment objective learns little correlation information solely within the vision or language modality. Subsequently, it is often used as an auxiliary loss in addition to other pre-training objectives (Zhang, Huang, et al., 2024).

2.3.3 Prompt Engineering

As mentioned in Subsection 2.2.6, pretrained LLMs with a sufficient amount of parameters gain the ability of in-context learning. This means that instead of fine-tuning an LLM for a new task, a natural language instruction can be enough for the model to perform reasonably well on the new task. Brown et al. (2020) trained GPT-3 with 175 billion parameters and studied the in-context learning abilities of the model. Besides providing a natural language description of the task without any examples, which is called zero-shot prompting, they also add some demonstrations of the task to the prompt. This addition of examples is called few-shot prompting. The number of examples is limited by the size of the context window of the model. When only one example is provided, this setting is referred to as one-shot prompting. The advantage of such prompting strategies is that no task-specific training data or just a small amount is required. Furthermore, the model does not need to be fine-tuned on the specific task, which would be computationally expensive (Brown et al., 2020). During their evaluation Brown et al. (2020) found that, especially in the few-shot setting, models without fine-tuning can reach performance levels of models specifically fine-tuned for the task. They also found that the performance improved steadily with a larger model size. Furthermore, the one-shot and few-shot settings outperformed the zero-shot setting.

Radford et al. (2021) expanded the concept of achieving in-context learning through the pretraining of large-scale models from LLMs to VLMs. Their model CLIP achieved competitive results with a fully supervised trained baseline in a zero-shot setting across different datasets, even outperforming the baseline on 16 of the 27 tested datasets. Radford et al. (2021) also found that the formulation of the textual prompt affects the performance. The adaptation of the prompt to the specific task is called prompt engineering and represents a tedious process since even slight changes to the wording of the prompt can result in large performance differences. Furthermore, even extensive tuning of the prompt does not guarantee that it is optimal for the task at hand (Zhou et al., 2022).

2.3.4 Fine-Tuning of VLMs

Besides prompt engineering, another possibility to improve the results on a specific task is to fine-tune a VLM on a task-specific dataset. This can be done with Parameter-Efficient Fine-Tuning (PEFT), which refers to a broad class of techniques that reconsider the traditional fine-tuning paradigm by updating only a small subset of model parameters or by incorporating lightweight task-specific modules. These techniques significantly reduce memory and computation requirements. There are five categories: Additive

Fine-tuning, Selective Fine-tuning, Reparameterized PEFT, Hybrid Approaches, and Mixture-of-Experts-based PEFT (Prottasha et al., 2025).

Additive Fine-tuning adds new, trainable parameters or modules to the existing architecture. These modules add task-specific information to the pretrained model. The original parameters are frozen, and thereby the general-purpose knowledge of the pretrained model is preserved.

Selective Fine-tuning updates only a subset of the original parameters to reduce the computational requirements while retaining task-specific effectiveness. The important parameters that should receive training updates are identified based on their contribution to the task at hand. To determine their contribution, gradient magnitudes, Fisher information, or sensitivity analysis are commonly used (Prottasha et al., 2025).

Reparameterized PEFT uses lower-dimensional representations of the model parameters during the training. During inference, they are mapped back into their original space. This preserves the model's abilities while reducing the resources needed for the fine-tuning. Techniques for the deconstruction of weights into lower-dimensional representations can be low-rank matrix factorization, tensor decomposition, and singular value decomposition. A popular example of Reparameterized PEFT is Low Rank Adaptation (LoRA) (Hu et al., 2022). It freezes an already pretrained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ and represents the weight update matrix $\Delta W \in \mathbb{R}^{d \times k}$ with decomposed matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$, where $r \ll min(d, k)$ is the rank. The updated layer parameters W are then computed by

$$W = W_0 + \alpha BA, \tag{2.3}$$

where α is a scaling factor that determines the strength of the updates. For full fine-tuning, ΔW would be computed based on the gradient of all $d \times k$ parameters, while for LoRA only the $r \times (d + k)$ parameters of the matrices A and B are computed. This drastically reduces the memory and computation requirements for the training (Hu et al., 2022). This makes LoRA and Reparameterized PEFT methods in general especially useful for fine-tuning large-scale pretrained models (Prottasha et al., 2025). However, fine-tuning with LoRA can still be affected by overfitting, against which dropout is applied in the training process (Mao et al., 2024).

The concept of LoRA can, in principle, be applied to any weight matrix. For the Transformer architecture, the original LoRA paper (Hu et al., 2022) experimented with applying the concept to the attention layers. Therefore, they tested fine-tuning the weight matrices W_q, W_k, W_v, and W_o, which linearly transform the query, key, value, and output vectors. The MLP layers were frozen. They report that adapting only W_q and W_v, and leaving the remaining weights frozen, leads to the best performance.

Hybrid Approaches use elements from multiple of the named techniques in combination to leverage their strengths. They provide flexibility, adaptability, and robustness for different tasks and can dynamically determine fitting combinations of strategies for the best performance while being efficient. This dynamic adaptation is particularly useful for multi-task problem scenarios (Prottasha et al., 2025).

Lastly, Mixture-of-Experts-based PEFT only uses subsets of the parameters for a given task. A dynamic gating mechanism determines which parts of the model should be active. Therefore, Mixture-of-Experts-based PEFT is beneficial for multi-task problems where it reduces unnecessary computation (Prottasha et al., 2025).

PEFT generally enables the fine-tuning of pretrained models in a resource-constrained environment and preserves the pretrained knowledge of the model. However, all the different methods also increase the complexity of the training process and can be sensitive to newly introduced hyperparameters. This requires extensive experimentation to obtain optimal results. Furthermore, due to the limited number of updated parameters, the PEFT methods may struggle with tasks that require a substantial adaptation of the pretrained model (Prottasha et al., 2025).

2.3.5 Quantization

To reduce the memory requirements during inference and training of an LLM, quantization can be applied. It reduces the precision of the parameters of a matrix multiplication. For example, the 16 or 32-bit weights of a transformer could be mapped to 8-bit or even 4-bit precision and therefore reduce the memory requirement to store these weights. This quantization only leads to minor performance degradations of the LLM (Dettmers et al., 2022).

A popular quantization method for 8-bit quantization is LLM.int8() that was introduced by Dettmers et al. (2022). It mainly uses absmax quantization, which scales a 16-bit input matrix $\mathbf{X}_{f16}$ by $\frac{127}{\|\mathbf{X}_{f16}\|_\infty}$, effectively mapping each input into the 8-bit range $[-127, 127]$. For the multiplication of an input matrix $\mathbf{X}_{f16}$ with a weight matrix $\mathbf{W}_{f16}$, Dettmers et al. (2022) propose vector-wise quantization. It works by applying absmax quantization to each row of $\mathbf{X}_{f16}$ and each column of $\mathbf{W}_{f16}$. This also results in different scaling constants for each row of the input and for each column of the weights. The quantized matrices are denoted by $\mathbf{X}_{i8}$ and $\mathbf{W}_{i8}$. $\mathbf{c}_X$ and $\mathbf{c}_W$ denote the scaling constants for the input and the weights. To dequantize the result matrix, the result of multiplying the i-th row of $\mathbf{X}_{i8}$ with the j-th column of $\mathbf{W}_{i8}$ is multiplied with $\frac{1}{\mathbf{c}_{X_i} \cdot \mathbf{c}_{W_j}}$ where $\mathbf{c}_{X_i}$ is the scaling factor for the i-th row of $\mathbf{X}_{i8}$ and $\mathbf{c}_{W_j}$ the scaling factor for the j-th column of $\mathbf{W}_{i8}$. This is equivalent to multiplying the result matrix element-wise by the inverse of the outer product of $\mathbf{c}_X$ and $\mathbf{c}_W$.

Absmax quantization performs poorly if the quantized parameters contain outliers with a large magnitude, because then a large part of the $[-127, 127]$ range of the 8-bit precision is unused. Dettmers et al. (2022) find that such outliers systematically occur in larger LLMs, but also that only around 0.1% of the parameters are outliers. They propose to exclude them from the quantization process and keep the 16-bit precision for them. Since there are only a few outliers, this does not significantly increase the memory requirements while retaining the performance of the quantization for the remaining parameters. Dettmers et al. (2022) call this mixed-precision decomposition.

3 Related Work

The SciVQA shared task invites participants to develop multimodal systems for VQA on scientific charts (Borisova et al., 2025). To support this, the task provides the SciVQA dataset, which is further explained in Section 3.1. Furthermore, the shared task created two baselines, one by querying a GPT-4.1 mini model and one by distributing the test set to five annotators with the task to answer the questions. These baselines are introduced in more detail in Section 3.2. Other related datasets and benchmarks in addition to the SciVQA dataset are presented in Section 3.3, and related VQA models are discussed in Section 3.4. The specific architecture of Qwen2.5-VL and the training that was performed on it by Bai et al. (2025) is introduced in Section 3.5, as Qwen2.5-VL is the model that is primarily used in this thesis. The approaches of other SciVQA participants are detailed in Section 3.6.

3.1 SciVQA Dataset

The SciVQA shared task provided a dataset consisting of 3000 images and 7 questions per image. We refer to the dataset as the SciVQA dataset[1]. The data originates from two other datasets. The first source is the SciGraphQA dataset (Li and Tajbakhsh, 2023), which contains images of scientific charts from Computer Science and Machine Learning ArXiv papers. The second source is the ACL-Fig dataset that provides scientific charts from the ACL Anthology. These papers all belong to the domain of computational linguistics (Karishma et al., 2023).
The images were annotated using the Gemini 1.5 Flash model, and these question-answer pairs were then manually validated by graduate students with a computational linguistics background[1]. There are four different question types in the dataset. Firstly, there are infinite answer set questions. They are posed in an open manner and can have any answer, most of the time a value that was read off the chart. Secondly, yes/no questions, as indicated by their name, can either be answered with yes or no. Thirdly, multiple-choice questions give four possible answer options. From the four options, one to four can be correct. Lastly, some questions are unanswerable. The gold answer in

1. https://huggingface.co/datasets/katebor/SciVQA

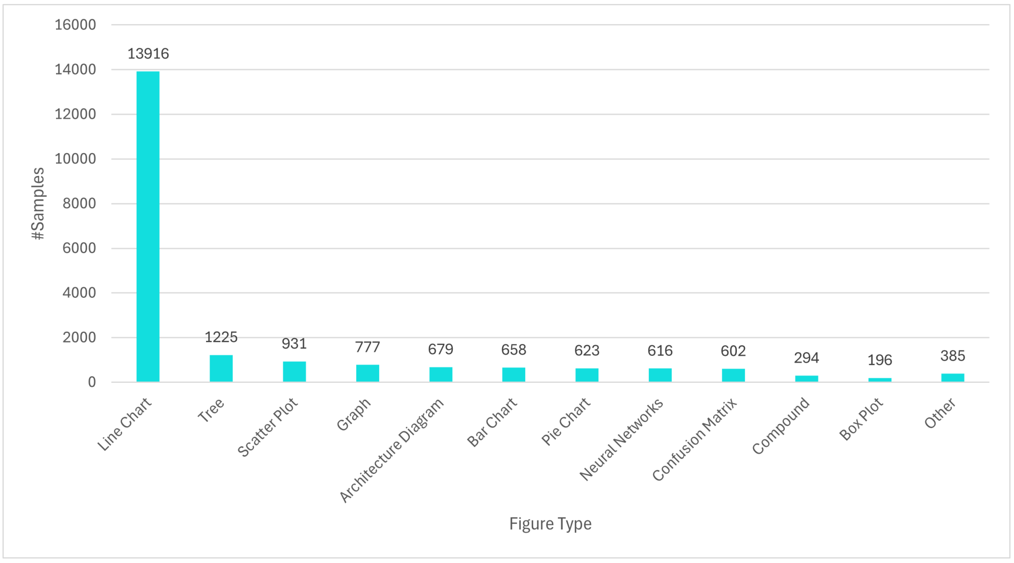


Figure 3.1: The distribution of the different figure types across the whole SciVQA dataset. In total, the SciVQA dataset contains 21K samples.

that case is always "It is not possible to answer this question based only on the provided data." Except for the unanswerable questions, the other question types are again split into two groups. They are either visual or non-visual questions. Visual means that the question references visual elements using their size, height, position, color, shape, or direction. Non-visual questions do not contain any of the six visual aspects. The answers do not contain an explanation and are generally rather short, with an average length of 14.3 characters across the whole dataset. Apart from the images with the questions and answers, the dataset also contains the image captions and further metadata like the figure type and the number of sub-figures in an image [1].

The distribution of figure types is presented in Figure 3.1. With 13.9K samples (67%), by far the largest portion of samples are line charts. Other classical chart types are represented by scatter plots with 931 samples (4%), bar and pie charts with 658 (3%) and 623 (3%) questions, and box plots with 196 (1%) instances. 1225, so 6% of the samples are tree diagrams, 777 (4%) graphs, e.g., of finite state machines, 616 (3%) are neural networks, and confusion matrices contribute 602 (3%) samples. 294 (1%) of the questions correspond to a compound of different figure types. For instance, figures that contain both a line chart and a bar chart. In total, 39.3% of the figures contain multiple sub-figures.

The whole dataset is split into a training set, a validation set, and a test set. The training set contains 15120, the validation set 1680, and the test set 4200 questions. In each of these subsets, the 7 question subtypes are equally distributed, so there are as many unanswerable questions as visual multiple-choice or non-visual infinite answer set questions. The figure types are also roughly equally distributed across the subsets.

3.2 Baselines

To create a technical baseline, the SciVQA task employs few-shot prompting with the GPT-4.1 mini model. They use five examples, which are dynamically picked for each test sample. The examples are chosen to match the question type and the figure type of the current sample. If there are not enough examples where both the question and the figure type match, random examples of the same question type are used. Since giving five unanswerable questions as examples for an unanswerable test sample would reveal the correct answer, for unanswerable questions, only two unanswerable examples are given together with 3 random samples with a different question type (Borisova et al., 2025). However, in a realistic scenario, this setup requires significant prior knowledge about the question at hand. It assumes that one can perfectly distinguish if it is a yes/no or infinite answer set question, and more importantly, can accurately detect the figure type of a given figure. Furthermore, only providing unanswerable questions if the question is unanswerable and otherwise only using examples of the same question type would also require the knowledge of whether the question is unanswerable in advance.

The examples are embedded in a system prompt, which details, depending on the question type, how to answer the question. The user prompt contains, besides the question and the image, the question type and the caption of the image. Prompting GPT-4.1 mini with the few-shot setting led to an average F1-Score of 0.7957 across the ROUGE-1, ROUGE-L, and BERTScore metrics.

For a human baseline, five annotators each receive 840 questions, which they have not previously seen. They are tasked to answer the questions and are instructed to produce concise answers and to use the template response for unanswerable questions. Furthermore, they should state if they do not understand a question, or believe that none of the given answer options is correct, by responding with *"I don't know"*. The human annotators achieve an average F1-Score of 0.8801 (Borisova et al., 2025).

3.3 Other Related Datasets

The SciVQA shared task aligns with the growing research interest in chart-based VQA, where existing benchmarks such as ChartQA (Masry et al., 2022) have seen performance plateaus among large VLMs, largely due to limited data diversity. ChartQA, as well as its successor ChartQAPro (Masry et al., 2025), crawled their chart images from online platforms and therefore do not match perfectly our context of charts from scientific papers (Masry et al., 2025). Nevertheless, ChartQA is one of the most popular benchmarks in chart VQA. It partially contains human-written question-answer pairs on chart images crawled from Statista, Pew Research, Our World In Data, and the OECD. The remaining question-answer pairs are generated by fine-tuned T5 models based on the human-written chart summaries, which were also crawled from the online platforms. The human-annotated questions are split into visual questions that refer to visual attributes and compositional questions that include mathematical or logical operations (Masry et al., 2022). ChartQAPro improved on ChartQA by introducing more diverse charts from more sources and also including questions with different question types, such as unanswerable and multiple-choice questions. The ChartQA dataset largely contains factoid questions based on simple data extraction and simple arithmetic operations. To create the question-answer pairs, humans first annotated some question-answer pairs, and these were then used as examples for the VLMs GPT-4o, Gemini, and Claude, which were prompted to generate five additional question-answer pairs per example. These generated pairs were then manually verified, leading to a total of close to 2000 question-answer pairs.

Besides these datasets with charts from online platforms, SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024) have recently been introduced and contain diverse charts from real-world scientific papers.
SpiQA collects over 25K PDFs from academic research conferences from a variety of computer science domains and extracts figures and tables from them. In total, the SpiQA training dataset contains more than 147K questions on figures. These were generated by Gemini 1.5 pro and then filtered to ensure that each question-answer pair is correct, concise, and can be answered from the image. This training data was used to fine-tune an InstructBLIP-7B model (Dai et al., 2023) and a LLaVA-1.5-7B model (Liu et al., 2024), leading to great improvements compared to their zero-shot performance (Pramanick et al., 2024).
ArXivQA collected papers from ArXiv across various domains such as computer science, mathematics, physics, and economics. To create the ArXivQA dataset, 35K images from these papers were sampled. Then GPT-4V was used to generate multiple-choice questions. This led to 100K question-answer pairs with on average 4.2 answer options per question (Li et al., 2024).

3.4 Chart-Related VQA Models

Recent progress in the field of VLMs includes models for general vision-related tasks, such as Qwen2.5-VL (Bai et al., 2025), a successor to Qwen2-VL (Wang, Bai, et al., 2024). Although Qwen2-VL is not only trained for chart VQA tasks, it has achieved SOTA results in that field (Li et al., 2025; Masry et al., 2025). As Qwen2.5-VL is the model that is mainly used in this thesis, its architecture and training are further detailed in Section 3.5. InternVL3 (Zhu et al., 2025) is another general-purpose VLM and builds upon the pretrained Qwen2.5 LLM. Zhu et al. (2025) report a performance on chart VQA tasks that is competitive to Qwen2.5-VL. Besides these open source models, there are also various closed source VLM models. One of them is Claude 3.5 Sonnet (Anthropic, 2024), which reaches the highest score on both the ChartQA benchmark and its successor, the ChartQAPro benchmark (Masry et al., 2025). Claude 3.5 Sonnet was followed by Gemini 2.0 Flash (Google, 2024) and Gemini 1.5 Flash (Georgiev et al., 2024) with the second and third best performance on the ChartQAPro benchmark. The worst-performing closed-source model tested by Masry et al. (2025) on the ChartQAPro benchmark was GPT-4o (OpenAI et al., 2024).

In parallel, a growing number of models have been developed to specifically address the challenges of chart VQA. ChartLlama (Han et al., 2023), for example, performed fine-tuning on a curated chart dataset and achieved good results on the ChartQA benchmark. ChartAssistant (Meng et al., 2024) and ChartVLM (Xia et al., 2024) fine-tuned models to perform chart-to-table translation. The output of these models is then used as input to specialized models fine-tuned for VQA.

The new datasets presented in Section 3.1 and 3.3 provide additional training data to fine-tune VLMs. Li and Tajbakhsh (2023) found a positive correlation between the size of the training set and the model performance when fine-tuning a LLaVA-13B model (Liu et al., 2023) for chart VQA on up to 296K samples. Furthermore, Wu et al. (2024) showed that the used prompt has a significant influence on the results of the task of VQA on charts, underlining the importance of prompt engineering. They prompted GPT-4o with multiple prompting strategies and found that, for instance, Chain-of-Thought (CoT) prompting (Wei et al., 2022) increased the performance. Masry et al. (2025) also report performance increases through CoT prompting for GPT-4o and other closed source VLMs but find that smaller open source VLMs often perform worse when asked to perform long-form reasoning like CoT. Furthermore, Masry et al. (2025) found that small models that were specifically trained for chart VQA, such as TinyChart-3B (Zhang, Hu, et al., 2024) or ChartInstruct-LLama2-7B (Masry et al., 2024), had the lowest performance on ChartQAPro. Their hypothesis is that these models are overfitted to specific question types and therefore perform poorly on more general questions.

3.5 Qwen2.5-VL Model Description

This section describes the architecture of the Qwen2.5-VL model (Bai et al., 2025) that is used primarily in this thesis in Subsection 3.5.1. The training process of the model is subsequently explained in Subsection 3.5.2.

3.5.1 Architecture of Qwen2.5-VL

The architecture of the Qwen2.5-VL model (Bai et al., 2025) uses an LLM as its foundation, which is initialized with the pretrained weights of the Qwen2.5 LLM (Yang et al., 2025). Its pretraining was performed on a dataset with 18 trillion tokens. As a Vision Encoder, a redesigned ViT architecture is used. A major challenge is the quadratic computational complexity when dealing with images of varying sizes as inputs. To face this challenge, only four layers in the architecture employ full Self-Attention, while the remaining use windowed attention. Therefore, the Vision Encoder resizes an input image to multiples of 28 and then splits the image into 14×14 patches with a stride of 14. In the windowed attention, each position can only attend to a maximum window size of 112×112, which is equal to 8×8 patches. This leads to a linear complexity scaling in the image size instead of a quadratic complexity scaling. Through this architecture, the model can process images at their original input resolution without unnecessary scaling or distortion (Bai et al., 2025).

For positional encoding, 2D Rotary Positional Embedding (RoPE) is used, which encodes relative positional relationships between the image patches. It applies sinusoidal rotations to the query and key vectors during Self-Attention, based on the position of each patch in the image (Heo et al., 2025; Bai et al., 2025).

The projector of the Qwen2.5-VL model takes the patch features created by the ViT and groups spatially adjacent sets of four patch features together and concatenates them. This concatenation is then passed through an MLP with two layers that projects the concatenated features into a dimension that aligns with the text embeddings of the LLM. The outputs of the projector are then fed into the Qwen2.5 LLM Decoder along with the text input. Finally, the Decoder generates the output text (Bai et al., 2025).

3.5.2 Training of Qwen2.5-VL

In this section, the training process of the Qwen2.5-VL model is presented (Bai et al., 2025). It was pretrained in three phases. In the first pretraining phase, only the Vision Encoder was trained from scratch to align it with the LLM and to lay a foundation for multimodal understanding. The training data were image captions, visual knowledge (e.g., celebrity and landmark identification), and Optical Character Recognition (OCR) data.

In the second phase, the whole model was trained on multimodal image data to improve the model's capability to process complex visual information. More reasoning-intensive datasets were used, such as VQA, interleaved data, and multimodal mathematics datasets. They should enhance the model's abilities to establish deeper connections between the visual and the linguistic modality, and therefore improve its performance on more difficult tasks.

In the third pretraining phase, the reasoning capabilities over longer sequences should be strengthened. Therefore, the sequence length of the model was increased, and video and agent-based data were used for the training. This increased the precision of the model on more advanced multimodal tasks that required long-range dependencies and complex reasoning (Bai et al., 2025).

After the pretraining, a post-training process consisting of two phases was performed. During both phases, the weights of the ViT were frozen. First SFT on diverse multimodal data is applied. The data consists of image-text pairs, video, and pure text. After that, in the second phase Direct Preference Optimization (DPO) (Rafailov et al., 2023) aligns the model with human preferences (Bai et al., 2025).

3.6 VQA Models from the SciVQA Task

Besides the models for general VQA on scientific charts, the other participants of the SciVQA shared task provide a number of different approaches to improve the results on the SciVQA dataset. Table 5.14 shows the leaderboard of the SciVQA task with the final results of the model that performs best for each team. Bhat et al. (2025) achieved first place in the competition by including additional context to answer a question. To do that, they used Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) to extract text passages from the original paper that correspond to the chart. In total, three passages were extracted. Firstly, the caption of the chart image, then the text passage that was most similar to the caption, and lastly, the text passage that was most similar to the question. These text passages were then given to a fine-tuned VLM as additional context to answer the question. They also up-scaled the chart images to double their original dimensions, while preserving the edge sharpness to improve the image clarity. With this setup, they could improve the results by 2% compared to a fine-tuned VLM that only had access to the chart and the question. In their final model, they could improve the results by another 0.49% by including 2500 additional human-annotated training samples from the ChartQA dataset (Masry et al., 2022) and by applying post-processing to the generated answers. For answers that provided a range as an answer (e.g., "between 0.6 and 0.7"), the middle of that range was calculated and used as the final answer. Unfortunately, no experiments were conducted to determine whether the performance improvements in the first scenario were due to the image sharpening or the RAG system, or whether, in the second scenario, it resulted from the additional training data or the post-processing.

Resulting in the second best result Ventura et al. (2025) fine-tuned Qwen2.5-VL 7B (Bai et al., 2025) using a CoT prompting strategy. Besides the 7B model variant, they also fine-tuned a Qwen2.5-VL 72B model with the same configuration, but surprisingly, the 7B model outperformed the 72B model.

The third-ranking participants of the SciVQA shared task, Jaumann et al. (2025), used an approach without fine-tuning a VLM. Instead, they used InternVL3-78B (Zhu et al., 2025) and prompted it with a one-shot example. This one-shot example was selected depending on the current question-image pair by filtering the training dataset for samples with the same figure type and extracting the closest matching example. To find the most similar example, BLIP-2 (Li et al., 2023) was used to generate embeddings for the question-image pairs, which could then be compared pair-wise to compute the cosine similarity. For the returned answer, the confidence was calculated based on the predicted probability of the generated answer tokens. If this confidence is above 0.9, the answer is kept. Otherwise, the question is relayed to a different model depending on the question type. For binary questions, the Pixtral-Large-Instruct-2411[2] model is used instead of InternVL3-78B, and two examples are chosen by computing the similarity of only the question embeddings that were generated using SBERT (Reimers and Gurevych, 2019). Infinite answer set questions are also relayed to a Pixtral-Large-Instruct-2411 model with two examples, but the embeddings are now computed again from the image and the question using CLIP (Radford et al., 2021). Multiple-choice and unanswerable questions are handled again by InternVL3-78B, but now one example is chosen based on the embedding of the question computed by SBERT. A problem with this setup is that it assumes that the question types are given, which is not realistic in a real-world scenario. Nevertheless, it is impressive that they could reach third place without fine-tuning a model.

Our system (Schleid et al., 2025), which is the base for this thesis, reached the fourth position in the SciVQA shared task and is followed by Movva and Marupaka (2025). Similar to Ventura et al. (2025), they also used CoT prompting but expanded the approach by first using one prompt to extract the relevant information from the chart and then querying the VLM a second time to generate an answer based on the retrieved context. This improved the results in comparison to a single prompt without CoT by 1.1%. They tried this approach with multiple VLMs, achieving the best results on InternVL3-8B (Zhu et al., 2025), where the model without multi-step CoT prompting already reached good results. The second best VLM was Qwen2.5-VL 7B (Bai et al., 2025) where the multi-step CoT prompting led to substantial improvements of 5.6%.

2. https://huggingface.co/mistralai/Pixtral-Large-Instruct-2411

4 Experiments

To explore the influence of the prompting strategy, the effectiveness of domain-specific fine-tuning, and the importance of the training dataset size, we conducted multiple experiments to tackle the task. Firstly, we tried zero-shot inference in combination with prompt engineering (Section 4.1). Secondly, we performed one-shot prompting with one example (Section 4.2). Third, the VLMs were fine-tuned on the dataset provided by the task[1] (Section 4.3). Fourth, we expanded the fine-tuning by including the SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024) datasets in the training data (Section 4.4). Fifth, in reaction to the evaluation results, the infinite answer set questions in the SciVQA train dataset were manually annotated, and another model was trained on the updated training dataset (Section 4.5). Lastly, to confirm the choice of the modules targeted in the fine-tuning, different settings were compared (Section 4.6).

All approaches were tested on the 7B and 32B variants of the Qwen2.5-VL model (Bai et al., 2025) and the first two on GPT-4o mini[1].

4.1 Zero-Shot

In the zero-shot prompt experiment, the model is given clear instructions on how to respond to different types of questions. Suppose the question cannot be answered with the given information, a desired output is provided to reduce hallucinations. Together, these basic instructions on the answer format compose the baseline prompt, which can be seen in the Appendix A.1.

A refined prompt provides the model with the caption of the chart as an additional information source. The system prompt of this refined template also incorporates a role instruction to prime the model to act as an expert in data analysis. Furthermore, the prompt encourages the model to answer in a short and precise manner. This refined prompt is given in Appendix A.2. It is compared against the baseline prompt template to evaluate the effectiveness of the prompt engineering that led to the refined version.

1. https://openai.com/index/GPT-4o-mini-advancing-cost-efficient-intelligence/

4.2 One-Shot

To assess the influence of using one-shot prompting, the refined user prompt is expanded with an example. If the target question is a multiple-choice question, we align the example to the target by using a multiple-choice question as an example. Otherwise, an infinite answer set question is used. The complete one-shot prompt can be seen in the Appendix in Figure A.5. The multiple-choice example question and the infinite answer set example question were selected from the training split of the SciVQA dataset to have one visual and one non-visual question. The specific sample was chosen since it requires multiple steps to reach the answer. For the infinite answer set question, the example requires first determining which line corresponds to which category in order to filter out one category specified in the question. Then it needs to be determined which of the remaining lines has the maximum value at a specified position on the x-axis. For the multiple-choice question, it is also necessary to first determine the correct interval on the x-axis that is specified in the question and then to find the line with the maximum value in the interval. The goal is to allow the models to recognize and replicate each sub-step from this single example.

4.3 Fine-tuning Qwen2.5-VL for Scientific VQA

From the PEFT methods explained in Subsection 2.3.4, the Reparameterized PEFT method LoRA was chosen to fine-tune the Qwen2.5-VL models to perform VQA on charts. Nonetheless, one could create a hybrid Mixture-of-Experts-based model by fine-tuning one model for each chart type or dedicated models for different question types. However, to keep it simple and to reduce the number of models that need fine-tuning and, therefore, also hyperparameter tuning, we decided to focus on fine-tuning one general model. LoRA excels in such fine-tuning of large-scale models in environments with constrained computational resources, which makes it ideal for our needs.

Hyperparameter tuning determined the LoRA parameters rank, alpha, and dropout. The target metric for the hyperparameter tuning was the average of the ROUGE-1, ROUGE-L, and BERTScore F1-Scores. These metrics are explained more closely in Section 5.1, and their average also determined the ranking in the SciVQA competition. The evaluation of the hyperparameter tuning was done on the SciVQA validation split (1680 questions). Due to the high computational cost, the hyperparameter tuning is only performed for up to two epochs. After that, the resulting LoRA parameters are used to train a 32B model for one to four epochs to find the optimal number of training epochs. Finally, a 7B and a 32B model are trained with the optimal LoRA parameters for the optimal number of epochs. The original LoRA paper by Hu et al. (2022) tested which weight matrices should be adapted with LoRA, given a limited budget of parameters. They achieve the best performance by adapting the W_q and the W_v weight matrices that project the queries and values in the attention modules. Therefore, these are also the weight matrices that receive fine-tuning in our experiments. This decision is also verified by experimenting with different target modules in the experiment described in Section 4.6. To further reduce the memory requirements, we use 8-bit quantization for all experiments. This quantization is performed with the LLM.int8() method of Dettmers et al. (2022) introduced in Subsection 2.3.5. As a loss function, cross-entropy loss was used, and the optimizer was the AdamW optimizer (Loshchilov and Hutter, 2019) with a linear learning rate scheduler and a warm-up ratio of 0.03. The β_1 and β_2 parameters for the AdamW optimizer were the default values of 0.9 and 0.999, respectively. The maximum learning rate is set to 2×10^{-4}. A learning rate of 1×10^{-4} was also tried but showed inferior results. The effective batch size is 4, since larger batch sizes exceed the VRAM capacity for the 32B model. For the fine-tuning of the 7B models, one NVIDIA RTX A6000 (48GB VRAM) GPU was used, and for the 32B model, one NVIDIA A100 (80GB VRAM).

As a prompt template, the refined prompt without a one-shot example was used (see Appendix A.2) and both hyperparameter tuning and final fine-tuning were performed on the SciVQA training dataset[1] (15K questions). The final fine-tuned models were automatically evaluated on the SciVQA test set, and the 32B variant was additionally manually evaluated on the SciVQA validation dataset to ensure the correctness of the evaluation results. The evaluation metrics are explained in more detail in Section 5.1.

4.4 LoRA Fine-Tuning with Other Datasets

This section first describes the creation of a combined dataset from SciVQA, ArXivQA, and SpiQA, and subsequently the fine-tuning of the Qwen2.5-VL 7B and 32B models with this dataset.

4.4.1 Expanding the SciVQA Dataset

To explore the effects of using more domain-specific data for fine-tuning, we sought datasets similar to SciVQA[1]. We therefore incorporated the SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024) datasets as additional data sources, as they also use real-world scientific charts and primarily contain infinite answer set and multiple-choice questions, respectively. To avoid overlap, any questions from papers that were also scraped in the SciVQA dataset were excluded.

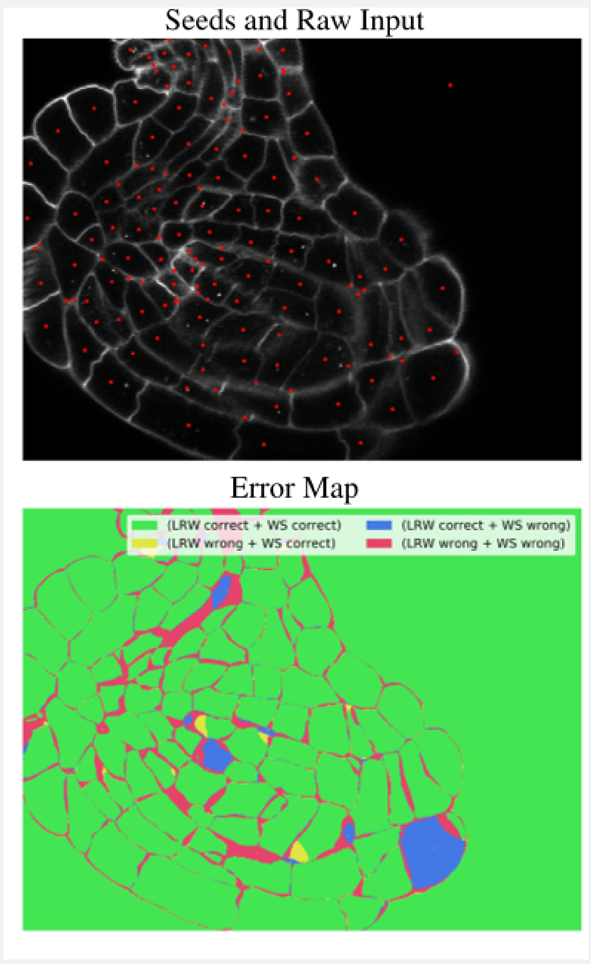
Furthermore, the SpiQA dataset also contains questions about tables. These questions were filtered out. To align the remaining SpiQA questions closer to the SciVQA questions, only questions with answers that have at most 50 characters were retained, leaving 39K mostly infinite answer set questions. The resulting average answer length in the filtered SpiQA questions is 15.3 characters, and therefore relatively close to the average answer length of 14.3 characters of the SciVQA dataset. Figure 4.1 illustrates an example from the resulting training split of the SpiQA dataset. In addition to classical line charts, SpiQA also includes special visualizations, such as the colored photograph shown. The answer "Green." in Figure 4.1 demonstrates the shortness of the responses in the SpiQA dataset but also inconsistencies, as some answers end with punctuation while others do not.

From ArXivQA, only multiple-choice questions with 4 options were kept, since the multiple-choice questions in the SciVQA dataset also have 4 options. This yielded 61K questions, and one example of these questions is shown in Figure 4.2. The format of the answer option labels was matched to the format used in the SciVQA dataset. Unfortunately, the ArXivQA dataset does not provide the captions of the crawled images.

The images from both datasets were resized to a maximum of 500K pixels while preserving the aspect ratio. From these filtered datasets, a test set of 4200 questions was split off to be available for cross-evaluation later on. It is ensured that the questions in the test sets do not refer to figures that were also referred to in the training sets. Apart

Example question from SpiQA dataset

Image:

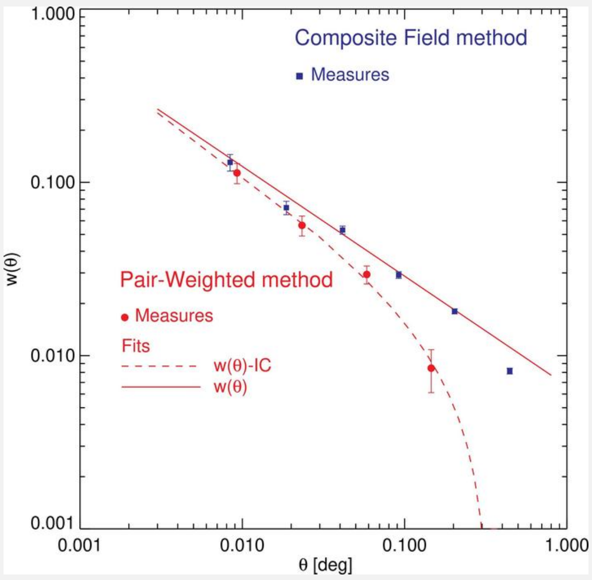


Caption:
Qualitative comparison between Learned Random Walker and Watershed applied on confocal microscopy data. Blue and yellow areas represent the regions where the Learned Random Walker outperformed Watershed and vice versa, respectively.
Question:
In the image, what color represents the regions where both the Learned Random Walker (LRW) and Watershed (WS) algorithms correctly identified the cell boundaries?
Answer:
Green.

Figure 4.1: A sample from the training split of the filtered and processed SpiQA dataset.

Example question from ArXivQA dataset

Image:



Question:
What method is shown to have a consistently lower value of w(θ) over the range of θ depicted?
Answer Options:
A: Composite Field method
B: Pair-Weighted method
C: Both methods show similar values
D: The method cannot be determined from the graph
Answer:
B

Figure 4.2: A sample from the training split of the filtered and processed ArXivQA dataset.

from that, the samples in the test splits are randomly sampled from the corresponding datasets. The remaining train splits of the SpiQA and ArXivQA datasets, along with the SciVQA train dataset, were merged into a combined dataset with 107K questions.

4.4.2 Applying the Combined Dataset for Fine-Tuning

The combined dataset, whose creation was described in Subsection 4.4.1, was used to fine-tune both the 7B and 32B Qwen2.5-VL models. To ensure a fair comparison of the results obtained in this experiment with the results from the fine-tuning on the original SciVQA dataset alone (see Section 4.3), hyperparameter tuning was again performed by training 7B models on the combined dataset with different training parameters. The resulting LoRA parameters rank, alpha, and dropout were used both for the 7B and the 32B model variants. The other training parameters and the used GPUs were the same as described in Section 4.3.

To explore the effectiveness of the different datasets individually, the data from ArXivQA, as well as SpiQA, were used separately as training data for Qwen2.5-VL 7B models. Furthermore, test splits from these datasets were set aside to employ cross-validation across the different models trained on the different datasets. Both test splits contain 4200 questions. To ensure a fair comparison, hyperparameter tuning was also employed for the ArXivQA model and for the SpiQA model. Since both the ArXivQA and the SpiQA datasets are substantially larger than the SciVQA dataset, a subset of 15120 questions was randomly sampled from both of them and used to train another model for each dataset. Furthermore, a Qwen2.5-VL 7B model that did not receive fine-tuning was also prompted in a zero-shot fashion with the questions from the test splits. The results are presented in Subsection 5.3.1.

4.5 Improving SciVQA Training Data

Our evaluation showed that the infinite answer set questions, especially the visual subset of them, pose a major challenge for the models. Furthermore, during the manual evaluation, some inaccuracies and even errors were found in the golden answers of the validation set. Motivated by this, all infinite answer set questions of the SciVQA train split were manually assessed to ensure the correctness and accuracy of the answers. This resulted in changing 1020 of the 4320 infinite answer set answers. From these 1020 changed answers, 338 belonged to non-visual questions and 682 to visual questions. In addition, 73 questions were removed since no certain answer could be determined for the question. We published the improved dataset[2].

Using this changed training dataset, another Qwen2.5-VL 32B model was trained. Apart from the training dataset, the same configuration as described in Section 4.3 was employed.

4.6 Fine-Tuning Different Modules

As described in Section 4.3, the original LoRA paper reached its best results by fine-tuning only the W_q and W_v weight matrices in the attention layers of the LLM Decoder (Hu et al., 2022). Therefore, the previously described experiments only fine-tuned these two modules. This experiment should assess whether fine-tuning a different combination of modules can lead to superior results. Therefore, in one experiment, in addition to the W_q and W_v weight matrices, also the W_k and W_o weight matrices that project the keys and outputs in the self-attention modules will be adapted using LoRA. Secondly, in another experiment, the fine-tuning of all linear modules is tested. This includes the linear layers in the vision transformer, the projector, and the projection matrices in the LLM module. For the LoRA parameters, a rank of 64, an alpha of 128, and a dropout of 0.2 were used. This is the setting that was determined in the hyperparameter tuning on the SciVQA training set (see Table 5.4). It has to be noted that during this hyperparameter tuning, only W_q and W_v received fine-tuning, which could bias the

2. https://github.com/Flo0620/SciVQA_Repo/blob/main/Fine-Tuning/shared_task/SciVQA_train_combined_manually_labeled.json

results in favor of these two target modules. The remaining training settings were kept the same as described in Section 4.3. In both experiments, a 7B and a 32B model are fine-tuned and evaluated on the validation dataset. In addition, the training time and the number of parameters that received fine-tuning will be compared.

5 Evaluation

This chapter presents the results of the experiments that were introduced in Chapter 4. Therefore, Section 5.1 first presents the evaluation metrics employed in the SciVQA competition and subsequently adopted in this thesis. It also explains how the additional manual evaluation was conducted and why it was necessary.

Subsequently, Section 5.2 evaluates the experiments in order to answer the research questions from Section 1.1. Subsection 5.2.1 and Subsection 5.2.2 deal with the impact of prompt engineering and the influence of a one-shot example, uncovering shortcomings of the automatic evaluation metrics. The evaluation of the model fine-tuned on the original SciVQA dataset in Subsection 5.2.3 results in the best performance among all experiments. This was also confirmed by a manual evaluation. The results of the fine-tuning on the combined dataset of SciVQA, ArXivQA, and SpiQA are evaluated in Subsection 5.2.4, where a decrease in performance in comparison to only fine-tuning on SciVQA is discovered. Subsection 5.2.5 also finds a reduction in performance when fine-tuning on a manually improved version of the SciVQA dataset. A manual evaluation confirms that although the accuracy on the infinite answer set questions is increased, the overall accuracy decreases. Finally, Subsection 5.2.6 assesses the results on different configurations of target modules for the fine-tuning. It confirms the initial choice, which was supported by the original LoRA paper, to target the query and value projection matrices.

In the ablation studies in Section 5.3, cross-validation between the SciVQA, the ArXivQA, and the SpiQA datasets was performed to gain further insight into the performance reduction caused by the fine-tuning on a combination of the datasets in Subsection 5.3.1. Furthermore, Subsection 5.3.2 studies the effect of the training dataset size and confirms that with an increase in the training dataset size, the performance also increases. It also suggests that better results might be possible if the SciVQA training dataset were larger. Subsection 5.3.3 analyzes the accuracy of the 32B model fine-tuned on the original SciVQA data per figure type.

Finally, Section 5.4 compares the achieved results with the results that other participants of the SciVQA shared task achieved.

5.1 Evaluation Metrics

In this section, the evaluation metrics used in this thesis, namely the BERTScore and ROUGE, are introduced. These were also used in the SciVQA shared task, where the average of the F1-Scores of the BERTScore, ROUGE-1, and ROUGE-L determined the leaderboard. Additionally, Subsection 5.1.3 explains the manual evaluation procedure, which was necessary as the automatic methods were found to be biased against long answers. This was a problem if the model did not adhere to the short answer format of the ground-truth answers.

5.1.1 ROUGE

ROUGE was created as a metric to automatically evaluate the quality of generated summaries, given a reference summary that could, for instance, have been created by a human. It includes various methods such as ROUGE-1 and ROUGE-L, which are the methods used in this thesis.
ROUGE-1 is the simplest metric and measures how many words from the reference summary are also in the generated output. The original paper that introduced ROUGE (Lin, 2004) only defined the recall-oriented version of ROUGE-1 as

$$\text{ROUGE-1}_{\text{recall}} = \frac{\sum_{u \in R} \text{Count}_{\text{match}}(u)}{\sum_{u \in R} \text{Count}_R(u)} \tag{5.1}$$

where R is the reference summary and $\text{Count}_R(u)$ counts how often the unigram u occurs in the reference summary. $\text{Count}_{\text{match}}(u)$ is the minimum of the number of occurrences of the unigram in the generated summary and the number of occurrences of the unigram in the reference summary.
In this thesis the implementation by Google-Research[1] was used, which defines the ROUGE-1 precision as

$$\text{ROUGE-1}_{\text{precision}} = \frac{\sum_{u \in R} \text{Count}_{\text{match}}(u)}{\sum_{u \in R} \text{Count}_G(u)} \tag{5.2}$$

where $\text{Count}_G(u)$ is the number of occurrences of a given unigram in the generated summary.
The ROUGE-1 F1-Score is defined as

$$\text{ROUGE-1}_{\text{f1}} = \frac{2 \cdot \text{ROUGE-1}_{\text{recall}} \cdot \text{ROUGE-1}_{\text{precision}}}{\text{ROUGE-1}_{\text{recall}} + \text{ROUGE-1}_{\text{precision}}} \tag{5.3}$$

Apart from the ROUGE-1 metric, the ROUGE-L metric is used. This measures the longest common subsequence between the reference and the generated summary. A common subsequence is defined as a sequence of tokens that occurs in the same order in the reference summary, but in the reference summary, the sequence can be interrupted by other tokens. $LCS(R, G)$ is then the length of the longest common subsequence between the reference summary R and the generated summary G. The recall of the ROUGE-L metric is then defined as

$$\text{ROUGE-L}_{\text{recall}} = \frac{LCS(R, G)}{m} \tag{5.4}$$

1. https://github.com/google-research/google-research/tree/master/rouge

where m is the length of the reference summary. The precision is defined as

$$\text{ROUGE-L}_{\text{precision}} = \frac{LCS(R,G)}{n} \tag{5.5}$$

where n is the length of the generated summary. The equation for the F1-Score of the ROUGE-L metric is

$$\text{ROUGE-L}_{\text{f1}} = \frac{(1+\beta^2)\cdot \text{ROUGE-L}_{\text{recall}} \cdot \text{ROUGE-L}_{\text{precision}}}{\text{ROUGE-L}_{\text{recall}} + \beta \cdot \text{ROUGE-L}_{\text{precision}}} \tag{5.6}$$

where β is a factor to put more weight on the recall or precision (Lin, 2004). In the implementation[1] used in this thesis, β is set to 1, which results in the usual formula for the F1-Score.

5.1.2 BERTScore

Instead of exact matches, the BERTScore metric, proposed by Zhang et al. (2020), computes a similarity score between each token in a generated sequence and a reference sequence. This similarity score is the cosine similarity between the embedded tokens. To create the token embeddings, a pretrained BERT model (Devlin et al., 2019) is used to generate contextual embeddings, which can robustly match paraphrases and capture distant dependencies and ordering. To achieve that, contextual embeddings also take the surrounding words into account to generate a vector representation for a token. That is, the same word can have different embeddings in different sentences (Zhang et al., 2020).

Given a tokenized reference sequence $x = \langle x_1, \dots, x_m \rangle$, contextual embedding vectors $\langle \mathbf{x}_1, \dots, \mathbf{x}_k \rangle$ are generated for the sequence. Also, the generated sequence $\hat{x} = \langle \hat{x}_1, \dots, \hat{x}_n \rangle$ is mapped to a sequence of embedded vectors $\langle \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_l \rangle$. Between the embedded vectors, the pairwise cosine similarity can be computed with $\frac{\mathbf{x}_i^\top \hat{\mathbf{x}}_j}{\|\mathbf{x}_i\| \|\hat{\mathbf{x}}_j\|}$, where $\mathbf{x}_i$ is an embedded reference token and $\hat{\mathbf{x}}_\mathrm{j}$ is an embedded token from the generated sequence. Since the BERTScore uses pre-normalized embedding vectors, the calculation of the cosine similarity reduces to $\mathbf{x}_i^\top \hat{\mathbf{x}}_j$ (Zhang et al., 2020).
The BERTScore now matches each token in x with a token in $\hat{x}$ to compute the recall. This matching is done in a greedy fashion, where each token is matched to the most similar token in the other sequence. To compute the precision, each token in $\hat{x}$ is matched to a token in x. The resulting equations for recall, precision, and F1 score are:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \tag{5.7}$$

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \tag{5.8}$$

$$F_{\text{BERT}} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{5.9}$$

5.1.3 Manual Evaluation

In the SciVQA dataset, the ground-truth answers are very short, usually only containing one word or number. The average answer length across the whole dataset is 14.3

characters. If the long preset answers of the unanswerable questions are excluded, an answer is on average 4.2 characters long. This is a problem if the models do not generate answers that adhere to this short format and instead provide long additional explanations in an unstructured output that prevents the extraction of the actual final answer. For those long answers, the recall of the ROUGE and BERTScore metrics is capped at one. The precision, however, reduces drastically since the long generated answer contains the one word from the ground-truth answer at most a few times, and all other words reduce the precision value. This leads to bad F1-Scores even if the long answer is correct.

Therefore, especially in the zero-shot and one-shot scenarios where the models have not adapted the short answer format, another metric is required. After the fine-tuning, the models provide the answers in the short format required by the dataset. Nonetheless, to ensure the correctness and interpretability of the results and to enable a comparison with the results from the zero-shot and one-shot experiments, manual evaluation is still conducted for the final model fine-tuned on the SciVQA dataset. For this manual evaluation, one annotator assessed the given answers in the context of the question, the chart image, and the ground-truth answer and decided if the answer was correct or not. The formulation of the answer was ignored. For questions that required reading values from the charts, it was difficult to establish a strict guideline for the required level of precision due to the diversity of the charts. To increase the quality of the results, these answers were evaluated rather strictly, such that even some of the ground-truth answers would have been considered incorrect. If, for instance, the model was asked to provide the y-value of a data point and it returned the value of the closest y-axis label, but the data point was clearly between two y-axis labels, the answer was considered false. Qualitatively, the models usually returned answers for these questions that were in the region of the correct result, but often not precise enough. The final score of the manual evaluation is the accuracy of the responses, quantified as:

$$\text{Accuracy} = \frac{\text{Number of correctly answered questions}}{\text{Total number of questions}} \tag{5.10}$$

5.2 Main Results

In the SciVQA competition, the experiments were evaluated using the average of the F1-Scores of ROUGE-1, ROUGE-L, and BERTScore on the test split of the SciVQA dataset, which contains 4200 questions. To enable the comparison with the results of the other participants, the automatically evaluated results of our experiments are presented in Table 5.1. For the zero-shot and one-shot experiments, but also for the fine-tuning on the SciVQA data, manual evaluation is additionally employed on the SciVQA validation dataset (1680 questions).

5.2.1 Zero-Shot

The goal of the zero-shot experiment is to assess the effectiveness of prompt engineering and to get a baseline for the following experiments. Therefore, the results of a baseline prompt template are compared against the results with a refined prompt. The evaluation was done on the SciVQA validation dataset. The average F1-Scores for ROUGE-1, ROUGE-L, and the BERTScore for the Qwen2.5-VL 7B and 32B models, as well as for the GPT-4o mini model, are shown in Table 5.2. It is visible that the refined prompt led

Experiment	GPT-4o mini	Qwen 7B	Qwen 32B
Zero-Shot*	0.5435	**0.5984**	0.5206
One-Shot*	**0.6341**	0.5986	0.5258
Fine-tuning on SciVQA	-	0.8128	**0.8361**
Fine-tuning + other datasets	-	0.7989	**0.8272**
Fine-tuning on manually labeled SciVQA	-	0.8066	**0.8263**

Table 5.1: Performance comparison of the Qwen2.5-VL 7B, Qwen2.5-VL 32B, and GPT-4o mini models on the SciVQA test set (4200 questions) across different learning paradigms: zero-shot, one-shot, fine-tuning on SciVQA, fine-tuning with additional datasets, and fine-tuning on the manually improved SciVQA dataset. Reported values are the average of the F1-Scores of ROUGE-1, ROUGE-L, and BERTScore. The best score in each setting is highlighted in bold.
* In the zero-shot and one-shot settings, the models did not adhere to the answer format and therefore gave long answers. This leads to biased results for the automatic evaluation metrics. The results for the zero-shot and one-shot experiments are only provided to enable a comparison with other SciVQA participants under the competition's evaluation protocol, but do not necessarily reflect the quality of the answers.

	Qwen2.5-VL 7B	Qwen2.5-VL 32B	GPT-4o mini
Baseline Prompt	0.6148	0.4586	0.5045
Refined Prompt	0.6279	0.5197	0.5460

Table 5.2: Average of the F1-Scores of BERTScore, ROUGE-1, and ROUGE-L across the Qwen2.5-VL 7B and 32B models and for the GPT-4o mini model in the zero-shot setting with the baseline and the refined prompt. The evaluation was performed on the SciVQA validation dataset (1680 questions).

to better results than the baseline prompt for each of the models. However, surprisingly, the 7B model has the overall best scores. Due to its larger size, one would expect that the 32B model should outperform the 7B model. Although the size of the GPT-4o mini model is not public, it is very likely that it is also larger than the 7B Qwen2.5-VL model, and therefore, one would expect that it reaches better results than the 7B model.

However, taking a closer look at the provided answers revealed that the answers given by the 32B model with the refined prompt had an average length of 225.2 characters, while the 7B model only had an average answer length of 39.3 characters. The GPT-4o mini model had an average length of 59.7 characters. In comparison, the validation set answers only have an average answer length of 14.4 characters. If the unanswerable questions with their relatively long preset answer are ignored, the average answer length of the remaining questions in the validation set is reduced to 4.3 characters. As described in Subsection 5.1.3, the automatic evaluation metrics are biased against long answers. This bias can be seen in our results, as the average answer length of the models aligns with the automatic performance evaluation, where the 7B model performs the

Question Type	**Qwen2.5-VL 7B**			**Qwen2.5-VL 32B**			**GPT-4o mini**		
	Base-line	**Re-fined**	**One-Shot**	**Base-line**	**Re-fined**	**One-Shot**	**Base-line**	**Re-fined**	**One-Shot**
Infinite									
visual	0.3583	0.4625	0.2167	0.5542	0.5625	0.4917	0.2000	0.4500	0.3875
non-visual	0.4208	0.6333	0.3792	0.7000	0.7500	0.7500	0.4583	0.6833	0.6500
Yes/No									
visual	0.1875	0.3542	0.2042	0.7208	0.7708	0.7833	0.3708	0.6625	0.6458
non-visual	0.0875	0.4458	0.2500	0.7208	0.7875	0.7583	0.4333	0.7125	0.7000
Multiple-Choice									
visual	0.7083	0.6708	0.5750	0.6958	0.6958	0.6458	0.4708	0.5042	0.5042
non-visual	0.6208	0.6542	0.5792	0.6750	0.6958	0.6792	0.4458	0.5125	0.5667
Unanswerable	0.9833	0.8958	0.9917	0.7125	0.6250	0.6417	0.9208	0.6250	0.7708
Overall	**0.4810**	**0.5881**	**0.4565**	**0.6827**	**0.6982**	**0.6786**	**0.4714**	**0.5929**	**0.6036**

Table 5.3: Comparison of the accuracy of the Qwen2.5-VL 7B and 32B models, and the GPT-4o mini model across the zero-shot setting with the baseline and the refined prompt and in the one-shot setting. The evaluation was done manually on the SciVQA validation dataset (240 questions per type). The answer format is not taken into account.

best, followed by GPT-4o mini, and the 32B model performs the worst. Therefore, as long as the models do not adhere to the answer format, manual evaluation is required to ensure the correctness of the evaluation results.

The results of this manual evaluation on the SciVQA validation dataset are shown in Table 5.3. They confirm that the refined prompt leads to performance improvements across all models compared to the baseline prompt. The largest increase can be seen for the GPT-4o mini model, where the results with the refined prompt are improved by 25.8% relative to the baseline prompt. For the Qwen2.5-VL 7B model, the improvements are also substantial with an increase of 22.3%. For the overall best-performing model, the Qwen2.5-VL 32B model, the accuracy gains achieved by the refined prompt are not as big. The improvement is with 2.3%, much smaller than for the other models. For the 32B model, the baseline prompt even leads to better results on the unanswerable questions, but the refined prompt works better on the remaining question types.

Another notable finding that the manual evaluation reveals is that the Qwen2.5-VL 32B model achieves the best results by a significant margin. With the refined prompt template, it reaches an accuracy of 0.698, which corresponds to a relative increase of 17.8% in accuracy compared to the GPT-4o mini model with the refined template. The 7B model actually performs the worst with an accuracy of 0.588. These results fit with the assumption that the larger model size of the 32B model leads to better results than the 7B model. However, it is surprising that GPT-4o mini performs so poorly compared to the 32B model. A possible explanation for this could be that the pretraining of the Qwen2.5-VL models focused on visual inputs, as it is explicitly a model made for vision tasks, while GPT-4o mini might only have had a relatively small fraction of its pretraining data dedicated to visual inputs and focused more on solely language-based tasks.

Taking a closer look at the results per question type reveals that the Qwen2.5-VL 7B model, in general, and GPT-4o mini with the baseline prompt, have a strong accuracy on unanswerable questions but perform very poorly on infinite answer set and yes/no questions. The 7B model with the baseline prompt stated that 857 of the 1680 questions would be unanswerable. In fact, the validation set contains 240 unanswerable questions. 410 of the questions that were marked as unanswerable were yes/no questions. This explains the very poor performance on this question type. With the refined prompt, the 7B model still marked 574 questions as unanswerable, but this reduction led to an improvement of the results on the infinite answer set and yes/no questions. GPT-4o mini with the baseline prompt marks 733 questions as unanswerable. With the refined prompt, this is significantly reduced to 207, which leads to a drop in the accuracy on unanswerable questions but also to massive improvements on the infinite answer set and yes/no questions, subsequently resulting in the 25.8% improvement in the overall accuracy.

Although the improvement on the Qwen2.5-VL 32B model is with 2.3% only modest, the refined prompt led to massive accuracy gains on the 7B and GPT-4o mini models. Overall, these results highlight that while the benefits of prompt engineering depend on the underlying model, it remains a powerful tool for improving performance. The manual evaluation also confirmed that the average of the F1-Scores as an evaluation metric is not applicable as long as the models do not adhere to the answer format and provide long answers. Nonetheless, the automatically evaluated results on the test set are reported in Table 5.1 to provide the complete evaluation results. This enables a comparison with the results of the other SciVQA participants under the evaluation protocol of the SciVQA competition, although they do not reflect the true performance of the models. Due to the time-intensive nature of manually evaluating the answers and the larger size of the test set, the manual evaluation is limited to the validation set.

5.2.2 One-Shot

Adding a one-shot example to the refined prompt still does not cause the model to answer with just the result without any explanation. For the GPT-4o mini model, the one-shot prompt actually resulted in a decrease of the average answer length from 59.7 to 33.9 characters on the validation dataset. However, for the Qwen2.5-VL 7B and 32B models, the one-shot template led to an increase in the average answer length from 39.3 to 42.6 and from 225.2 to 445.3 characters, respectively. Therefore, again, the results need to be manually evaluated to ensure correctness. The results of this manual evaluation on the SciVQA validation dataset can be seen in Table 5.3. The table does not show unambiguous results on the effectiveness of adding a one-shot example. For the Qwen2.5-VL 7B model, the accuracy drops significantly from 0.588 to 0.457. The performance is also reduced for the 32B model, but the drop from 0.698 to 0.679 is smaller. On the other hand, for the GPT-4o mini model, a slight increase in accuracy can be noticed. However, the improvement is with 1.07 percentage points only small and still the 32B model in the zero-shot setting vastly outperforms the GPT-4o mini model with the one-shot example. Looking at the automatic evaluation metrics in Table 5.1 shows that the average of the F1-Scores improved a lot more for the GPT-4o mini model, increasing from 0.544 to 0.634 on the test set. The same can be observed on the validation dataset, where the average F1-Score of the GPT-4o mini model increases from 0.546 to 0.635. Together with the reduction of the average answer length for the GPT-4o mini model, this suggests that the model was able to extract the answer format through the one-shot example, but the actual answer quality, in terms of correctness, is only slightly increased. Notably, Table 5.3 shows that the performance gains of GPT-4o mini are mainly attributable to the unanswerable questions, where the accuracy was improved by 14.6 percentage points. The results on non-visual multiple-choice questions also improved by 5.4 percentage points. On the other hand, the accuracy on yes/no and infinite answer set questions worsened with the one-shot prompt. This is surprising since the one-shot example was a visual multiple-choice question if the question was a multiple-choice question and a non-visual infinite answer set question otherwise. Therefore, instead of the observed decrease, an increase in accuracy would have been expected for these question types.

For the 7B model, the one-shot prompt led to 861 of the 1680 questions being answered as unanswerable. For the refined zero-shot prompt, it marked 574. This leads to even poorer results on infinite answer set questions and yes/no questions, as visible in Table 5.3. For the Qwen2.5-VL 32B model, the overall loss in accuracy originates from a worse accuracy on visual infinite answer set questions and a slightly reduced performance on multiple-choice questions. Again, this is unexpected since the example in the one-shot prompt is an infinite answer set or multiple-choice question, depending on the question type, but still the performance on those question types is reduced.

In summary, the success of providing a one-shot example seems to depend on the underlying model. GPT-4o mini seems to be capable of overall gaining some insights from the examples, although it also reduces the performance on some question types. For the Qwen2.5-VL models, the one-shot prompt reduces the performance. Therefore, in the remaining experiments with the Qwen2.5-VL models, the refined zero-shot prompt is used.

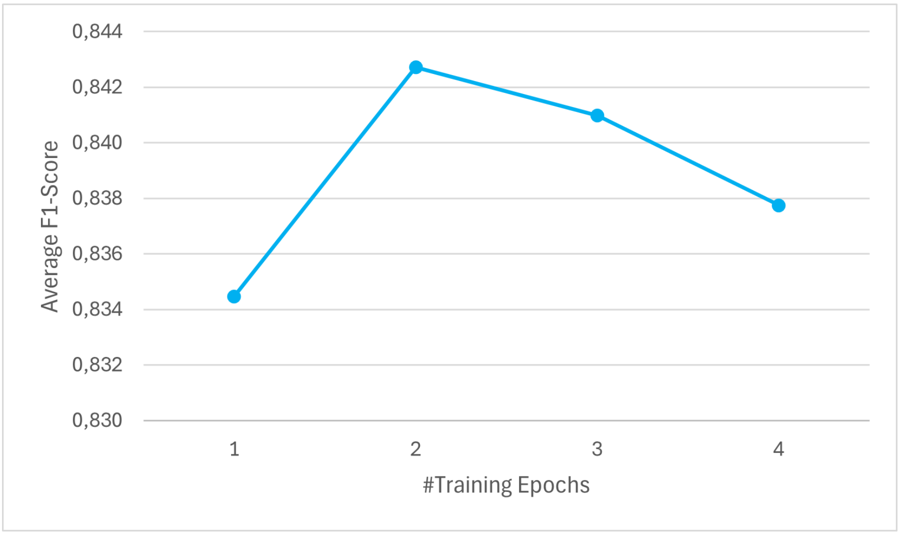


Figure 5.1: Average of the F1-Scores of BERTScore, ROUGE-1, and ROUGE-L across one to four training epochs for fine-tuning Qwen2.5-VL 32B with LoRA rank = 64, alpha = 128, dropout = 0.2. The fine-tuning was performed on the SciVQA train split, and the evaluation was done on the SciVQA validation dataset (1680 questions).

5.2.3 Fine-Tuning with the Original SciVQA Dataset

This section evaluates the effectiveness of fine-tuning the Qwen2.5-VL models on the SciVQA training dataset. As described in Section 4.3, before the final fine-tuning, hyperparameter tuning was employed first to determine the optimal LoRA parameters. After the fine-tuning, the models adhere to the answer format of the dataset. For instance,

			Train Epochs: 1				Train Epochs: 2			
r	α	**d**	**BERT**	**R-1**	**R-L**	**Avg.**	**BERT**	**R-1**	**R-L**	**Avg.**
16	16	0.1	0.9814	0.7250	0.7242	0.8102	0.9810	0.7230	0.7219	0.8087
16	16	0.2	0.9810	0.7278	0.7268	0.8119	0.9806	0.7270	0.7262	0.8113
16	32	0.1	0.9815	0.7259	0.7250	0.8108	0.9825	0.7323	0.7313	0.8154
16	32	0.2	0.9811	0.7188	0.7178	0.8059	0.9817	0.7190	0.7183	0.8063
32	32	0.1	0.9811	0.7288	0.7281	0.8127	0.9810	0.7330	0.7323	0.8155
32	32	0.2	0.9821	0.7271	0.7267	0.8120	0.9822	0.7279	0.7271	0.8124
32	64	0.1	0.9816	0.7381	0.7374	0.8190	0.9819	0.7355	0.7346	0.8173
32	64	0.2	0.9808	0.7301	0.7288	0.8133	0.9810	0.7347	0.7337	0.8165
64	64	0.1	0.9817	0.7318	0.7310	0.8149	0.9828	0.7435	0.7425	0.8230
64	64	0.2	0.9823	0.7400	0.7391	0.8205	0.9819	0.7427	0.7420	0.8228
64	128	0.1	0.9805	0.7315	0.7307	0.8156	0.9811	0.7316	0.7304	0.8144
64	**128**	**0.2**	0.9826	0.7388	0.7378	0.8197	**0.9822**	**0.7452**	**0.7437**	**0.8237**
128	128	0.1	0.9823	0.7401	0.7392	0.8205	0.9827	0.7434	0.7425	0.8228
128	128	0.2	0.9821	0.7376	0.7367	0.8188	0.9819	0.7437	0.7427	0.8228
128	256	0.1	0.9803	0.7270	0.7260	0.8111	0.9815	0.7404	0.7395	0.8205
128	256	0.2	0.9821	0.7329	0.7318	0.8156	0.9823	0.7361	0.7352	0.8179
256	256	0.1	0.9799	0.7292	0.7279	0.8123	0.9808	0.7332	0.7320	0.8153
256	256	0.2	0.9804	0.7302	0.7291	0.8133	0.9808	0.7334	0.7320	0.8154
256	512	0.1	0.9809	0.7263	0.7251	0.8108	0.9813	0.7319	0.7306	0.8146
256	512	0.2	0.9825	0.7249	0.7236	0.8103	0.9818	0.7359	0.7348	0.8175

Table 5.4: Evaluation with the SciVQA validation dataset (1680 questions) on the fine-tuned Qwen2.5-VL 7B model for the different hyperparameters LoRA rank, alpha, and dropout. The learning rate was always 2×10^{-4}, and the learning rate scheduler was linear. As the training dataset, the SciVQA train split was used. The metrics are the F1-Scores of BERTScore, ROUGE-1, ROUGE-L, and their average.

the average answer length of the final fine-tuned 32B model on the SciVQA validation dataset is, with 14.45 characters, nearly identical to the actual average answer length of the validation dataset (14.42 characters). Therefore, the bias of the automatic evaluation metrics against long responses no longer degrades the quality of the results, and the automatic metrics can be used for hyperparameter tuning.

The results of the hyperparameter tuning, which was performed on the Qwen2.5-VL 7B model with the SciVQA training dataset, can be seen in Table 5.4. As visible, the best performance after two training epochs was reached by setting the rank to 64, alpha to 128, and dropout to 0.2. To ensure that the performance peaks after two training epochs, a Qwen2.5-VL 32B model was fine-tuned on one to four epochs with the determined LoRA parameters. Figure 5.1 shows that the performance indeed reaches its maximum after two training epochs, possibly due to overfitting in later epochs. Therefore, the 7B and 32B models that were fine-tuned for two epochs with the LoRA parameters rank = 64, alpha = 128, and dropout = 0.2 are evaluated on the SciVQA test set.

The described fine-tuning solely on the SciVQA dataset led to the best result we could achieve across our experiments. As visible in Table 5.1, the Qwen2.5-VL 32B model reached an average F1-Score of 0.836. The expected superiority of the 32B model

Question Type	Qwen 32B zero-shot	Fine-tuned Qwen 32B
Infinite		
visual	0.5625	0.5458
non-visual	0.7500	0.7500
Yes/No		
visual	0.7708	0.8000
non-visual	0.7875	0.8167
Multiple-Choice		
visual	0.6958	0.7583
non-visual	0.6958	0.6958
Unanswerable	0.6250	0.9792
Overall	**0.6982**	**0.7637**

Table 5.5: Manual evaluation of results obtained for the Qwen2.5-VL 32B model, first in the zero-shot setting with the refined prompt and secondly fine-tuned for two epochs on the training split of the SciVQA dataset (15K questions). The table shows the fraction of correctly answered questions on the SciVQA validation dataset (1680 questions) per question type. Each question type contains 240 questions.

is also evident here, as it outperformed the fine-tuned 7B variant by 0.023. The GPT-4o mini models, which did not receive fine-tuning, were also substantially outperformed by a margin of 0.202.

Nonetheless, to ensure the correctness of the evaluation results and to enable comparability against the results from the zero-shot and one-shot experiments, the results of the fine-tuned 32B model on the SciVQA validation dataset are manually evaluated. This also improves the interpretability of the results, as the accuracy conveys the answer quality much more clearly than the average of F1-Scores of the automatic metrics. Table 5.5 shows the fractions of correctly answered questions for the 32B model that was fine-tuned for 2 epochs and compares it against the 32B model that is prompted with the refined prompt in the zero-shot setting. These results show the effectiveness of domain-specific fine-tuning, which led to a substantial improvement of 9.4% compared to the zero-shot prompting. A significant improvement was observed for unanswerable questions, suggesting that fine-tuning may have reduced hallucinations and helped the model to estimate when not to answer. In the zero and one-shot settings, an improvement on the unanswerable questions typically led to a performance reduction on the yes/no and infinite answer set questions. For the visual infinite answer set questions, this reduction is actually given, although it is relatively small. However, on the yes/no questions, the performance is improved compared to the zero-shot scenario. Another substantial increase in accuracy can be observed for the visual multiple-choice questions, improving from 0.696 to 0.758. Additionally, there is a marked difference in performance between visual and non-visual infinite answer set questions for both models. Referencing visual elements appears to be considerably more challenging in the infinite answer set context. Interestingly, this difficulty was not as pronounced for the yes/no questions, and the opposite is true for multiple-choice questions. This observation led to the evaluation of the accuracy of the model after fine-tuning it on training data, where the infinite answer set questions were manually improved with respect to their correctness and precision in Subsection 5.2.5.

			Train Epochs: 1				**Train Epochs: 2**			
r	**α**	**d**	**BERT**	**R-1**	**R-L**	**Avg.**	**BERT**	**R-1**	**R-L**	**Avg.**
16	16	0.1	0.9801	0.7142	0.7137	0.8027	0.9792	0.7069	0.7061	0.7974
16	16	0.2	0.9798	0.7179	0.7172	0.8050	0.9811	0.7238	0.7229	0.8093
16	32	0.1	0.9811	0.7177	0.7166	0.8051	0.9806	0.7084	0.7077	0.7989
16	32	0.2	0.9810	0.7177	0.7169	0.8052	0.9821	0.7227	0.7216	0.8088
32	32	0.1	0.9804	0.7222	0.7215	0.8080	0.9811	0.7233	0.7224	0.8090
32	32	0.2	0.9798	0.7256	0.7247	0.8101	0.9812	0.7220	0.7210	0.8081
32	64	0.1	0.9811	0.7178	0.7170	0.8053	0.9828	0.7128	0.7117	0.8025
32	64	0.2	0.9798	0.7117	0.7109	0.8008	0.9804	0.7174	0.7164	0.8047
64	**64**	**0.1**	0.9810	0.6922	0.6913	0.7882	**0.9834**	**0.7288**	**0.7277**	**0.8133**
64	64	0.2	0.9812	0.7231	0.7220	0.8088	0.9810	0.7202	0.7190	0.8067
64	128	0.1	0.9809	0.7095	0.7085	0.7996	0.9823	0.7206	0.7201	0.8077
64	128	0.2	0.9804	0.7140	0.7133	0.8026	0.9812	0.7114	0.7101	0.8009
128	128	0.1	0.9792	0.7141	0.7133	0.8022	0.9818	0.7147	0.7140	0.8035
128	128	0.2	0.9799	0.7035	0.7025	0.7953	0.9811	0.7146	0.7139	0.8032
128	256	0.1	0.9778	0.6983	0.6980	0.7914	0.9817	0.7243	0.7234	0.8098
128	256	0.2	0.9792	0.7005	0.6991	0.7929	0.9830	0.7225	0.7218	0.8091
256	256	0.1	0.9795	0.7038	0.7034	0.7956	0.9799	0.7142	0.7135	0.8025
256	256	0.2	0.9803	0.7069	0.7060	0.7978	0.9807	0.7124	0.7118	0.8017
256	512	0.1	0.9777	0.6212	0.6204	0.7398	0.9774	0.6623	0.6614	0.7670
256	512	0.2	0.9765	0.6825	0.6816	0.7802	0.9779	0.6975	0.6967	0.7907

Table 5.6: Evaluation with the SciVQA validation dataset (1680 questions) on the Qwen2.5-VL 7B model that was fine-tuned on the combined training datasets of SciVQA, SpiQA, and ArXivQA with the different hyperparameters LoRA rank, alpha, and dropout. The learning rate was always 2×10^{-4}, and the learning rate scheduler was linear. The metrics are the F1-Scores of BERTScore, ROUGE-1, ROUGE-L, and their average.

5.2.4 Fine-Tuning with Other Datasets

As explained in Subsection 4.4.2, for the fine-tuning of the Qwen2.5-VL models on the combined dataset, hyperparameter tuning was again performed by fine-tuning 7B models on the combined dataset with different training parameters. As visible in Table 5.6, the best results were achieved by training for two epochs with a rank of 64, an alpha of 64, and a dropout of 0.1. These parameters were used to fine-tune both the 7B and the 32B variants. The results of this fine-tuning are visible in Table 5.1. They show that adding more training data from the SpiQA (Pramanick et al., 2024) and ArXivQA (Li et al., 2024) datasets resulted in scores of 0.799 for the Qwen2.5-VL 7B model and 0.827 for the 32B model. It therefore reduced the performance in comparison to the models fine-tuned only on the SciVQA dataset. The reason for that is not clear, and further studies are needed to explain the performance drop. A possible hypothesis

Question Type	Combined Dataset	SciVQA
Infinite		
visual	0.6572	0.6878
non-visual	0.7879	0.7877
Yes/No		
visual	0.8110	0.8527
non-visual	0.8471	0.8611
Multiple-Choice		
visual	0.7931	0.8227
non-visual	0.8105	0.7821
Unanswerable	0.9653	0.9669
Overall	**0.8103**	**0.8230**

Table 5.7: Comparison of the average F1-Scores of the ROUGE-1, ROUGE-L, and BERTScore metrics by question type between the Qwen2.5-VL 7B model that was fine-tuned on the combined dataset and the 7B model exclusively fine-tuned on the SciVQA dataset. The evaluation was done on the SciVQA validation dataset (1680 questions). Each question type contains 240 questions.

could be that especially ArXivQA covers a wider range of scientific fields than the SciVQA dataset[1] (Li et al., 2024; Li and Tajbakhsh, 2023; Karishma et al., 2023). This could mean that the ArXivQA dataset could potentially reduce the performance on the SciVQA test split. In Subsection 5.3.1, a cross-validation across the SciVQA, ArXivQA, and SpiQA datasets is performed. It showed that fine-tuning solely on ArXivQA or SpiQA does not greatly improve the results in comparison with a Qwen2.5-VL 7B model that did not receive fine-tuning. Training on the whole ArXivQA dataset even harms the performance. This supports the hypothesis that the domains of ArXivQA and SpiQA are too different from the SciVQA dataset to improve the results.

To further investigate the poorer results of the fine-tuning on the combined dataset, we compare them with the scores of the 7B model fine-tuned solely on SciVQA in Table 5.7. Using only SciVQA as training data led to a better score on most question types. On non-visual infinite answer set questions, the results of the model trained on the combined dataset are minimally better than those of the model fine-tuned exclusively on SciVQA. For non-visual multiple-choice questions, the score of the model trained on the combined dataset is, with a difference of 0.0284, notably higher. However, fine-tuning on the combined dataset leads to a 0.0296 worse score than fine-tuning solely on SciVQA for visual multiple-choice questions. Possibly, the combined dataset contains mostly non-visual questions, and therefore, the results on visual questions are worse.

5.2.5 Fine-Tuning with Manually Labeled Infinite Answer Set Questions

As described in Section 4.5, all infinite answer set questions in the training split of the SciVQA dataset were manually labeled, and the updated training data was used to fine-tune a Qwen2.5-VL 32B model. To assess whether manually labeled training data

Question Type	Orig. data	Man. labeled data
Infinite		
visual	0.5458	0.5542
non-visual	0.7500	0.7833
Yes/No		
visual	0.8000	0.7875
non-visual	0.8167	0.8125
Multiple-Choice		
visual	0.7583	0.7375
non-visual	0.6958	0.6958
Unanswerable	0.9792	0.9542
Overall	**0.7637**	**0.7607**

Table 5.8: Manual evaluation of results obtained from fine-tuning two Qwen2.5-VL 32B models for two epochs, first on the original SciVQA training data, and secondly on the SciVQA training data with manually labeled infinite answer set questions. The table shows the fraction of correctly answered questions on the SciVQA validation dataset (1680 questions) per question type. Each question type contains 240 questions.

actually improved the accuracy of the answers, and given that parts of the validation and test splits of the SciVQA dataset also contain inaccuracies, the results were manually evaluated on the validation split. The results are compared with the 32B model trained on the original SciVQA training split in Table 5.8. Surprisingly, overall, the accuracy of the model fine-tuned on the manually labeled data is reduced slightly by 0.3 percentage points compared to the model fine-tuned on the original SciVQA data. But the accuracy on only the infinite answer set questions is noticeably improved by using the manually labeled training data. On visual questions, the accuracy is improved by 0.84 percentage points, and on non-visual questions, even by 3.33 percentage points, which is quite a substantial increase. These improvements are expected since the training data for these question types were manually reviewed and improved. However, the results on all other question types got worse when adapting the infinite answer set training data. Especially on unanswerable questions, the accuracy drops by 2.5 percentage points from 0.9792 to 0.9542. This is even more surprising, given that 73 questions that could not be certainly answered but were labeled as answerable were removed during the manual labeling. Therefore, using the original training data, the model was sometimes encouraged to try to answer unanswerable questions. Notably, the accuracy on the visual questions is impaired stronger than the accuracy on the non-visual questions. For yes/no questions, the accuracy dropped by 1.25 percentage points for visual questions and only by 0.42 percentage points for non-visual questions. On multiple-choice questions, the difference is even larger, with a 2.08 percentage point decrease on visual questions, while the accuracy on non-visual questions was the same for both models. The reason for this pattern is unknown.

Fine-tuned modules	Average F1-Score	Training Time	#Trained Parameters
Qwen2.5-VL 7B			
W_q, W_v	0.8237	6.5h	20M
W_q, W_v, W_k, W_o	0.8314	6.4h	40M
all linear modules	0.6717	11.7h	207M
Qwen2.5-VL 32B			
W_q, W_v	0.8460	14.9h	67M
W_q, W_v, W_k, W_o	0.8390	16.8h	134M
all linear modules	0.6810	21.5h	583M

Table 5.9: The average of the F1-Scores of BERTScore, ROUGE-1, and ROUGE-L on the validation split of the SciVQA dataset and the number of trained parameters and the training time when adapting different modules of the Qwen2.5-VL 7B and 32B models. The models are all fine-tuned on the SciVQA training dataset for two epochs.

5.2.6 Fine-Tuning Different Target Modules

The goal of this experiment was to test whether fine-tuning different parts of the VLM than proposed by the LoRA paper by Hu et al. (2022) could improve the results. As shown in Table 5.9 for the Qwen2.5-VL 7B model, adapting the weight matrices W_q, W_v, W_k, and W_o with LoRA actually outperformed fine-tuning only W_q and W_v. Surprisingly, the measured training time was also slightly lower. This is unexpected since the number of trained parameters is doubled since twice as many parameters are added by LoRA. Still, for this setting where the four projection modules of the LLM in the 7B model receive fine-tuning, only 0.48% of all parameters receive weight updates. Therefore, the runtime is dominated by the forward and backward passes through the frozen parameters. The runtime difference is therefore likely caused by minor, uncontrollable variations in the training environment.

Adding LoRA adapters to all linear layers and therefore also adapting layers in the vision transformer and in the projector leads to much worse results. For this setup, the training times also increased significantly.

For the 32B model, adapting the W_q and W_v modules still leads to the best results, confirming the choice of this training setting. Noticeably, the training times are also as expected for the 32B model, since adapting the W_q and W_v modules took the least time.

5.3 Ablation Studies

In this section, further experiments were conducted. Subsection 5.3.1 performs a cross-validation between the SciVQA, ArXiv, and SpiQA datasets to further explain the worse results when training on a combination of these datasets in comparison to training only on the SciVQA dataset. As this experiment showed that using fewer samples from the ArXivQA dataset leads to better results on the SciVQA test set, Subsection 5.3.2 investigates the relationship between the number of training samples from the SciVQA dataset and the performance. Subsection 5.3.3 provides the manual evaluation results of the fine-tuned 32B model, detailing the accuracy for each figure type.

5.3.1 Cross-Validation with Different Datasets

As described in Section 4.4, the ArXivQA and SpiQA datasets were also individually used to fine-tune a model each. Then, cross-validation with the different trained models across the test sets of the different datasets was performed. To perform a fair comparison, hyperparameter tuning was performed for the models trained on the ArXivQA and SpiQA datasets in the same way as it was done for the SciVQA dataset in Section 4.3. The results are visible for ArXivQA in Table 5.10 and for SpiQA in Table 5.11. The decision which hyperparameters would be used was again based on the average of the F1-Scores of the BERTScore, ROUGE-1, and ROUGE-L. For ArXivQA, this meant to fine-tune a Qwen2.5-VL 7B model with a LoRA rank of 16, an alpha of 32, and a dropout of 0.1 for one epoch. For SpiQA, a rank of 128, an alpha of 256, and a dropout of 0.1 led to the same maximum average F1-Score of 0.6959 for both training durations of one and two epochs. In the cross evaluation, the model that was trained for two epochs was used. The same hyperparameters were used to train a model for subsets of the two datasets, which contained 15120 questions each.

The trained models are evaluated on the test splits of all three datasets. For comparison, the base Qwen2.5-VL 7B model without fine-tuning is also added. Table 5.12 shows the results. As expected, for each test dataset, the model that was trained on the corresponding training dataset reaches the best results. On the SciVQA test split, the model trained on the full SpiQA dataset reaches a score of 0.694 and therefore outperforms the ArXivQA models, which only reached 0.676 when trained on the subset of ArXivQA. This makes sense since the ArXivQA dataset consists of multiple-choice questions, while the SpiQA dataset contains a mixture of question types, including infinite answer set, yes/no, but also multiple-choice questions. Therefore, the SpiQA dataset is overall more similar to the SciVQA dataset than the ArXivQA dataset. More surprisingly, the model that was trained on the full ArXivQA dataset performed 0.028 worse than the model trained on the subset. Furthermore, the Qwen2.5-VL model that did not receive fine-tuning also reaches, with a score of 0.666, a better result than the model trained on the complete ArXivQA dataset. Possibly, the fact that all multiple-choice questions in the ArXivQA dataset have only one correct answer inhibits the ability of the ArXivQA model to answer with multiple answers to the multiple-choice questions in the SciVQA dataset, where multiple answers can be correct. Analyzing the data revealed that the ArXivQA model responded only 22 times with multiple answer options, while the model without fine-tuning answered 197 times with multiple options. The SciVQA test split contains 162 multiple-choice questions where multiple answers are correct. Though with this hypothesis, one would expect that the model trained on the subset of ArXivQA also performs worse than the model without fine-tuning, but this is not the case. Potentially, one could speculate that the fine-tuning on 15120 samples was enough to improve the question answering abilities of the model, but to prevent the model from answering with more than one option, even though the system prompt states that multiple answers can be correct, more training samples are needed. Although the SpiQA score on the SciVQA test split is higher than the score of the model without fine-tuning, the difference of 0.028 is not large. This could further explain why

			Train Epochs: 1				Train Epochs: 2			
r	α	d	BERT	R-1	R-L	Avg.	BERT	R-1	R-L	Avg.
16	16	0.1	0.9435	0.5067	0.5015	0.6506	0.9556	0.5340	0.5306	0.6734
16	16	0.2	0.9526	0.5449	0.5410	0.6795	0.9320	0.4772	0.4718	0.6270
16	**32**	**0.1**	**0.9564**	**0.5477**	**0.5443**	**0.6828**	0.9266	0.4587	0.4542	0.6132
16	32	0.2	0.9545	0.5326	0.5284	0.6718	0.9497	0.5191	0.5156	0.6614
32	32	0.1	0.9511	0.5240	0.5200	0.6650	0.9542	0.5369	0.5328	0.6747
32	32	0.2	0.9420	0.5040	0.4984	0.6481	0.9485	0.5203	0.5163	0.6617
32	64	0.1	0.9531	0.5342	0.5307	0.6727	0.9500	0.5251	0.5215	0.6656
32	64	0.2	0.9452	0.5080	0.5050	0.6527	0.9476	0.5247	0.5220	0.6647
64	64	0.1	0.9541	0.5353	0.5328	0.6741	0.9544	0.5281	0.5253	0.6693
64	64	0.2	0.9439	0.5109	0.5075	0.6541	0.9539	0.5387	0.5356	0.6761
64	128	0.1	0.9537	0.5399	0.5361	0.6766	0.9543	0.5322	0.5294	0.6720
64	128	0.2	0.9562	0.5379	0.5349	0.6763	0.9556	0.5374	0.5351	0.6760
128	128	0.1	0.9565	0.5406	0.5373	0.6781	0.9533	0.5436	0.5394	0.6788
128	128	0.2	0.9543	0.5402	0.5365	0.6770	0.9530	0.5443	0.5416	0.6797
128	256	0.1	0.9533	0.5385	0.5351	0.6756	0.9380	0.4965	0.4924	0.6423
128	256	0.2	0.9545	0.5343	0.5312	0.6733	0.9541	0.5354	0.5323	0.6739
256	256	0.1	0.9450	0.5126	0.5099	0.6559	0.9354	0.4975	0.4942	0.6424
256	256	0.2	0.9540	0.5342	0.5313	0.6732	0.9433	0.5077	0.5038	0.6516
256	512	0.1	0.9455	0.4821	0.4790	0.6355	0.9453	0.5104	0.5072	0.6543
256	512	0.2	0.8986	0.3258	0.3203	0.5149	0.9075	0.3906	0.3860	0.5613

Table 5.10: Evaluation with the SciVQA validation dataset (1680 questions) on the Qwen2.5-VL 7B model that was fine-tuned on the ArXivQA training dataset with the different hyperparameters LoRA rank, alpha, and dropout. The learning rate was always 2×10^{-4}, and the learning rate scheduler was linear. The metrics are the F1-Scores of BERTScore, ROUGE-1, ROUGE-L, and their average.

the model that was fine-tuned on the combination of all three datasets performed worse than the model fine-tuned only on SciVQA in Subsection 5.2.4.

For the ArXivQA test split, the fraction of correctly answered questions can easily be determined since the test split only contains multiple-choice questions with one correct answer. Therefore, the fraction of exactly matching answers is identical, apart from a few exceptions, to the fraction of correctly answered questions. The few exceptions were manually verified by checking all responses that did not contain only a single character. The result shows that the SciVQA model slightly outperforms the SpiQA model. Here again, the model trained only on the subset of the SpiQA dataset performs a bit better than the model fine-tuned on the full SpiQA dataset. This does not align with the expectation that more training data would lead to better results. For the ArXivQA test split, both the fine-tuning on the SciVQA dataset as well as on the SpiQA dataset improved the results.

Interestingly, on the SpiQA test split, the ArXivQA model reaches significantly better F1-Scores than the SciVQA model, even though the ArXivQA model only contains multiple-choice questions. Still, all fine-tuned models reach substantially higher scores

			Train Epochs: 1				Train Epochs: 2			
r	α	d	BERT	R-1	R-L	Avg.	BERT	R-1	R-L	Avg.
16	16	0.1	0.9380	0.5323	0.5287	0.6663	0.9361	0.5328	0.5288	0.6659
16	16	0.2	0.9408	0.5537	0.5503	0.6816	0.9385	0.5468	0.5424	0.6759
16	32	0.1	0.9406	0.5559	0.5522	0.6829	0.9363	0.5405	0.5365	0.6711
16	32	0.2	0.9369	0.5449	0.5407	0.6741	0.9342	0.5357	0.5316	0.6672
32	32	0.1	0.9394	0.5520	0.5481	0.6798	0.9353	0.5409	0.5365	0.6709
32	32	0.2	0.9385	0.5490	0.5453	0.6776	0.9352	0.5385	0.5343	0.6693
32	64	0.1	0.9394	0.5590	0.5553	0.6846	0.9374	0.5480	0.5446	0.6767
32	64	0.2	0.9410	0.5622	0.5583	0.6872	0.9389	0.5539	0.5493	0.6807
64	64	0.1	0.9400	0.5607	0.5572	0.6860	0.9364	0.5555	0.5512	0.6810
64	64	0.2	0.9390	0.5641	0.5604	0.6879	0.9357	0.5505	0.5466	0.6776
64	128	0.1	0.9351	0.5516	0.5481	0.6783	0.9334	0.5498	0.5458	0.6763
64	128	0.2	0.9401	0.5561	0.5525	0.6829	0.9383	0.5534	0.5498	0.6805
128	128	0.1	0.9399	0.5662	0.5629	0.6897	0.9378	0.5574	0.5535	0.6829
128	128	0.2	0.9395	0.5609	0.5575	0.6860	0.9382	0.5545	0.5510	0.6812
128	**256**	**0.1**	0.9405	0.5751	0.5720	0.6959	**0.9386**	**0.5767**	**0.5725**	**0.6959**
128	256	0.2	0.9394	0.5554	0.5523	0.6824	0.9383	0.5522	0.5483	0.6796
256	256	0.1	0.9385	0.5618	0.5590	0.6864	0.9345	0.5486	0.5444	0.6758
256	256	0.2	0.9385	0.5523	0.5494	0.6800	0.9372	0.5590	0.5551	0.6838
256	512	0.1	0.9338	0.5453	0.5426	0.6739	0.9346	0.5562	0.5518	0.6809
256	512	0.2	0.9362	0.5393	0.5355	0.6703	0.9358	0.5453	0.5408	0.6740

Table 5.11: Evaluation with the SciVQA validation dataset (1680 questions) on the Qwen2.5-VL 7B model that was fine-tuned on the SpiQA training dataset with the different hyperparameters LoRA rank, alpha, and dropout. The learning rate was always 2×10^{-4}, and the learning rate scheduler was linear. The metrics are the F1-Scores of BERTScore, ROUGE-1, ROUGE-L, and their average.

on the SpiQA test split than the model that did not receive fine-tuning. This again shows the effectiveness of fine-tuning. Although on the SpiQA test split, a larger dataset size improved the results, the differences between the models trained on the subset versus the models trained on the full dataset are, with 0.003, very small for both SpiQA and ArXivQA.

5.3.2 Influence of the Training Dataset Size

In Subsection 5.3.1, the results showed that, against the expectation, fine-tuning with 15K samples from the ArXivQA dataset led to better results on the SciVQA test split than fine-tuning with 57K samples. The same was found for the training with the SpiQA dataset when evaluating on the ArXivQA dataset. Motivated by this surprise, this experiment tests the relation between the training dataset size and the performance for the SciVQA dataset. Therefore, a number of smaller datasets are subsampled from the original SciVQA training dataset with 15120 samples. This subsampling is not done randomly, instead, always the first x samples of the 15120 samples are used to create the subset with x entries. Each subsplit still contains a balanced amount of questions from each question type. This is done to ensure that, for example, the model trained

Model	**SciVQA Test Split Avg. F1**	**ArXivQA Test Split fraction correct ans.**	**SpiQA Test Split Avg. F1**
7B zero-shot	0.6655	0.6069	0.4786
SciVQA			
#train samples: 15,120	**0.8206**	0.6660	0.6183
ArXivQA			
#train samples: 15,120	0.6762	0.7283	0.6853
#train samples: 57,339	0.6482	**0.7312**	0.6885
SpiQA			
#train samples: 15,120	0.6792	0.6595	0.7378
#train samples: 35,313	0.6938	0.6507	**0.7409**

Table 5.12: The average of the F1-Scores of BERTScore, ROUGE-1, and ROUGE-L on the test split of the SciVQA and SpiQA datasets and the fraction of correctly answered questions on the ArXivQA test split. The models are all Qwen2.5-VL 7B models, where one did not receive fine-tuning and was prompted in a zero-shot fashion with the refined prompt template, while the others were each fine-tuned on the train split of the SciVQA, ArXivQA, and SpiQA datasets, respectively. For ArXivQA and SpiQA, a subset with 15120 samples was also used to train a model.

on 756 samples did not perform better than the model trained on 1512 samples simply because the smaller subset happened to contain unusually high-quality data that was not included in the larger subset. The red line in Figure 5.2 shows the results of the experiment. Overall, as expected, performance improves with a growing number of training samples. However, there is one exception, as the model fine-tuned for two epochs on 378 samples reaches a better score than the model fine-tuned on 756 samples.

The fact that each model was fine-tuned for two epochs with different dataset sizes means that the model fine-tuned on the full SciVQA dataset received many more gradient update steps than the model that was fine-tuned on only 378 samples. Therefore, there is the possibility that not the higher number of training samples, but the increase in training steps led to the performance increases. To rule this out, the experiment was repeated, but instead of training for two epochs per subset, the fine-tuning was done for 7560 training steps, which equals two epochs of fine-tuning with the full dataset, and an effective batch size of four. The results are presented by the blue line in Figure 5.2. As visible in the graph, increasing the dataset size leads to a strict increase in the performance. Therefore, it is actually the larger number of examples rather than the larger number of training steps that increases performance. When comparing the performance of a model that was trained for 7560 steps with the corresponding model that received fine-tuning for two epochs, for smaller training dataset sizes, the results of fine-tuning for 7560 steps are worse. This might result from the model overfitting due to the small amount of training examples, which are repeatedly shown to the model. Again, this does not apply to the model fine-tuned on 756 samples for two epochs, which seems to be an outlier.

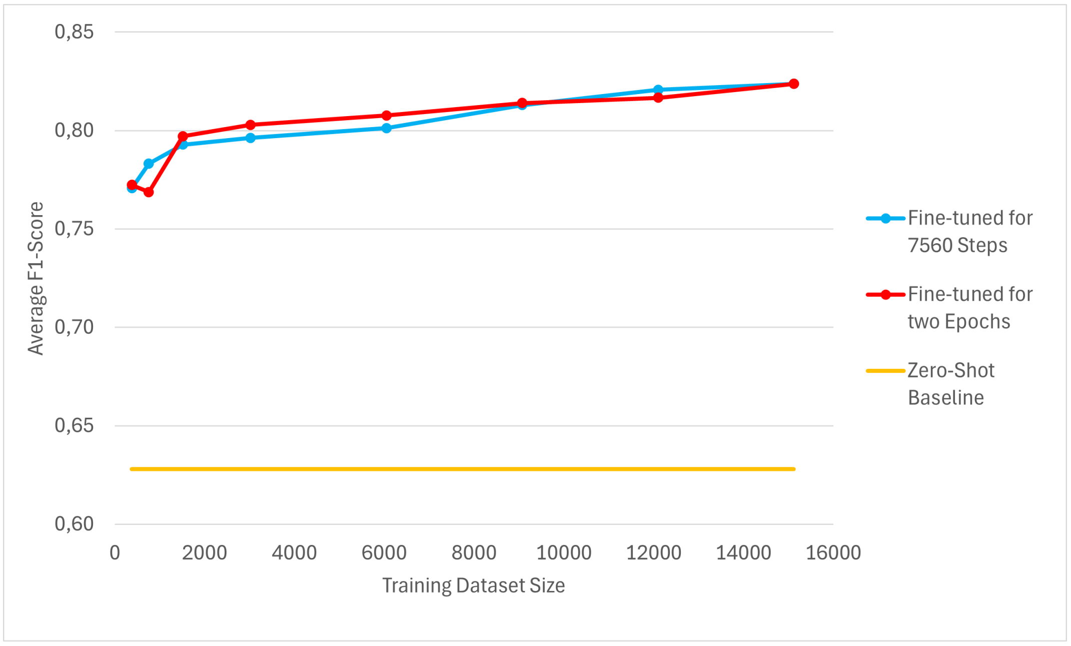


Figure 5.2: Average of the F1-Scores for BERTScore, ROUGE-1, and ROUGE-L across different sizes of training datasets. The red line represents the performance on the SciVQA validation dataset of a Qwen2.5-VL 7B model that was fine-tuned for two epochs on a subset of the SciVQA training dataset with the specified size. The blue line shows the performance of a Qwen2.5-VL 7B model, which was fine-tuned for 7560 steps (equal to two epochs with 15120 samples) on datasets of the specified size. Apart from the different dataset sizes, the fine-tuning setup for both models was as described in Section 4.3. The orange line shows the performance of the Qwen2.5-VL 7B model without fine-tuning on the SciVQA validation dataset in a zero-shot setting.

Both curves in Figure 5.2 still show notable performance improvements for the dataset size increase from 12K to 15K samples. This suggests that even higher performance might be achievable with additional samples from the same domain. As shown in Subsection 5.2.4, SpiQA and ArXivQA are too different from the SciVQA dataset to lead to such improvements, although they also consist of charts from scientific papers. Comparing the results of the fine-tuned models with the zero-shot baseline model that was not fine-tuned again suggests that even fine-tuning on very small amounts of training data leads to an increase in performance, although, as discussed in Subsection 5.2.1, the automatic metrics in the zero-shot setting are not fully reliable.

5.3.3 Performance per Figure Type

Table 5.13 shows the accuracy of the Qwen2.5-VL 32B model that was fine-tuned for two epochs (see Subsection 5.2.3) for each figure type. The highest accuracy was achieved for figures of neural networks, followed by architecture diagrams. The category with by far the most samples, namely line charts, only reaches an accuracy of 0.743, which is lower than the accuracy on most other figure types. Scatter plots and graphs also seem to be rather challenging for the model. Due to the small number of samples, it is difficult to draw meaningful conclusions from the results on other chart types. Nonetheless, generally, charts where specific values need to be read from continuous scales seem to be harder. The distribution of figure types in the validation set is similar to the distribution

Figure Type	**Accuracy**	**#samples**
Line Chart	0.7425	1099
Tree	0.8286	105
Scatter Plot	0.7429	70
Architecture Diagram	0.8750	56
Neural Networks	0.8929	56
Graph	0.7500	56
Pie Chart	0.8163	49
Bar Chart	0.8163	49
Confusion Matrix	0.7857	42
Compound Of Different Types	0.7500	28
Box Plot	0.5714	14
Histogram	0.7143	14
Other	0.8333	42
Total	**0.7637**	**1680**

Table 5.13: Manually evaluated accuracy for the Qwen2.5-VL 32B model that was fine-tuned with the parameters described in Section 4.3 and Subsection 5.2.3 for the different figure types on the SciVQA validation set.

in the training dataset. Therefore, it should be noted that the accuracy does not seem to be directly correlated with the number of training examples. For instance, only 3.24% of the training samples are bar charts, but the accuracy on them is still 0.816, and therefore, better than the accuracy on line charts. Possibly this is the case because bar charts were used in the pretraining of the Qwen2.5-VL models. Bai et al. (2025) explicitly mentions bar charts as part of the training data but not line charts. Although this does not mean that line charts were not used, it could be an indicator that bar charts account for a larger portion of the training data used by Bai et al. (2025).

5.4 Comparison with other SciVQA Participants

The final results of the shared task can be seen in Table 5.14. These results show that Bhat et al. (2025) achieved first place in the SciVQA competition with an average F1-Score of 0.8647. The last improvement they employed was to add 2500 human-annotated samples from the ChartQA dataset (Masry et al., 2022) to their training data and to postprocess their answers. Together, these changes led to an improvement of 0.49%. Unfortunately, they do not provide any experimental results on how much improvement can be attributed to the addition of external training data. Nonetheless, it shows that the approach of adding additional training data can be beneficial, although it could also indicate that the number of additional samples should not be too large in order to prevent the model from adapting the style and domain of the other dataset. Also, the

System		ROUGE-1 (F1)	ROUGE-L (F1)	BERTScore (F1)	Avg. F1
Human		**0.8291**	**0.8285**	**0.9826**	**0.8801**
Baseline		0.7062	0.7055	0.9756	0.7957
ExpertNeurons	(Bhat et al., 2025)	**0.8049**	**0.8043**	**0.9849**	**0.8647**
THAii_LAB	(Ventura et al., 2025)	0.7899	0.7892	0.9839	0.8543
Coling_UniA	(Jaumann et al., 2025)	0.7862	0.7856	0.9817	0.8512
Ours	(Schleid et al., 2025)	0.7631	0.7621	0.9831	0.8361
Infyn	(Movva and Marupaka, 2025)	0.7350	0.7345	0.9787	0.8161
Soham Chitnis		0.7057	0.7052	0.9811	0.7709
psr123		0.6068	0.6056	0.9587	0.7237

Table 5.14: F1-Scores of all participants in the SciVQA shared task as well as the baselines (Borisova et al., 2025).

quality of the samples that they used might have been superior to the additional samples from the SpiQA and ArXivQA datasets used in this thesis, since Bhat et al. (2025) only used human-annotated samples.

Ventura et al. (2025) reported the second-best results. They also fine-tuned Qwen2.5-VL models, but instead of experimenting with expanding the training dataset, they focused on the design of the prompt. Specifically, they added a CoT segment to the prompt, which motivated the model to step-by-step analyze the chart and extract the relevant elements and information. The prompt is also dynamically generated based on the metadata provided by the SciVQA dataset. This includes telling the model the figure type and the number of subfigures. Furthermore, depending on whether the question is a visual or a non-visual question, the model was instructed to pay attention to visual aspects like color, position, etc., or to focus on numerical and textual values. Furthermore, for binary questions, the model was explicitly asked to only answer with yes or no. This extent of dynamic prompting is problematic since, in an actual use case of the system, the information of the exact question type, including whether it is a visual or a non-visual question, and the figure type and the number of subfigures, would most likely not be provided. Therefore, this thesis does not use this metadata to generate the prompt. Nonetheless, Ventura et al. (2025) were able to score the second-best score with 0.8543, underlining the impact of the prompt template. Interestingly, they reported better results when fine-tuning the Qwen2.5-VL 7B model than when fine-tuning the larger 72B variant. This is unexpected and does not align with our results, where fine-tuning the 32B variant always outperformed the results of fine-tuning the 7B variant. We would have expected that this increase in performance with larger model size continues, and therefore, we would assume the best results for the 72B model.

The third ranking team, Jaumann et al. (2025), showed that a high performance can be reached by specifically selecting few-shot examples that are similar to the question at hand, through the calculation of the cosine similarity on embeddings of the question-image pairs. With only one or two examples per question and without fine-tuning, an average F1-Score of 0.8512 was reached. This is very different from our results in Subsection 5.2.2, where the usage of a one-shot example led to a decreased performance for the Qwen2.5-VL 7B and 32B models and only to minor improvements on GPT-4o mini. Possibly, the smaller Qwen models are just not capable of exploiting the one-shot example. Notably, Jaumann et al. (2025) used the InternVL3 model (Zhu et al., 2025) and the Pixtral-Large-2411[2] model with 78B and 124B parameters, respectively. The

2. https://huggingface.co/mistralai/Pixtral-Large-Instruct-2411

substantially larger model size, together with the more refined approach to selecting the examples, could explain the difference in the results.

The system in fifth place mainly employed prompt engineering and queried the model in two steps for each question (Movva and Marupaka, 2025). The first step requests the model to extract the relevant information for the question, and the second query instructs the model how to answer the question with the extracted information. As the second ranking system by Ventura et al. (2025), the prompt template uses the metadata from the dataset and fits the prompt to the figure and question type. They did not perform fine-tuning, but tested four different base models and found that InternVL3 reached the best results. Qwen2.5-VL only reached the second-best scores. Interestingly, the performance improvement by the two-step querying was much larger for the Qwen2.5-VL model than for the InternVL3 model. However, the performance of InternVL3 was already 8.85% better than Qwen2.5-VL in the single-step zero-shot setting. With the two-step inference, InternVL3 outperforms Qwen2.5-VL by 4.19%. Nonetheless, the fact that the first ranking participants (Bhat et al., 2025) also report superior results using InternVL3 in comparison to Qwen2.5-VL, with the InternVL3 models being only slightly larger than the Qwen2.5-VL models against which they are compared, suggests that InternVL3 is better suited for the task of visual question answering on charts.

Altogether, the results of the other participants confirm the finding from this thesis that fine-tuning leads to substantial performance increases. Furthermore, based on their results, CoT prompting improves the answer quality and, if such metadata are available, specifically tailoring the prompt to the current chart and question type also leads to better results. The usage of additional training data, which was extensively studied in this thesis but could not lead to performance increases, was not thoroughly tested by the other participants. Lastly, Bhat et al. (2025) showed that providing the model with more context from the text of the paper allows the model to improve the quality of the responses.

As visible in Table 5.14, compared to the SciVQA baseline, our fine-tuned 32B model outperforms it by 5.08%. However, the human baseline is still 5.26% better than our results. It is also noteworthy that multiple models, including ours, surpassed the F1-Score of the BERTScore metric of the human baseline. This again puts the feasibility of the BERTScore as an evaluation metric in question.

6 Conclusion

This chapter summarizes the results of the experiments that were conducted to answer the research questions of this thesis. For each research question, in addition to the summarized results, limitations and possible future work are presented.

6.1 Research Question 1

What is a good prompt template for VQA on scientific charts in the context of the SciVQA shared task, and can a one-shot example improve the performance?
The prompt template plays a vital role in the quality of the results. Together with the results of the other participants, it can be seen that more context and prompts that specifically instruct the model how to solve the target question increase the performance. The inclusion of metadata to adapt the prompt even closer to the question and figure type supports this through further improvements. However, the reliance on such metadata limits the potential applications of the system, as this metadata is not always available. In addition, although all models employed in this thesis gained in accuracy through the prompt engineering, the effect depended strongly on the underlying model.

Using one-shot examples led to mixed results in our experiments. For the Qwen2.5-VL models, manual evaluations showed that although the performance on the automatic metrics was increased, the actual accuracy of the generated answers dropped. Only for the GPT-4o mini model, the accuracy increased slightly, and this improvement was mainly driven by gains on the unanswerable questions while losing accuracy on infinite answer set and yes/no questions. Therefore, the value of a one-shot sample, which is not targeted to the specific question, is limited, especially when using smaller models. Jaumann et al. (2025) nonetheless showed that few-shot examples that are specifically selected for the question at hand can greatly improve the performance on larger models.

Limitations
In the zero-shot and one-shot settings, the models do not adhere consistently to the very short answer format of the ground-truth answers. Since the automatic evaluation metrics are biased against long answers, this prevents an accurate evaluation of whether

the questions are answered correctly. This gets especially evident in the evaluation of the effect of prompt engineering for the Qwen2.5-VL 32B model. The automatic metrics suggest that the 32B model is the weakest model, while the manual evaluation proves it to be the most accurate. This shows that, especially for the zero-shot and one-shot inference, the automatic evaluation metrics are only of limited value as the models do not consistently adhere to the short answer format. This makes testing different prompt templates and one-shot examples very difficult, since performing a manual evaluation for each attempt is not feasible due to its time-intensive nature. A more general problem with prompt engineering is that even slight changes in the template could lead to further improvements in the results, and one cannot be sure if the optimal prompt was found.

Future Work
Considering the success of Jaumann et al. (2025), using one-shot examples that are specifically retrieved for the question at hand could improve the results. However, these results should be evaluated either manually or with a different automatic metric. Potentially, the LLM-as-a-judge evaluation technique could help to evaluate the answers without such a strong focus on the answer format. However, this can also introduce further uncertainty in the evaluation process. Another option would be to try to prompt the models to answer with a JSON that contains the final answer and further explanations or thought processes as separate properties. That would allow us to use the automatic metrics only on the short final answers. But just requesting JSON outputs in the prompt does not guarantee that the model consistently adheres to the JSON format (Salinas and Morstatter, 2024). However, it is possible to constrain the tokens that can be generated by the model to enforce a valid JSON output, as is done by OpenAI's Structured Outputs functionality[1]. Nonetheless, Salinas and Morstatter (2024) report worse performance using this enforced structured output compared to both inference without a specified format and inference with a request for a JSON output in the prompt. Therefore, careful experiments would be needed to prevent a degradation of the answer quality.

6.2 Research Question 2

How does fine-tuning a VLM impact the performance of VQA on scientific charts, and what hyperparameter configurations yield the best results?
Fine-tuning Qwen2.5-VL models using LoRA and quantization led to the best results achieved in this thesis. It substantially outperformed the GPT-4o mini model, which was queried in a one-shot setting, both in terms of the automatic metrics and when evaluated manually. The manual evaluation also proved that the fine-tuning not only adapted the model to return the answers in the correct format, but that the model actually improved in the quality and correctness of the answers. The model especially excelled on unanswerable questions, where it answered 97.9% of the questions correctly. In comparison, the GPT-4o mini model queried with a one-shot example only gave the correct answer 77.1% of the time for this question type. The ability to recognize unanswerable questions is especially important if such VQA systems were used to create meta-analysis, since hallucinations could lead to wrong results and, in the extreme case, e.g., to false conclusions about certain treatments in the medical field.

1. https://openai.com/index/introducing-structured-outputs-in-the-api/

As the original LoRA paper proposed, we could confirm that targeting the query and value projection matrices in the attention modules led to the best results. Furthermore, it was determined that even with a dropout of 0.2 and a learning rate of 2×10^{-4}, the optimal performance on the SciVQA dataset is reached after only two epochs of fine-tuning.

Hyperparameter tuning was performed for the SciVQA, the ArXivQA, and the SpiQA datasets individually and for the combination of all three. The fact that the results of these hyperparameter tunings led to rather different parameters shows the importance of performing hyperparameter tuning for effective fine-tuning.

Limitations

A key limitation of the fine-tuned model lies in its performance on visual questions with an infinite answer set. For such questions, the manual evaluation showed that the model frequently fails to return the exact value of a target datapoint, often producing approximate answers that are close but fall outside the acceptable error margin.

Future Work

In order to address the substantially worse results on the visual infinite answer set questions, one could try to gain further insight by fine-tuning a model only on this question type to find out if the other question types might inhibit the ability of the model to adapt to the visual infinite answer set questions.

In general, the results might be further improved by changing the base model. Although Ventura et al. (2025) reported worse results when using the larger Qwen2.5-VL 72B model versus the 7B variant, we found that the 32B model consistently outperformed the 7B model after fine-tuning. Therefore, fine-tuning the 72B variant could lead to further improvements. The use of InternVL3 (Zhu et al., 2025) could also further improve the results, as reported by other participants of the SciVQA shared task (Bhat et al., 2025; Movva and Marupaka, 2025). Another very recent model, which might be worth trying, is Qwen3-VL[2], the successor of Qwen2.5-VL.

6.3 Research Question 3

Does fine-tuning on a combination of the SciVQA dataset with similar datasets, or on a manually improved version of the SciVQA dataset, enhance the answer quality of the model?

Adding more training data from other datasets did not lead to better results. Although the used datasets, ArXivQA and SpiQA, both contain charts extracted from scientific papers, slight domain shifts, such as differences in the research area or question phrasing, may inhibit the model's ability to transfer learned concepts effectively. Instead of increasing the performance, the performance was reduced, and the cross-validation showed that using a smaller number of training samples from these datasets sometimes even led to better results. Nonetheless, ablation studies indicated that increasing the number of training samples that are from the domain of the SciVQA dataset would probably result in better performance.

In addition, the manual enhancement of the training data did not lead to overall performance gains. Improving the quality of the infinite answer set questions enables the model to perform better on this question type, but worsens the results on the other.

2. https://huggingface.co/Qwen/Qwen3-VL-235B-A22B-Instruct

Limitations
This thesis only tested the ArXivQA and the SpiQA datasets as additional sources for training data. Furthermore, it was not tried to use only a small amount of their samples. Other datasets, possibly even with human-annotated samples or with samples that more closely resemble the SciVQA dataset in both the domain and the question types, might lead to different results.

Future Work
Bhat et al. (2025) report slight performance increases that might be attributable to the inclusion of 2500 human-annotated samples from the ChartQA dataset. This could be a hint that a smaller number of additional samples with a focus on the quality of the samples could improve the results. The improved results for the infinite answer set question type motivate a Mixture-of-Experts-based approach. One could fine-tune a model for infinite answer set, yes/no, and multiple-choice questions each, with an additional model that predicts the question type of a question at hand and forwards the question to the corresponding model. This might enable each model to focus exclusively on one single question type, possibly leading to overall better results.

Appendices

A
Appendix

A.1 Baseline Prompt Template

In this section, Figure A.1 and Figure A.2 provide the baseline prompt templates. The baseline prompt only provides basic information on how to format the answer, along with the question and the image.

Baseline System Prompt

You will be given an image and a question that you should answer.
If you are sure that you do not have enough information to answer the question answer with: 'It is not possible to answer this question based only on the provided data.'
If it is a multiple choice question, list only the letter of the correct answers in the order they are given without spaces between them.

Figure A.1: The baseline system prompt.

A.2 Refined Prompt Templates

This section contains the final system prompt in Figure A.3 and the user prompt used in the zero-shot experiment and to fine-tune the models in Figure A.4. Figure A.5 shows the user prompt with an example used for the one-shot experiment.

Baseline User Prompt

```
This is the Question:
{{ question }}
{% if answer_options %}
You have the following answer options to choose from. Multiple answers
may be correct. List only the letter of the correct answers in the order they
are given without spaces between them.
Answer Options:
{{ answer_options }}
{%endif%}
<image>
```

Figure A.2: The baseline user prompt.

Refined System Prompt

```
You are an expert data analyst. You will be given an image of a chart and a
question.
You will answer the question based on the image of the chart.
If you are sure that you do not have enough information to answer the
question answer with: 'It is not possible to answer this question based
only on the provided data.'

Output Format:
Answer:
```

Figure A.3: System Prompt used for fine-tuning the Qwen2.5-VL models, as well as for the zero-shot and one-shot inference.

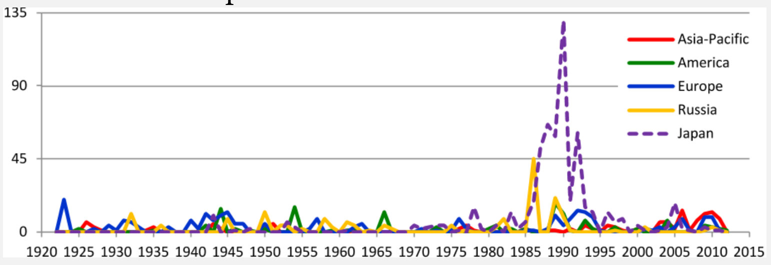
Refined User Prompt

```
Here is the caption of the image:
{{ caption }}
This is the Question:
{{ question }}
{% if answer_options %}
You have the following answer options to choose from. Multiple answers
may be correct. List only the letter of the correct answers in the order they
are given without spaces between them.
Answer Options:
{{ answer_options }}
{%endif%}
Give a short and precise answer:
<image>
```

Figure A.4: User Prompt for fine-tuning the Qwen2.5-VL models and for the zero-shot inference.

One Shot User Prompt

Here is an example:
The caption of the image is:
Figure 3. Annual frequency of USA being mentioned with Russia, Japan, and G20 countries
This is the Question:
{% if answer_options %}
The line of which color had highest annual mention frequency before 1925?
You have the following answer options to choose from. Multiple answers may be correct. List only the letters of the correct answers in the order they are given without spaces between them.
Answer Options:
A: Red line
B: Green line
C: Blue line
D: Yellow line
{% else %}
Which country, besides the USA, is mentioned the most frequently in the year 1990?
{%endif%}
Give a short and precise answer:



{% if answer_options %}
Answer: C
{% else %}
Answer: Japan
{%endif%}

This is the real query you should answer:
Here is the caption of the image:
{{ caption }}
This is the Question:
{{ question }}
{% if answer_options %}
You have the following answer options to choose from. Multiple answers may be correct. List only the letter of the correct answers in the order they are given without spaces between them.
Answer Options:
{{ answer_options }}
{%endif%}
Give a short and precise answer:
<image>

Figure A.5: User Prompt with one-shot example for the one-shot inference.

References

Grammarly and ChatGPT were used as support for the rephrasing of individual sentences and to search for scientific papers. ChatGPT was not used as an independent source, and no information provided by it was included in this thesis unless it could be verified and substantiated by the cited scholarly sources.

Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Esesn, Abdul A. S. Awwal, and Vijayan K. Asari. 2018. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. ArXiv:1803.01164, September. (Cited on page 5).

Rohan Anil et al. 2023. PaLM 2 Technical Report. ArXiv:2305.10403, September. (Cited on page 13).

Anthropic. 2024. Introducing Claude 3.5 Sonnet, June. Accessed August 24, 2025. (Cited on page 22).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations,* edited by Yoshua Bengio and Yann LeCun, (ICLR 2015). San Diego, CA, USA, May. (Cited on page 9).

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-VL Technical Report. ArXiv:2502.13923, February. (Cited on pages 2 sq., 19, 22 sqq., 52).

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems,* edited by T. Leen, T. Dietterich, and V. Tresp, vol. 13. MIT Press. (Cited on page 8).

Nagaraj N Bhat, Joydeb Mondal, and Srijon Sarkar. 2025. ExpertNeurons at SciVQA-2025: Retrieval Augmented VQA with Vision Language Model (RAVQA-VLM). In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 221–229. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 24, 52 sqq., 57 sq.).

Ekaterina Borisova, Nikolas Rauscher, and Georg Rehm. 2025. SciVQA 2025: Overview of the First Scientific Visual Question Answering Shared Task. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 182–210. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 1 sq., 19, 21, 53).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems,* 33:1877–1901. Curran Associates, Inc. (Cited on pages 3, 12, 16).

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research* 25 (70): 1–53. (Cited on page 12).

Gheorghe Comanici et al. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. ArXiv:2507.06261, July. (Cited on page 2).

Matthew A. Crane, Mickey Nguyen, Audrey Lam, Zackary D. Berger, Yannis M. Paulus, John A. Romley, and Ruth R. Faden. 2023. Figure accessibility in journals: analysis of alt-text in 2021–23. Publisher: Elsevier, *The Lancet* 402, no. 10419 (December): 2287–2289. (Cited on page 1).

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N. Fung, and Steven Hoi. 2023. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. *Advances in Neural Information Processing Systems* 36 (December): 49250–49267. (Cited on page 22).

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *Advances in Neural Information Processing Systems* 35 (December): 30318–30332. (Cited on pages 18, 27).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* edited by Jill Burstein, Christy Doran, and Thamar Solorio, 1:4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics, June. (Cited on pages 12 sq., 15, 35).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations.* International Conference on Learning Representations, (ICLR 2021). Austria: OpenReview.net, May. (Cited on page 14).

Petko Georgiev et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. ArXiv:2403.05530, December. (Cited on page 22).

Google. 2024. Gemini 2.0: Our latest, most capable AI model yet, December. Accessed August 24, 2025. (Cited on page 22).

Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. ArXiv:2407.21783, November. (Cited on pages 2, 12).

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. ChartLlama: A Multimodal LLM for Chart Understanding and Generation. ArXiv:2311.16483, November. (Cited on page 22).

Mark A. Hanson, Pablo Gómez Barreiro, Paolo Crosetto, and Dan Brockington. 2024. The strain on scientific publishing. *Quantitative Science Studies* 5, no. 4 (November): 823–843. (Cited on page 1).

Simon S. Haykin. 2009. Neural networks and learning machines. 3rd ed. New York: Prentice Hall. (Cited on pages 5 sq.).

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked Autoencoders Are Scalable Vision Learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 15979–15988. ISSN: 2575-7075. June. (Cited on page 15).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition,* 770–778, (CVPR 2016). Las Vegas, NV, USA: IEEE Computer Society, June. (Cited on pages 12, 14).

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. 2025. Rotary Position Embedding for Vision Transformer. In *Lecture Notes in Computer Science,* 289–305. Cham: Springer Nature Switzerland. (Cited on page 23).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, no. 8 (November): 1735–1780. (Cited on page 7).

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations.* ICLR 2022, April 25–29. (Cited on pages 17, 27, 31, 46).

Kung-Hsiang Huang, Hou Pong Chan, May Fung, Haoyi Qiu, Mingyang Zhou, Shafiq Joty, Shih-Fu Chang, and Heng Ji. 2025. From Pixels to Insights: A Survey on Automatic Chart Understanding in the Era of Large Foundation Models. *IEEE Transactions on Knowledge and Data Engineering* 37, no. 5 (May): 2550–2568. (Cited on page 1).

Christian Jaumann, Annemarie Friedrich, and Rainer Lienhart. 2025. Coling-UniA at SciVQA 2025: Few-Shot Example Retrieval and Confidence-Informed Ensembling for Multimodal Large Language Models. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 230–239. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 25, 53, 55 sq.).

Zeba Karishma, Shaurya Rohatgi, Kavya Shrinivas Puranik, Jian Wu, and C. Lee Giles. 2023. ACL-Fig: A Dataset for Scientific Figure Classification. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 37th AAAI Conference on Artificial Inteligence,* edited by Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, and Viet Dac Lai, vol. 3656. CEUR Workshop Proceedings. Washington DC, USA: AAAI. (Cited on pages 19, 44).

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35, no. 3 (March): 400–401. (Cited on page 8).

Anukriti Kumar and Lucy Lu Wang. 2024. Uncovering the New Accessibility Crisis in Scholarly PDFs: Publishing Model and Platform Changes Contribute to Declining Scholarly Document Accessibility in the Last Decade. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility,* 1–16. ASSETS '24. New York, NY, USA: Association for Computing Machinery, October. (Cited on page 1).

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. April. (Cited on page 12).

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems,* 33:9459–9474. Curran Associates, Inc. (Cited on page 24).

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning,* 19730–19742. ISSN: 2640-3498. PMLR, July. (Cited on page 25).

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. 2024. Multimodal ArXiv: A Dataset for Improving Scientific Comprehension of Large Vision-Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics,* edited by Lun-Wei Ku, Andre Martins, and Vivek Srikumar, 1:14369–14387. Bangkok, Thailand: Association for Computational Linguistics. (Cited on pages 2 sq., 22, 26, 28, 43 sq.).

Shengzhi Li and Nima Tajbakhsh. 2023. SciGraphQA: A Large-Scale Synthetic Multi-Turn Question-Answering Dataset for Scientific Graphs. ArXiv:2308.03349, August. (Cited on pages 19, 22, 44).

Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. 2025. A Survey of State of the Art Large Vision Language Models: Alignment, Benchmark, Evaluations and Challenges. ArXiv:2501.02189, April. (Cited on pages 14 sq., 22).

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out,* 74–81. Barcelona, Spain: Association for Computational Linguistics, July. (Cited on pages 34 sq.).

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A Survey of Transformers. *AI Open* 3 (January): 111–132. (Cited on page 12).

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved Baselines with Visual Instruction Tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 26286–26296. ISSN: 2575-7075. June. (Cited on page 22).

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. *Advances in Neural Information Processing Systems* 36 (December): 34892–34916. (Cited on page 22).

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. ArXiv:1711.05101, January. (Cited on page 27).

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019,* edited by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 13–23. NeurIPS 2019. Vancouver, BC, Canada, December. (Cited on page 13).

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2024. A survey on LoRA of large language models. *Frontiers of Computer Science* 19, no. 7 (December): 197605. (Cited on page 17).

Iain J. Marshall, Blair T. Johnson, Zigeng Wang, Sanguthevar Rajasekaran, and Byron C. Wallace. 2020. Semi-Automated Evidence Synthesis in Health Psychology:Current Methods and Future Prospects. *Health psychology review* 14, no. 1 (March): 145–158. (Cited on page 1).

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2025. ChartQAPro: A More Diverse and Challenging Benchmark for Chart Question Answering. ArXiv:2504.05506, April. (Cited on pages 21 sqq.).

Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Findings of the Association for Computational Linguistics.* ACL 2022. Dublin, Ireland: Association for Computational Linguistics. (Cited on pages 3, 21, 24, 52).

Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. ChartInstruct: Instruction Tuning for Chart Comprehension and Reasoning. In *Findings of the Association for Computational Linguistics,* edited by Lun-Wei Ku, Andre Martins, and Vivek Srikumar, 10387–10409. ACL 2024. Bangkok, Thailand: Association for Computational Linguistics, August. (Cited on page 23).

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. ChartAssistant: A Universal Chart Multimodal Language Model via Chart-to-Table Pre-training and Multitask Instruction Tuning. In *Findings of the Association for Computational Linguistics,* 7775–7803. ACL 2024. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics. (Cited on pages 1, 22).

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, Workshop Track Proceedings,* edited by Yoshua Bengio and Yann LeCun. ICLR 2013. Scottsdale, Arizona, USA, May. (Cited on page 8).

Prahitha Movva and Naga Harshita Marupaka. 2025. Enhancing Scientific Visual Question Answering through Multimodal Reasoning and Ensemble Modeling. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 252–262. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 25, 53 sq., 57).

Faith Wavinya Mutinda, Kongmeng Liew, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. 2022. Automatic data extraction to support meta-analysis statistical analysis: a case study on breast cancer. *BMC Medical Informatics and Decision Making* 22, no. 1 (June): 158. (Cited on page 1).

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation Learning with Contrastive Predictive Coding. ArXiv:1807.03748, January. (Cited on page 15).

OpenAI et al. 2024. GPT-4 Technical Report. ArXiv:2303.08774, March. (Cited on pages 2, 12, 22).

Antonio Picornell Rodríguez, Sandro Hurtado, María Antequera-Gómez, Cristóbal Barba-González, Rocío Ruiz-Mata, Enrique Gálvez-Montañez, Marta Recio, M.Mar Trigo, Jose Aldana Montes, and Ismael Navas Delgado. 2023. A deep learning LSTM-based approach for forecasting annual pollen curves: Olea and Urticaceae pollen types as a case study. *Computers in Biology and Medicine* 168 (November): 107706. (Cited on page 7).

Shraman Pramanick, Rama Chellappa, and Subhashini Venugopalan. 2024. SPIQA: A Dataset for Multimodal Question Answering on Scientific Papers. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024,* edited by Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang. NeurIPS Vancouver, BC, Canada, December 10 - 15, 2024. (Cited on pages 2 sq., 22, 26, 28, 43).

Nusrat Jahan Prottasha, Upama Roy Chowdhury, Shetu Mohanto, Tasfia Nuzhat, Abdullah As Sami, Md Shamol Ali, Md Shohanur Islam Sobuj, Hafijur Raman, Md Kowsher, and Ozlem Ozmen Garibay. 2025. PEFT A2Z: Parameter-Efficient Fine-Tuning Survey for Large Language and Vision Models. ArXiv:2504.14117, April. (Cited on pages 2, 14, 17 sq.).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning,* 8748–8763. Extended version available at http://arxiv.org/abs/2103.00020. PMLR, July. (Cited on pages 16, 25).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training (June). (Cited on page 12).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners, (Cited on page 12).

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems,* edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, 36:53728–53741. Curran Associates, Inc. (Cited on page 24).

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* (October). (Cited on page 12).

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 3980–3990. Hong Kong, China: Association for Computational Linguistics. (Cited on page 25).

Marco Rolandi, Karen Cheng, and Sarah Pérez-Kriz. 2011. A Brief Guide to Designing Effective Figures for the Scientific Paper. *Advanced Materials volume 23,* 4343–4346. (Cited on page 1).

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65 (6): 386–408. (Cited on page 5).

R. Rosenfeld. 2000. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE* 88, no. 8 (August): 1270–1278. (Cited on page 8).

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, no. 6088 (October): 533–536. (Cited on page 6).

Abel Salinas and Fred Morstatter. 2024. The Butterfly Effect of Altering Prompts: How Small Changes and Jailbreaks Affect Large Language Model Performance. In *Findings of the Association for Computational Linguistics: ACL 2024,* edited by Lun-Wei Ku, Andre Martins, and Vivek Srikumar, 4629–4651. Bangkok, Thailand: Association for Computational Linguistics, August. (Cited on page 56).

Florian Schleid, Jan Strich, and Chris Biemann. 2025. Visual Question Answering on Scientific Charts Using Fine-Tuned Vision-Language Models. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 211–220. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 25, 53).

Minghao Shao, Abdul Basit, Ramesh Karri, and Muhammad Shafique. 2024. Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges. *IEEE Access* 12:188664–188706. (Cited on page 12).

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27,* edited by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, 3104–3112. Annual Conference on Neural Information Processing Systems 2014. Montreal, Quebec, Canada, December. (Cited on pages 8 sq.).

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. ArXiv:2302.13971, February. (Cited on page 12).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. ArXiv:2307.09288, July. (Cited on page 12).

Dimitrios Tsirmpas, Ioannis Gkionis, Georgios Th. Papadopoulos, and Ioannis Mademlis. 2024. Neural natural language processing for long texts: A survey on classification and summarization. *Engineering Applications of Artificial Intelligence* 133 (July): 108231. (Cited on pages 9, 11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems,* vol. 30. Curran Associates, Inc. (Cited on pages 9 sqq.).

Viviana Ventura, Lukas Amadeus Kleybolte, and Alessandra Zarcone. 2025. Instruction-tuned QwenChart for Chart Question Answering. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025),* edited by Tirthankar Ghosal, Philipp Mayr, Amanpreet Singh, Aakanksha Naik, Georg Rehm, Dayne Freitag, Dan Li, Sonja Schimmler, and Anita De Waard, 240–251. Vienna, Austria: Association for Computational Linguistics, July. (Cited on pages 24 sq., 53 sq., 57).

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. ArXiv:2409.12191, October. (Cited on page 22).

Zichong Wang, Zhibo Chu, Thang Viet Doan, Shiwen Ni, Min Yang, and Wenbin Zhang. 2024. History, Development, and Principles of Large Language Models-An Introductory Survey. ArXiv:2402.06853, (Cited on page 8).

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. 2024. CharXiv: Charting Gaps in Realistic Chart Understanding in Multimodal LLMs. *Advances in Neural Information Processing Systems* 37 (December): 113569–113697. (Cited on page 3).

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (December): 24824–24837. (Cited on page 23).

Yifan Wu, Lutao Yan, Leixian Shen, Yunhai Wang, Nan Tang, and Yuyu Luo. 2024. ChartInsights: Evaluating Multimodal Large Language Models for Low-Level Chart Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024,* 12174–12200. Miami, Florida, USA: Association for Computational Linguistics. (Cited on page 22).

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, and Yu Qiao. 2024. ChartX & ChartVLM: A Versatile Benchmark and Foundation Model for Complicated Chart Reasoning. ArXiv:2402.12185, (Cited on page 22).

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 483–498. Association for Computational Linguistics. (Cited on page 12).

Chuqiao Yan, Hans-Peter Hutter, Felix M. Schmitt-Koopmann, and Alireza Darvishy. 2025. Chart Accessibility: A Review of Current Alt Text Generation. *IEEE Access* 13:94040–94056. (Cited on page 1).

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 Technical Report. ArXiv:2412.15115, January. (Cited on pages 12, 23).

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2024. A Survey on Multimodal Large Language Models. ArXiv:2306.13549, *National Science Review* 11, no. 12 (November). (Cited on page 15).

Hongzhi Zhang and M. Omair Shafiq. 2024. Survey of transformers and towards ensemble learning using transformers for natural language processing. *Journal of Big Data* 11, no. 1 (February): 25. (Cited on page 12).

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-Language Models for Vision Tasks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, no. 8 (August): 5625–5644. (Cited on pages 14 sqq.).

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024. TinyChart: Efficient Chart Understanding with Program-of-Thoughts Learning and Visual Token Merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing,* edited by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, 1882–1898. Miami, Florida, USA: Association for Computational Linguistics, November. (Cited on page 23).

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. April. (Cited on page 35).

Yunong Zhang. 2023. Encoder-decoder models in sequence-to-sequence learning: A survey of RNN and LSTM approaches. *Applied and Computational Engineering* 22, no. 1 (October): 218–226. (Cited on pages 5 sqq.).

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* edited by Anna Korhonen, David Traum, and Lluís Màrquez, 1441–1451. Florence, Italy: Association for Computational Linguistics, July. (Cited on page 12).

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. Publisher: arXiv Version Number: 16, (Cited on page 13).

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision* 130, no. 9 (September): 2337–2348. (Cited on pages 3, 16).

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models. ArXiv:2504.10479, April. (Cited on pages 22, 25, 53, 57).

Affidavit

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellen- verzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe. Sofern im Zuge der Erstellung der vorliegenden Abschlussarbeit generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten Wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

I hereby declare in lieu of an oath that I have written this thesis for the Master's degree programme in Computer Science independently and have not used any aids other than those specified - in particular no Internet sources not named in the list of sources. All passages taken verbatim or in spirit from publications are labelled as such. I further certify that I have not previously submitted the thesis in another examination procedure. If, in the course of preparing this thesis, generative artificial intelligence (gAI)-based electronic tools were used, I affirm that my own contribution was the primary focus and that a complete documentation of all tools used is provided in accordance with Good Scientific Practice. I accept responsibility for any incorrect or distorted content, faulty references, violations of data protection or copyright law, or plagiarism generated by the gAI.

Hamburg, 01.11.2025	F. Schleid
Date	Signature
	(Florian Schleid)

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, 01.11.2025	F. Schleid
Date	Signature
	(Florian Schleid)