Trier University

Vision-Based Similarity Computation for Case-Based Retrieval of Argument Graphs

Thesis

submitted in fulfillment of the requirements for the degree of
Bachelor of Science (B.Sc.) in Business Informatics

Reviewer:	Prof. Dr. Ralph Bergmann
	Mirko Lenz, M.Sc.
Supervisor:	Mirko Lenz, M.Sc.

Kilian Bartz
Schmiedestr. 12, 54636 Meckel
№ 1538561
s4kibart@uni-trier.de

April 4, 2024

Abstract

Over the last few years, there has been a surge of interest in *Artificial Intelligence* (AI) vision models, however, using them for automatic processing of drawn natural visual representations like graphs has not yet been evaluated. This could enable many new applications, especially the implementation of a fast and scalable retrieval for argumentation machines.

In this thesis, I propose a new approach to implement structural retrieval in the context of a case-based argumentation machine using a Swin Transformer v2 model to transform visualized *Argument Graph*s (AGs) into dense embeddings on which similarities can be calculated very efficiently. In an experimental, iterative approach, I conceptualize multiple suitable visualization designs based on node-link graph drawings and treemaps, which aim to capture an AG's structure. These visualizations are used to train corresponding vision models for the embedding process. I demonstrate that even though treemap-based designs show more promising training behavior, more comprehensive node-link drawings exhibit better retrieval performance for complex queries. The A* search from previous works outperforms vision-based argument retrieval for simple queries; however, a higher query complexity noticeably increases the quality of vision-based argument retrieval. Furthermore, vision-based similarity computation can improve computation times by several orders of magnitude and also exhibits promising scaling of retrieval quality with larger training data sets and longer training durations. However, due to the complex and opaque information processing of Transformer models, the structural similarity values produced by the vision-based approach can be unpredictable and counterintuitive. Additionally, vision-based argument retrieval does not represent a complete replacement for A* search as it does not provide mappings from the query to the case base arguments.

Keywords Case-based reasoning, argumentation machines, embedding, retrieval, k-NN, Vision Transformer

Contents

Acronyms

ADU *Argumentative Discourse Unit*

AE *Auto Encoding*

AG *Argument Graph*

AI *Artificial Intelligence*

CB *Case Base*

CBR *Case-based Reasoning*

I-node *Information Node*

K-NN *K-Nearest Neighbors*

NLP *Natural Language Processing*

S-node *Scheme Node*

ViT *Vision Transformer*

List of Figures

List of Tables

List of Algorithms

1. Introduction

Argumentation plays an important role in daily life and is essential for cultural, social, and intellectual progress (Van Eemeren 2018). While news papers and political columns are prevalent sources for argumentative texts, they can also be presented in a structured manner, e.g. as *Argument Graph*s (AGs). AGs can be visualized using established graph visualization techniques, such as node-link graph drawings or hierarchical treemaps. These visualizations cannot only be used by human argumentation experts to understand and process the structure of an argument graph, but can also be processed by a computer vision model to enable the efficient implementation of downstream tasks on these arguments. Concretely, this thesis integrates a current vision model into the context of the argumentation machine envisioned by the ReCAP project (Bergmann et al. 2018) to enable efficient argument retrieval.

This envisioned argumentation machine is a system which can retrieve relevant arguments and synthesize new arguments for not yet well explored topics (Lenz et al. 2019). The argument retrieval of such an argumentation machine could support many scientists, journalists, and politicians. A user could use it to query an argument data base to obtain a general overview of arguments for and against a specific topic (Bergmann et al. 2019). While content-based (semantic) retrieval of relevant arguments could ideally result in a more informed decision-making process for politicians, structural retrieval might allow them to find arguments of which they can adapt the structure to form a more persuasive new argument. Journalists, on the other hand, could use the argumentation machine retrieval to verify facts. This could make an argumentation machine a vital tool for public discourse, it could mitigate the spread of false news, and could lead to better decision making. An argumentation machine, which cannot only work on the argumentative texts itself but also has a sound understanding of the argument structure, could be vastly superior compared to technologies like traditional search engines and text summarization tools for these use cases.

Argument retrieval can be implemented in the context of a *Case-based Reasoning* (CBR) system (Bergmann et al. 2019; Lenz et al. 2019). The argumentation machine manages the case base of known arguments and can retrieve relevant arguments by performing a *K-Nearest Neighbors* (K-NN) search to find the arguments with the highest similarity to a particular user query. For this argumentation machine to provide a high practical usefulness in public discourse, it is beneficial if the argumentation machine can operate on a large argument data base to allow searching through a comprehensive and diverse corpus of arguments. Problematically, previous works exploring argument retrieval in this context struggle with a high computational complexity during graph-based similarity computation due to the resource-intensive A* search (Bergmann and Gil 2014) used. This severely limits the scalability of the database and the maximum complexity of the

arguments, which can be retrieved in a feasible amount of time by such an argumentation machine.

Motivated by recent breakthroughs in the domain of computer vision, in this thesis, I explore a vision-based approach for structural argument retrieval using the visualizations of argument graphs mentioned above. Although vision models are traditionally used with real-world photos (e.g. for image classification (Dosovitskiy et al. 2021)) and have not yet been extensively explored for processing of drawn natural representations, such as graph drawings, they can potentially enable the implementation of argument retrieval, which is scalable and flexible enough to handle a large and diverse knowledge base. In comparison to previous works based on A* search, this approach is likely to produce more inaccurate similarities; however, as the process of first visualizing and subsequently working with visual representations of argument graphs closely resembles the process of humans, vision-based argument retrieval might even improve the alignment of retrieval results to the baseline rankings produced by human experts.

To develop the concept of vision-based structural argument retrieval, I conducted a literature research to explore the possibilities and requirements of contemporary text-, graph-, and image-based embedding and retrieval models. I then conducted a second literature research to develop an effective, readable, and fast-interpretable visualization. This visualization was used to train the first *Vision Transformer* (ViT) model, using a training recipe adapted from successful text and image embedding models. To improve the suitability of the visualization design for ViT models, I use an experiment-driven approach and systematically iterate on visualization attributes, using findings from earlier visualizations to improve model training behavior for subsequent visualizations.

In this thesis, the problem of argument graph retrieval is explored in isolation of other argumentation machine tasks and does not include the generation of mappings between the query and the individual case base arguments, which are a byproduct of A* search in previous works and could be used in subsequent argumentation machine tasks. Further, the actual gathering and building of the knowledge base, especially with respect to the required diversity, reliability, and correctness of the arguments, lies outside of the scope of this thesis.

Concretely, the following **research question** is evaluated in this thesis: Does vision-based structural similarity computation present a faster and more scalable alternative to structural similarity computation based on the A* search for argument graphs?

The **contributions** of my thesis are as follows:

- I propose a novel vision-based approach for computing structural AG similarity and a suitable pipeline for structural retrieval to enable efficient argument retrieval in the context of an argumentation machine.
- I compare several different visualization techniques for AGs to find out which of them are best-suited for training ViT models and can provide a high alignment to the reference rankings of human experts in the structural argument retrieval task.
- I evaluate the performance of the structural retrieval pipeline, comparing it to the previous A* search, and perform individual studies on the impact of graph changes

on the computed similarity, the improvements resulting from extended model training, as well as how well both approaches scale regarding graph complexity.

My thesis is structured as follows: First, I explain the core foundations which are necessary for understanding this thesis (Section 2) and refer to related work (Section 3). In Section 4 I introduce the five visualizations explored in this thesis, as well as the corresponding methodology and vision-based similarity computation pipeline. After that, I evaluate the performance of the vision-based argument retrieval in Section 5 and end with a conclusion and an outlook for future work in Section 6.

2. Foundations

In this section, the terminology, core concepts, and technologies that are essential to understand this thesis will be introduced. This includes the problem domain of argumentation theory, as well as technical foundations in *Case-based Reasoning* (CBR), MAC/FAC retrieval, embedding tasks, the Transformer architecture and graph drawing.

2.1. Argumentation Theory

Argumentation is defined as a communicative and interactional act complex aimed at resolving a conflict of opinion by means of reasoning (Van Eemeren 2018). This is done by convincing the opposing party of the acceptability of a standpoint by presenting a constellation of propositions that meet shared *critical standards of reasonableness* (Van Eemeren 2018). To communicate a point of view reasonably, an argument in its simplest form (an Aristotelian syllologism, see Lewiński and D Mohammed (2016)) consists of a set of *Argumentative Discourse Unit*s(ADUs), i.e., **premises** and **claims**. An example of this can be found in Figure 2.1.

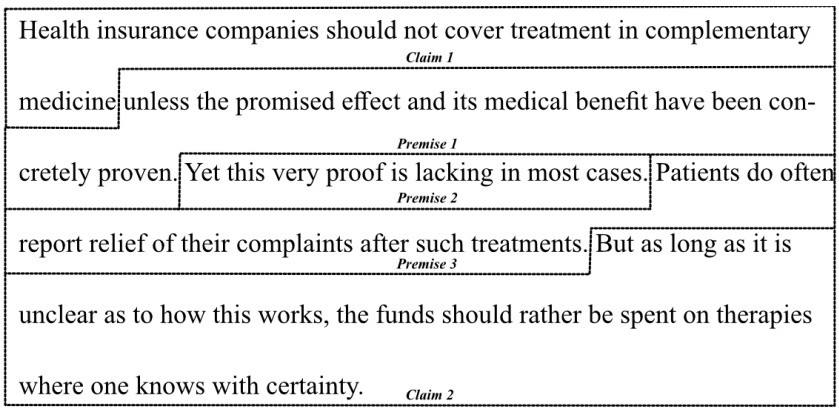

Figure 2.1.: Example for an argument, divided into its components. Argument source: Peldszus and Stede (2015)

The main area of argument theory scholarship deals with **argumentation schemes**, with which arguers can adapt to their audiences (Lewiński and D Mohammed 2016). Douglas Walton identified 96 argumentation schemes as stereotypical patterns of human reasoning in Walton et al. (2008). An example of an argumentation scheme is *Argument from Analogy* which can be seen in Figure 2.2. These schemes or rules of inference can be used to embed the relationship between a premise and a claim (i.e. supporting or

Argument from Analogy

Major Premise: Generally, case C1 is similar to case C2. Minor Premise: Proposition A is true (false) in case C1. Conclusion: Proposition A is true (false) in case C2.

Figure 2.2.: Walton's argumentation scheme *Argument from Analogy*. Source: Lewiński and D Mohammed (2016)

attacking) into the aforementioned *critical standards of reasonableness* and can therefore be seen as the **local argumentation structure**. The roles of premises and claims inside these argumentation schemes, as well as the critical questions to assess each of them will not be considered in this thesis.

On the other hand, **global argumentation structure** focuses on the way complex inferences are organized (Lewiński and D Mohammed 2016). In general, the following three main structures have been identified (Lewiński and D Mohammed 2016):

- **serial** premises, which support each other in a chain that leads to the conclusion,
- **linked** premises, where several premises constitute together to lead to the conclusion
- **convergent** premises, where several independent lines of support all reach the same conclusion.

Chesñevar et al. (2006) proposed to represent an argument as an *Argument Graph* (AG), which is a directed graph where each node can be an *Information Node* (I-node) or a *Scheme Node* (S-node). I-nodes represent the contents of the argument, i.e., the domain-dependent claims of the argument. On the other hand, S-nodes are domain-independant and represent the applied argumentation schemes. An example of such an AG can be found in Figure 2.3.

Bergmann et al. (2019) and Lenz et al. (2019) formalized and extended an argument graph as a tuple (N, E, τ, λ, t), where N represents the set of the graph's nodes, E represents the set of its directed edges, τ maps each node to a type, λ maps each node to a semantic description, and t represents the overall topic of the graph.

2.2. Case-based reasoning

CBR is a sub-discipline of *Artificial Intelligence* (AI) and uses past experiences to solve new problems (Bergmann et al. 2009). The entire process of CBR can be represented as a cycle, which contains a **retrieval step**, **reuse step**, **revise step**, and **retention step** (Aamodt and Plaza 2001). Past experiences called cases are stored in *Case Base* (CB). A new problem q can be solved by querying CB for relevant cases (**retrieval**).

These relevant cases are the most similar cases contained in CB, because one of the core assumptions of CBR is that similar cases have similar solutions (Bergmann et al.

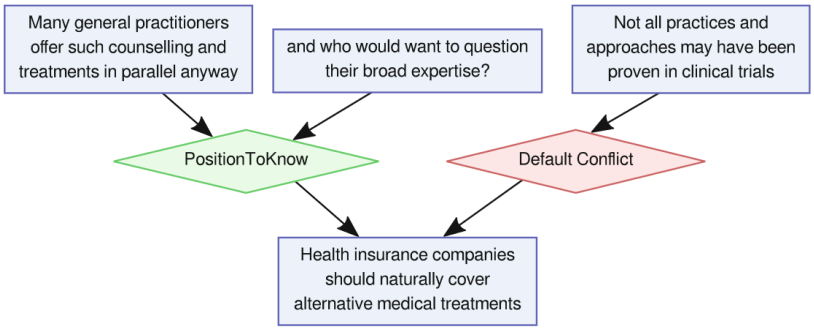

Figure 2.3.: Visualized argument graph. The blue rectangles represent I-nodes; the red and green rhombuses represent S-nodes. Illustration source: Lenz et al. (2019); argument source: Peldszus and Stede (2015)

2009) and a higher similarity value suggests that the respective case is more useful for solving the new problem (Bergmann and Gil 2014).

Finding the most similar cases is a K-NN task (Bergmann et al. 2009). In its simplest form, a similarity function sim is used to calculate the pairwise similarity $\text{sim}(q, c_i), c_i \in$ CB between the new problem and every case in CB. Then, the cases c_i with the highest similarity values are selected. The selected cases can then be **reused** by adapting them (**revision**) to the current problem, generating a new solution. This solution can finally be used to increase the knowledge stored in CB by persisting the new experience alongside past cases (**retention**) (Bergmann et al. 2009).

CBR can be divided into several categories, including structural and textual CBR (Bergmann et al. 2009). Structural CBR defines a fixed structure for case representations like attribute-value tables or graph structures. On the other hand, textual CBR represents cases as free texts, simplifying the work with large collections of existing know-how text documents (Bergmann et al. 2009).

This makes CBR very useful in an experience-driven customer management context, such as Customer Relationship Management (Choy et al. 2002) or Fraud Detection (Wheeler and Aitken 2000). However, CBR has also been explored for the retrieval of (business) workflow graphs in the context of process-oriented CBR (POCBR) (Bergmann and Gil 2014), as well as argument graph retrieval (Bergmann et al. 2019, Lenz et al. 2019). For the latter two cases, the cases are graphs (semantically labeled workflow graphs or argument graphs, respectively). For the retrieval phase, special graph-based similarity measures can be utilized that make use of the additional information present in these graphs.

2.3. Retrieval using MAC/FAC

To improve computation times for similarity-based retrieval as used in the context of CBR, a MAC/FAC (Forbus et al. 1995) approach (*many are called, but few are chosen*)

can be implemented. Here, the retrieval process is split into two phases. In the first phase (MAC phase), all cases within CB undergo a very efficient, high recall pre-filtering process, finding a small number of candidate cases. This can significantly cut down on the number of cases which have to be considered in the FAC phase, for which the actual, computationally more expensive similarity measure between the retrieval query and the cases has to be computed. The performance gain of applying a pre-filtering MAC phase increases with the complexity of the similarity function used in the FAC phase and the total size of CB.

2.4. Embeddings

In an embedding task, common goals are the quantification of **semantic similarity** between objects (L Yu et al. 2019, Su et al. 2022) and **dimensionality reduction**, i. e. transforming high-dimensional objects into lower-dimensional, often dense **vector representations** (Cai et al. 2018; Xu 2021). To achieve this, a deep neural network can be trained to learn a mapping $f : \mathbb{R}^b \rightarrow \mathbb{R}^a, a \ll b$ with $\mathbb{R}^b$ representing the original object space and $\mathbb{R}^a$ representing the lower-dimensional embedding space.

The most important attribute of the learned embeddings to enable semantic clustering of objects in the embedding space is the preservation of information, especially the **preservation of semantic distance**. Two objects, which are considered similar by users / experts, should be projected close to each other in the embedding space (L Yu et al. 2019). This allows the quantification of similarities between original objects by applying standard similarity measures such as the dot product or cosine distance to the embeddings of the objects (Xu 2021).

In a retrieval task, this property can enable the comparison of elements from two otherwise unrelated object spaces. For example, Radford et al. (2021) embedded images and texts into a joint multi-modal embedding space, which enables retrieval of an appropriate text caption using an image as the query. Girdhar et al. (2023) expanded on this and embedded audio, images, video, depth maps, and text into a shared embedding space, which enables cross-modal retrieval, embedding space arithmetic, and audio to image generation.

For graph embeddings, the reduction of object dimensionality while maximally preserving the graphs' structures is traditionally the main goal (Cai et al. 2018; Xu 2021). If the embeddings can provide a significant dimensionality reduction, this can greatly improve the efficiency of downstream computation by reducing computation times and decreasing memory complexity. Ideally, if the vector representation maximally preserves the properties of the object, these downstream computations remain accurate. This opens up the potential for new applications that might be too computationally expensive for the original objects.

2.5. Transformer Architecture

The Transformer architecture, originally developed for translation tasks (Vaswani et al. 2017), has been successfully applied to all types of *Natural Language Processing* (NLP) tasks and forms the backbone of every modern large language model (Brown et al. 2020, Touvron et al. 2023, Penedo et al. 2023). However, the Transformer architecture can also be applied to images (Dosovitskiy et al. 2021) where its heavy use of self-attention, large-scale pre-training and bidirectional feature encoding benefit several vision tasks like object detection or image classification (Khan et al. 2022).

The novelty of the Transformer architecture lies in its heavy use of self-attention layers (Khan et al. 2022). For each element in an input sequence $z = (z_1, z_2, \ldots, z_n) \in \mathbb{R}^{n \times d}$ with length n and embedding dimension d, a self-attention layer estimates the attention weights $A \in \mathbb{R}^{n \times n}$ of the elements to each other (Dosovitskiy et al. 2021). The self-attention value $\mathrm{SA}(z)$ is then calculated as a weighted sum over all values $z_i, i \in \{1, \ldots, n\}$ in the sequence (Dosovitskiy et al. 2021). This allows the model to capture interactions between all n elements in the sequence by updating each element in the sequence using aggregated global contextual information (Khan et al. 2022). This means that even long-range dependencies in a sequence can be taken into account, which might alter the context of an item significantly.

For example, in the sentence "I play the guitar", the word *play* would have a different self-attention value than in the context of the sentence "Do you play chess?", as the embedding values for the other words in the respective sentences (sequences) would be different.

To enable the Transformer architecture to work with image data, Dosovitskiy et al. (2021) proposed dividing an image into fixed-size patches, which are then fed into a linear projection layer. After combining the patch embeddings from the projection with position embeddings, they can be fed into a Transformer model as a sequence of vectors where self-attention can be applied.

Self-attention is especially beneficial when working with images in a self-supervised manner. Caron et al. (2021) showed that a ViT model may automatically learn image class-specific features, which leads to unsupervised object segmentation.

Based on the original ViT architecture (Dosovitskiy et al. 2021), Swin Transformer V1 and V2 improve on it by increasing its efficiency and suitability as a large-scale vision model.

V1 (Z Liu et al. 2021b) improves the efficiency of the original ViT model by limiting the computation of self-attention to non-overlapping local windows only (achieving linear complexity with respect to image size) as opposed to the original ViT's global implementation (quadratic complexity) (Z Liu et al. 2021b). To allow for cross-window connections, the model cyclically shifts the self-attention windows by alternating between the original image partition and a shifted image partition in subsequent Transformer blocks.

V2 (Z Liu et al. 2021a) builds on this and aims to improve the ability of the model to scale up to a large-scale vision model. For this reason, the training stability is improved by adjusting the model architecture, the resolution gap between low-resolution pre-

training images and high-resolution downstream images is addressed, and SimMim (Xie et al. 2022) is proposed as a self-supervised training framework to alleviate the need for massive amounts of labeled data during training (see Section 2.6).

2.6. Training of a Transformer-based model

The Transformer architecture shows remarkable scaling with model size and size and quality of the training data set (Dosovitskiy et al. 2021, Z Liu et al. 2021a). In general, the training process is divided into the two phases *pre-training* and *fine-tuning*.

2.6.1. Pre-training

During this phase, the model should be trained to understand the general domain and develop general-purpose low-level and high-level abilities (Raffel et al. 2020). In the context of NLP examples of low-level abilities could be understanding of spelling or meaning, while high-level abilities could be comprehension of abstract logic.

The pre-training phase can be performed supervised (Dosovitskiy et al. 2021) or self-supervised. While the choice does not limit later downstream abilities (in general, pre-trained models are extended by task-specific prediction heads and undergo an additional task-specific fine-tuning (Z Liu et al. 2021a)), current state-of-the-art models (Z Liu et al. 2021a; Oquab et al. 2024) employ self-supervised learning, as this allows the usage of large and diverse unlabeled training data sets. Various self-supervised approaches have been explored in the context of NLP (Z Liu and Shao 2022, Devlin et al. 2018, Radford et al. 2018) and vision (M Chen et al. 2020, W Wang et al. 2022, Xie et al. 2022).
The approaches for self-supervised pre-training can generally be divided into

1. masked language/image modeling,
2. *Auto Encoding* (AE) / reconstruction based training.

A schematic illustration of their functional principles can be found in Figure 2.4.

Masked modeling

In masked modeling, a set percentage of input tokens gets masked. When working with images, this means masking the pixel values for some of the image patches. The model is then trained by predicting the original pixel values before masking based on the remaining tokens. There are a few challenges when implementing this approach in the vision domain (Xie et al. 2022):

- **Locality**: Close pixels are highly correlated as they can only form an object together. While this can also be the case with text, most of the time, the tokens of a text are independent (separate words) and might also refer to objects that are very far apart in the sentence.

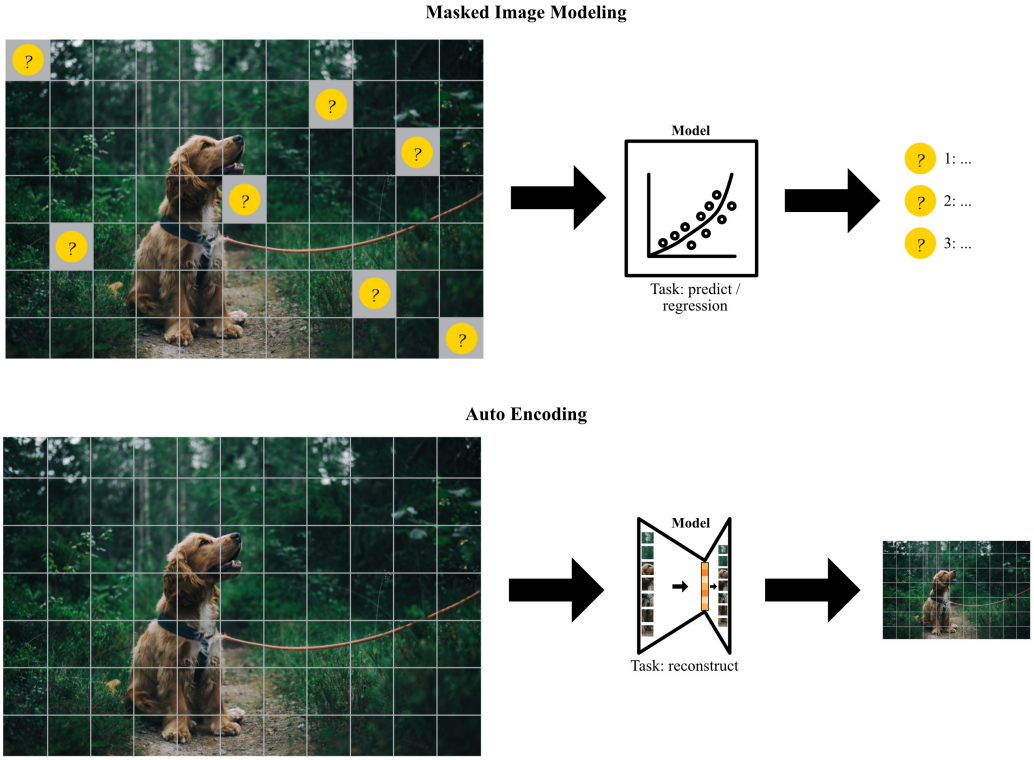

Figure 2.4.: Functional principle of masked modeling and auto encoding

- **Low vs. high level signals**: Text is human-made and consists of words representing high-level concepts. Image pixels, on the other hand, are low-level signals that do not carry any intrinsic meaning on their own.
- **Continuous vs discrete domains**: In a modeling perspective, a text is a sequence of tokens which are manifestations in a discrete feature space (i.e. the model's vocabulary). The color of a single image pixel, however, is a value from the continuous color space.

Because of this, masked image modeling cannot be easily formulated as a classification task in which the model simply predicts the label of masked patches. Although there are some complex approaches that avoid these challenges (M Chen et al. 2020, W Wang et al. 2022), according to Xie et al. (2022) formulating masked image learning as a regression task and predicting raw pixel values, performs as well as complex approaches while providing a much simpler model.

Auto Encoding

Masked modeling has the limitation that it cannot learn on the masked tokens, i.e., it cannot use them to understand the context and relation to other tokens, possibly causing

this approach to be inefficient in the use of its training samples. To mitigate this, training based on *Auto Encoding* (AE) uses an auto-encoder architecture, consisting of an encoder and a decoder. An input x is given to the encoder, which then projects it into a low-dimensional latent space (embedding space): $enc(x) = \overline{x}$. The decoder then attempts to reconstruct the original input using the output of the encoder: $dec(\overline{x}) \stackrel{!}{=} x$. The training objective is the minimization of reconstruction error. To increase the difficulty of the task, in many cases the decoder is very limited, often restricted to one layer (Z Liu and Shao 2022, Su et al. 2022). This is intended to force the encoder to compress a maximal amount of meaningful information into the embeddings, thus improving the embedding quality.

2.6.2. Fine-tuning

The fine-tuning phase is highly task-specific and usually uses specialized supervised data sets for the respective downstream tasks. In the case of an embedding model, a commonly used self-supervised technique is contrastive learning (T Chen et al. 2020a, T Chen et al. 2020b).

Contrastive learning has first been successfully used in the vision domain by T Chen et al. (2020a). It uses batches B of positive and negative samples $B = (x, X^+, X^-)$ with x representing a training sample, X^+ the set of positive samples and X^- the set of negative samples. The goal of an embedding model is to construct an embedding space that enables similar objects to be projected into close proximity, while dissimilar objects are as far apart as possible. To achieve this, the model is trained using the following contrastive loss (T Chen et al. 2020a):

$$\ell_i^{\text{NT-Xent}} = -\log \frac{\exp(\text{sim}(q_i, k_i)/\tau)}{\exp(\text{sim}(q_i, k_i)/\tau) + \sum_{j \neq i} \exp(\text{sim}(q_i, k_j)/\tau)} \tag{2.1}$$

for i, j in $\{0, \ldots, \text{batch\_size}\}$ where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, and τ represents *temperature* as a hyperparameter.

A big challenge when using contrastive learning is the possibility of representation collapse (Grill et al. 2020), where the model outputs the same vector for each input, as well as finding suitable sets X^+ and X^- for each training sample x in the data set. For vision models, random augmentations (cropping, color shifting, etc.) of x can be used to generate samples from X^+ (T Chen et al. 2020a, Grill et al. 2020) while all other samples from the training batch are grouped as X^- (in-batch negatives). Training is very sensitive to these augmentations, and the particular augmentations chosen significantly influence model performance (T Chen et al. 2020a). As opposed to exclusively using in-batch negatives, more sophisticated models also utilize hard-mined negatives (i.e., edge cases that should be considered as dissimilar retrieved by an external mechanism) to further improve performance (Asai et al. 2022; Qu et al. 2020).

2.7. Graph drawings

Visualizing graphs is a problem that connects two disciplines: visualization design and graph drawing. Visualization design is primarily focused on human perception and how to effectively convey information to the intended audience and is especially relevant in geography, library science and design (Munzner 2014). Graph drawing, as a sub-discipline of computer science, on the other hand, mainly focuses on the algorithmic problem of efficiently generating a visualization of a graph while adhering to certain rules (e.g. minimizing line crossings, minimizing bends, etc.) (Angelini and Hanxleden 2023; Bekos and Chimani 2023).

To improve interpretability and readability, the literature on visualization design provides guidelines on which visual elements (overall visualization techniques, colors, shapes, edge types) are effective in representing certain information (Munzner 2014).

2.7.1. Visualization techniques for hierarchical data

Graph drawings are well studied and are "the most universal model that can be structured relatively easily and that supports data visualization even with hundreds of thousands of nodes" (Kolomeets et al. 2023). In the context of this thesis, graph drawings refer to node-link tree drawings, which are generally made up of nodes, which represent objects, as well as edges, symbolizing the relations between objects (Kolomeets et al. 2023).

Graphs have the following limitations:

- Graph layout algorithms fail when data gets too large and complex (i.e. very heterogeneous with multiple links between nodes), or data is not always connected Kolomeets et al. (2023).
- A graph drawing generally inherits the shape of the underlying graph, which might be very wide or very deep, when using uniform node sizes.

Because of this, treemaps can be considered as alternative visualizations that can show complete information about the hierarchical structure, although they are generally used to indicate the containment of objects rather than their connection (Munzner 2014). All children of a tree node are enclosed within the area allocated to the parent node. Although they are not as effective in conveying a graph's topology (e.g., if the goal is to trace a certain path through the tree), they can be used to quickly spot outliers of attribute values (Munzner 2014).

2.7.2. Color

Color has been found to be the most powerful aspect to effectively encode information in a visualization and can be interpreted faster than other aspects, such as shape and size (Karim et al. 2019, Nowell et al. 2002). In addition to that, color is also very versatile and can be used to encode not only categorical, but also continuous data attributes (Munzner 2014). For categorical encoding, especially, it is important that colors have the same

perceived intensity, as differences could lead to differences in salience (Munzner 2014). Colors should also be distinguishable, which means choosing colors that are sufficiently distant and separable from other colors by a straight line in the color space (D'Zmura 1991). In addition, the number of colors used should be limited to a small number (Silva et al. 2011). Furthermore, Reda et al. 2021 found that color nameability also improves interpretability for human testers.

To provide linear separability and equidistance between colors and partition them into different named color regions, Healey (1996) proposed to first measure and transfer a monitor gamut to an isoluminant uniform color space such as CIELUV. There, the colors could be chosen as equally spaced points around the circumference of the largest inscribed circle.

2.7.3. Shapes

In graph visualization, shape plays a subordinate role, even though it can impair other aspects like color in conjunction with size. This is because the color of a node cannot be perceived if its area is too small (Munzner 2014). Even if shape has a greater impact on interpretability than size (Nowell et al. 2002) for user interface design, there is no concrete guidance in the literature as to which types of shape should be used.

2.7.4. Edge Types for Graph Drawings

While the most common type of edge used to draw graphs is a classic arrow, with the arrowhead indicating the direction of an edge, Holten and Van Wijk (2009) suggests that it is inferior to many other edge types. Specifically, tapered edges, which start wide at the source node and become narrower on their way to the target node, performed best in the user study they conducted, as they indicate direction very intuitively and reduce visual clutter.

3. Related Work

This thesis combines approaches from several different fields, namely structural and textual CBR, graph embedding and image-based retrieval.

3.1. CBR-based Argument Graph Retrieval

Bergmann et al. (2019) established the basis of CBR-based AG retrieval, combining structural CBR with graphs as a case representation with textual CBR methods (semantic text similarity). They proposed using a MAC/FAC approach (see Section 2.3) where AGs are first filtered based on their topics to semantically match the query and then ranked according to their graph-based similarity, which considers semantic and structural graph attributes. Concretely, in the MAC phase, the semantic embedding of the topic embedding t_i of every case base graph is compared to the embedding of the query's topic embedding t_q. The arguments with the k most similar topic vectors are then selected as the result of the MAC phase.

The graph-based similarity (FAC phase) between a query graph Q and a case base graph C proposed by them is based on admissible mappings $m : N_q \cup E_q \rightarrow N_c \cup E_c$ between the nodes and edges of the query (N_q, E_q) and the nodes and edges of the case base graph (N_c, E_c), respectively. Through such a mapping m, the local node similarities $\text{sim}_N(n_q, n_c)$ for $n_q \in N_q$ and $n_c \in N_c$ and the local edge similarities $\text{sim}_E(e_q, e_c)$ for $e_q \in E_Q$ and $e_c \in E_c$ can be calculated. These local similarities are then aggregated as a weighted average to get $\text{sim}_m(Q, C)$. The final similarity value is then calculated as $\text{sim}(Q, C) = \max\{\text{sim}_m(Q, C) \mid \text{admis. map } m\}$.

The mappings themselves are computed using the A*I algorithm (Bergmann et al. 2019). A search node S used in this A* implementation represent the current mapping S.m, the not yet mapped nodes S.N and the not yet mapped edges S.E. Starting with an empty mapping, in every iteration of this algorithm, the first search node S from the priority queue is expanded by extending S.m in all possible ways by mapping an additional node or edge. All these expended nodes are inserted into the priority queue based on the value of a function $f(S) = g(S) + h(S)$ where $g(S)$ represents the similarity of the current mapping S.m and the heuristic $h(S)$ estimates the additional similarity achieved when also mapping the remaining nodes and edges.

Concretely this means that for the final similarity value $\text{sim}(Q, C)$ between the query and a single case base graph, **at least** $|N_Q| + |E_Q|$ similarity values for smaller mappings have to be computed as part of the A*I algorithm (this case would need a perfect heuristic $h(S)$), setting the lower bound for time complexity for the FAC phase to $\Omega(k(|N_Q| + |E_Q|))$.

To experimentally evaluate their retrieval approach, Bergmann et al. (2019) compared the retrieval results of an isolated MAC phase, an isolated FAC phase, as well as the combined MAC/FAC phases in regards to their alignment with expert assessments on the *argumentative microtexts* corpus (Peldszus and Stede 2015). They showed that the combined MAC/FAC approach was able to improve computation time and retrieval quality.

Lenz et al. (2019) built upon this and evaluated additional semantic text similarity measures, as well as an ontology-based semantic similarity measure for argumentation schemes, and were able to improve the results reported by Bergmann et al. (2019).

3.2. Graph Embedding

In the domain of graph embedding, there are multiple works in which graph embedding was explored to reduce computational costs and memory requirements for graph analysis. Popular approaches are random walk-based methods, as utilized by Perozzi et al. (2014) and Grover and Leskovec (2016), and neural network-based methods, using Graph Convolutional Networks (Kipf and Welling 2016) or Graph Transformers (H Tang et al. 2020). They focus heavily on the structure of the argument graphs and embed graph nodes and edges individually.

To represent an entire graph as a vector instead, graph kernels are usually used (Cai et al. 2018). Here, the resulting vector contains the counts of the elementary substructures from which the graph is constructed. Different methods include decomposing a graph into graphlets (fixed-sized subgraphs), sub-tree patterns, or random walks (Cai et al. 2018).

3.3. Vision Transformer-based Image Retrieval

Although, to my knowledge, ViT embeddings have not been used to process graph drawings or other human-made visualizations, it has been successfully applied for general image retrieval (El-Nouby et al. 2021). The authors successfully trained a ViT model using a Siamese architecture and a metric learning objective to generate image descriptors (i.e., embeddings) which can then be used to retrieve appropriate images.

They show that fine-tuning an off-the-shelf ImageNet pre-trained ViT model and regularizing the output feature space to encourage uniformity significantly reduces overlap between positive and negative pairs and is subsequently able to improve the retrieval quality above the previous state-of-the-art in category-level retrieval held by convolutional models. Additionally, they suggest that ViT models can outperform more complex convolutional models in this task with short vector representations and are able to be used unmodified in other categories (e.g. particular image retrieval), which necessitated training a new model in the past.

Finally, there is a lot of research on text-based retrieval (L Gao and Callan 2021; Qu et al. 2020; L Wang et al. 2022a) and text embedding models (T Gao et al. 2021; Su et al. 2022; L Wang et al. 2022b; Xiao et al. 2023) which can be combined with a K-NN search

for retrieval tasks. As they also use the Transformer model architecture, many of the methods described in this work can be used to work with images by simply exchanging the (text) Transformer model they used with a ViT model.

4. Argument Graph Retrieval Using Vision-based Structural Similarities

This thesis explores a vision-based structural similarity computation on AGs. This computation can be used alongside content-based pre-filtering to implement a very efficient retrieval stage for an argumentation machine. Compared to previous work (Bergmann et al. 2019; Lenz et al. 2019), I expect this approach to result in a highly reduced computation time during inference because it offers good scaling in regards to graph complexity and case base size and also allows performing a major part of the similarity computation at store time. Ideally, the convincing image retrieval performance of ViT models (El-Nouby et al. 2021) and the similar process of processing argument graphs to experts (i.e., visualizing the graph first and then working on the structure visualization) should allow this approach to maintain decent alignment with expert assessments.

4.1. Problem Description

The problem domain is argument retrieval in the context of an argumentation machine. Concretely, the task consists of retrieving those arguments from a set S of known argument graphs, which are semantically (MAC phase) and structurally (FAC phase) closest to the user-submitted query, which is also an AG. This means that exactly those arguments $s \in S$ are relevant retrieval results for a query q, which match both the query's semantic requirements ($s \in \text{MAC}(q)$) and the query's structural requirement ($s \in \text{FAC}(q)$):

$$\text{rel}_S(q) = \text{MAC}_S(q) \cap \text{FAC}_S(q)$$

In this thesis, the problem will be relaxed by the following assumption: For an argumentation machine user, the semantic similarity is a requirement to consider a particular argument as a relevant candidate. This means that there are cases where good structural matches to q may be discarded if they are a bad semantic match.

Because of this, the retrieval process can be split into two stages. First, all argument graphs within S are filtered based on their semantic similarity to the user's query q in the MAC phase. The top k results are selected as retrieval candidates. The second stage (FAC) takes these k arguments as input and ranks them according to their structural similarity to q. The output of these two stages are the k arguments within S which are the best semantic and structural matches to q with respect to the assumption above.

As the I-nodes of the argument graphs are already processed as part of the semantic similarity computation during the MAC phase, I further assume that only the S-nodes of the semantically relevant AGs suffice to perform structural ranking.

4.2. Methodology

After conducting a literature research on the possibilities and requirements of text-, graph-, and image-based embedding and retrieval models, I developed the structural argument retrieval pipeline presented below. To convert AGs into a format that can be processed by a vision model, I conceptualized five different visualizations (V1-V5) based on classic node-link graph drawings and treemaps, focusing on encoding the structure of AGs. I conducted a literature research on visualization design to construct a visualization design which is effective in terms of interpretability and readability for visualization V1. These insights are adapted to the capabilities of the *Graphviz* tool[1]. Then, I trained a Swin Transformer v2 model on these images, using a training recipe adapted from successful text and image embedding models. To improve the suitability of the visualization design for ViT models, I use an experiment-driven approach and systematically iterate on visualization attributes to improve model training behavior for consecutive visualizations V2-V4 (see Appendix A), using the same training recipe as before (V5 is assumed to be similar enough to V4 to be compatible with the models for V4). Finally, I experimentally evaluate the capabilities of each visualization for argument retrieval by examining the alignment of the rankings produced by the visualizations and their corresponding vision models with expert assessments.

4.3. Concept

Like in previous works, the retrieval approach explored in this thesis will be CBR-based. There are several reasons why CBR is especially suitable for working with arguments in the context of an argumentation machine, which is why it will be used as a foundation for dealing with argument graphs in this thesis:

1. **Alignment with argumentation machine tasks**. CBR is a problem-solving architecture designed to allow the retrieval, reuse, and revision of past cases (Aamodt and Plaza 2001). This aligns well with the tasks of an argumentation machine, where we need to be able to retrieve arguments (cases in the case base) and to adapt an argument to the needs of the current problem to synthesize a new argument (this corresponds to reusing and revising a past case).

2. **Flexibility**. Structural CBR allows for argument representations which also take their structure into account. In contrast to approaches that solely represent arguments as strings, the integration of this added information into the retrieval process should allow for a better alignment with expert rankings.

[1] `https://graphviz.org/`

The general representation of arguments in the form of AGs in AIF format is adopted from the works of Bergmann et al. (2019) and Lenz et al. (2019). Similarly, the two-stage filtering process explored in this thesis is derived from these works. The MAC phase is adopted unchanged to efficiently filter the AGs in the CB on a content level to find the k case base arguments with the highest content similarity to a user query q. The graph-based similarity computation in the FAC phase, which takes place after that, is replaced by the pipeline presented below (Section 4.3) which ranks the k retrieval candidates based on their structural similarity.

It should be noted that my approach represents an adapted MAC/FAC pattern, where the FAC phase does not compute the complete similarity between query and case base arguments, like the graph-based similarity (which includes both semantic and structural elements of the argument graphs) from previous works does (Bergmann et al. 2019; Lenz et al. 2019)). Instead, the two phases of my approach filter CB based on two independent metrics (semantic and structural similarity), and are able to find overall matches when used in conjunction with each other. On the one hand, this reduces computational cost, because only one aspect of the AGs must be considered in the FAC phase. On the other hand, this approach is more flexible. It also allows for fast retrieval times in scenarios where the argumentation machine user is not interested in semantically pre-filtering the CB and might only be interested in structural matches regardless of the arguments' topics. In this use case (i.e. MAC(q) = CB), the MAC phase does not reduce the computation time for the actual graph-based similarity calculation in the FAC phase of the previous approach. This would result in infeasible retrieval times for even a moderate number of medium-sized argument graphs in the argumentation machine's CB (see Section 5.6).

Vision-based Structural Similarity Pipeline

The pipeline to calculate the structural similarities between a query and the case base graphs is represented by Algorithm 1.

Algorithm 1 Vision-based computation of the structural similarities between a query graph q in AIF format and the set of retrieval candidates RC in AIF format

procedure COMPUTE_STRUCTURAL_SIMILARITY(q, RC)
  $\text{vis}_q \leftarrow$ VISUALIZE(q)
  $\text{emb}_q \leftarrow$ EMBED(vis_q)
  for $r_i \in$ RC **do**
    $\text{vis}_{r_i} \leftarrow$ VISUALIZE(r_i)
    $\text{emb}_{r_i} \leftarrow$ EMBED(vis_{r_i})
    $\text{sim}_{q,r_i} \leftarrow$ COSINE_SIMILARITY($\text{emb}_q, \text{emb}_{r_i}$)
  sort RC according to sim
  return sorted RC

This means, that the AGs (q and $r_i \in$ RC) are processed in three phases:

1. Visualization using visualization designs V1-V5 presented below,

2. Embedding the visualization image with a ViT model,
3. Cosine similarity computation.

Embedding task

For the embedding task, ViT models might be especially suitable. This is because their architecture allows attending to long dependencies between the input sequence elements (Khan et al. 2022), as might be the case for large argument graphs. Furthermore, in the context of graph drawings, the unsupervised image segmentation possible with ViT models (see Section 2.5) might help to separate the background (white-space) from the actual information in form of the shapes and colors used to visualize the graph structure. Finally, several works (Girdhar et al. 2023; Radford et al. 2021) show the ability of ViT models to produce good embeddings of different modality objects, which might also transfer to AGs.

Several ViT models with different training mechanisms and different architectures can be used as base models for this task. Promising models include the original ViT (Dosovitskiy et al. 2021), DEiT (Touvron et al. 2020) and BEiT (Bao et al. 2021). However, due to its useful implementation as part of HuggingFace's transformer package [2] and the improvements to the original ViT model, I use Swin Transformer v2 (Z Liu et al. 2021a) models, trained as detailed in Section 5.2.1.

Following related work on text embeddings (Reimers and Gurevych 2019, B Li et al. 2020), I apply mean pooling of the hidden state of the last layer to generate an embedding for the respective image as the average of the patch embeddings of the input image:

$$\text{pooling} : \mathbb{R}^{s \times h} \rightarrow \mathbb{R}^{h}$$

Here, s is the sequence length of the flattened image patches, and h stands for the hidden size, the dimensionality of hidden states of the model.

Cosine similarity computation task

For the final computation of similarities based on the embeddings, traditionally, the choice is between the *dot product* and *cosine similarity*. There are cases where one similarity measure yields better results in practice than the other (Karpukhin et al. 2020; Steck et al. 2024; Thongtan and Phienthrakul 2019), but no general statement about the superiority of any of these measures can be made. Because of this, I chose *cosine similarity* for practical reasons arising from its normalization to $[0, 1]$: Firstly, the resulting values of this measure can be interpreted more easily, e.g. a *cosine similarity* of 1 indicates that the similarity was computed between two identical objects (and, by extension, identical original objects), while the values of the *dot product* between two vectors can only be interpreted in comparison to other *dot product* values. Additionally, this normalization can make the implementation of downstream tasks easier, since the exact value range for the similarity values is known.

[2] `https://huggingface.co/docs/transformers/model˙doc/swinv2`

In the following sections, I describe the visualization designs V1-V5 and the corresponding training procedure for the Swin Transformer v2 models.

Visualizing Graphs

In CBR, the quality of the similarity function is very important and should be modeled according to the particular problem domain (Bergmann and Gil 2014). The inner workings of the ViT model are very complex, and the exact resulting effect of a particular layer from the input towards the embedding are not very transparent. However, based on the assumption that a chosen visualization severely impacts the embeddings generated by a ViT model, visualization is the part of the pipeline which can be adapted to the problem domain the easiest and should therefore be leveraged to increase the quality of the similarity values produced by the ViT model and therefore increase the quality of the retrieval.

Visualization Design V1: Visual Graph Representation Based on Traditional Hierarchical Node-Link Drawings

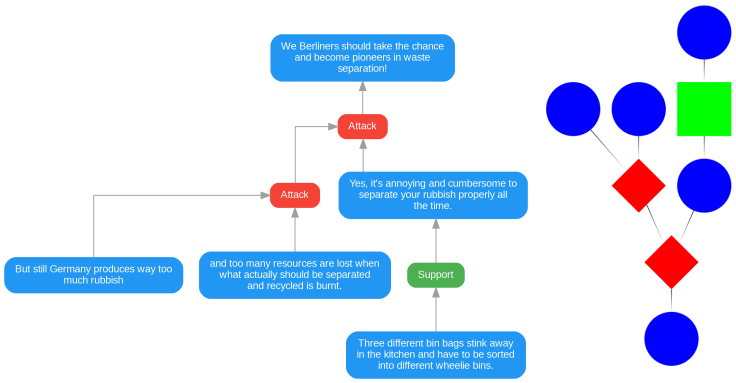

Figure 4.1.: Example of graph structure visualization V1: The AG from the argumentative microtexts corpus (Peldszus and Stede 2015) on the left is transformed into the condensed, structure-focused version on the right

The goal of this visualization design is to create a minimal and condensed graph representation, which focuses solely on the graph structure and offers the ability to approximate the structure of an underlying AG at a glance.

For the representation of AGs, node-link graph drawings are intuitively suitable, as the AGs used to evaluate the visualizations only contain fully connected nodes, which are all connected to the argument's major claim, and their complexity stays within the limits of graph drawings.

In alignment with the goal of visualization V1, I follow findings from relevant literature (see Section 2.7.1) and try to reduce its complexity as much as possible. This is because it should generally be favorable to maximally reduce the complexity of the problem,

allowing the use of smaller models and training data sets. Added to that, it should also reduce unwanted bias, which consequently should further reduce the amount of training samples needed. An example graph visualized using the methods and attributes explained below can be found in Figure 4.1.

Shapes and text labels

To keep the graph as simple as possible, I only use ellipses (representing I-nodes) and rhombuses (representing S-nodes) of constant size, without using any other shapes. Regarding text, while it is possible to process text occurring in images using Vision Transformers (J Wang et al. 2022, J Li et al. 2022) and even develop a multimodal model capable of understanding the relations between text and images (J Li et al. 2023, H Liu et al. 2023), the added complexity of processing and understanding text would likely require a much larger training data set. Furthermore, label texts are normally used to represent the relation type of the S-nodes and to display the content of a premise or claim on the I-nodes. Because I-node labels are not regarded for the vision-based argumentation structure, relying on text to encode further information is not necessary.

Edge type

In accordance with the literature findings on edge types, I utilize Graphviz's tapered edge implementation. An example of this can be found in Figure 4.2. The arrowhead, which does not have to be drawn for this edge type, should reduce visual clutter, particularly when dealing with very condensed images where there is little space between important visual features.

Color

ViT models operate in the RGB color space where a color c can be represented as a tuple $c = (x_1, x_2, x_3)$ of color proportions of the elementary colors red, green and blue. As such, many of the color-based deliberations (see Section 2.7.2) can be simplified. V1 adopts the important traits of equidistance between colors, constant color intensity, and good color separability. To ensure a constant color intensity, the sum of the color proportions can be fixed to a constant a: $\sum_{i \in \{1,2,3\}} x_i = a$ with $a = \text{const}$. In this case, I chose a to be 255. This means that the three elementary colors red $(255, 0, 0)$, green $(0, 255, 0)$, and blue $(0, 0, 255)$ satisfy these constraints.

Color will be used to encode the type of graph nodes: I-nodes are blue, attacking S-nodes are red and supporting S-nodes are green.

Visualization technique and layout

V1 is a node-link graph drawing. Regarding the layout, Graphviz' well-studied dot engine, which is based on the Sugiyama Framework (Sugiyama et al. 1981), will be used for V1. However, it should be noted that an alternative layout engine, such as PBF

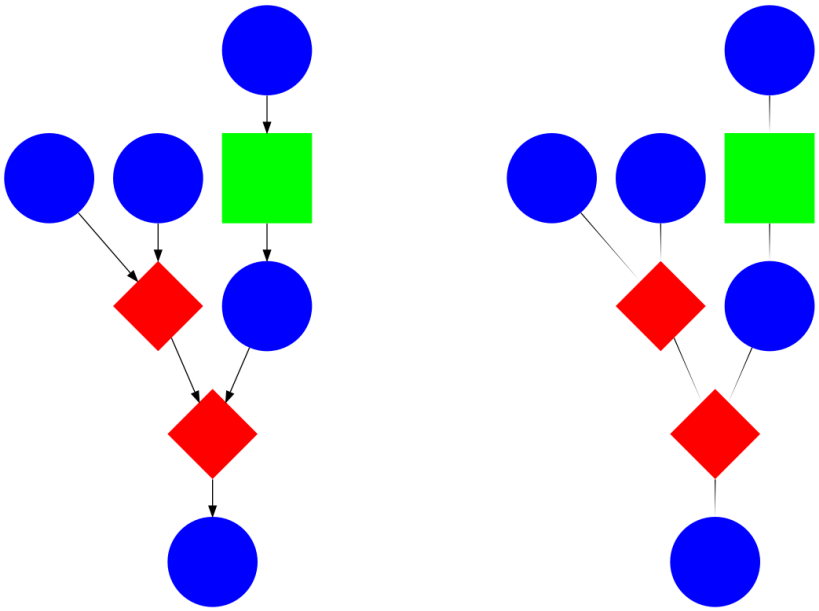

Figure 4.2.: Example for an AG from the *Microtexts* corpus (Peldszus and Stede 2015). The visualization on the left uses traditional edges; the visualization on the right uses tapered edges which indicate their direction by a gradual decrease of edge width from the source to the target node.

(Lionakis et al. 2023) could be superior and could increase the retrieval quality of the vision-based similarity pipeline.

Visualization Design V2: Visual Graph Representation using a Radial Layout

Many commonly used graph layout engines, such as hierarchical layouts or force-directed layouts, produce graphs that match the graph's overall shape, i.e., when a graph is very wide, the resulting image will also be very wide. This is not well suited for processing with a ViT, as its input is generally expected to be square. If a wide image is to be used as input for a ViT model, it has to be squeezed into the available square dimensions, which means that ViT cannot work on the original dimensions and instead has to rely on a distorted visualization.

To generate circular graph visualizations, which do not have to be distorted to fit into a square input window, I chose Graphviz' radial layout engine *twopi*, based on Wills (1999). This layout engine also has the side effect that individual branches of the argument graph tend to be placed close together when using the visualization, which puts ADUs with a relation between them closer together. This should help when processing the image with any vision model. An example for the visualization of an AG using V2 can be found in Figure 4.3.

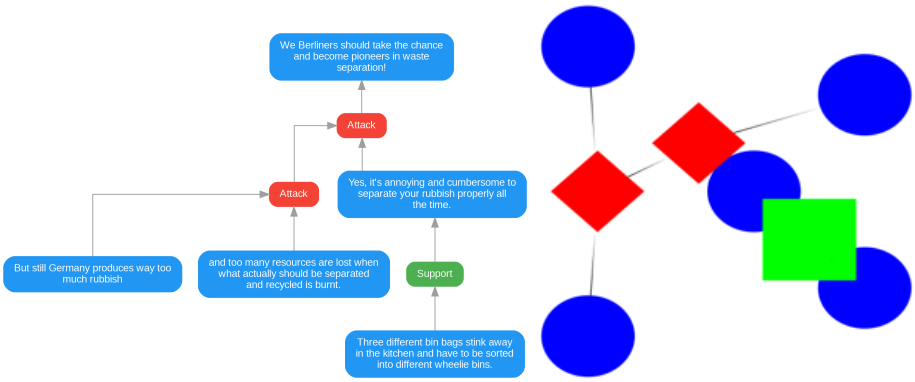

Figure 4.3.: Example of graph structure visualization V2: The AG from the argumentative microtexts corpus (Peldszus and Stede 2015) on the left is transformed into the visualization on the right. Note: Although the layout resembles a circle only for larger graphs, it is already able to spatially group certain related elements closer together for smaller graphs.

Visualization Design V3: Visual Graph Representation Based on Treemaps

As the arguments evaluated in this thesis are always trees, hierarchical visualizations such as treemaps can be used. Rectangular treemaps can be guaranteed to be square and fill out the entire input window of a ViT model. This should increase encoding efficiency as no space is wasted (i.e., no white-space, which carries no information), improving the more efficient input space usage of V2. An example of V3 can be found in Figure 4.4.

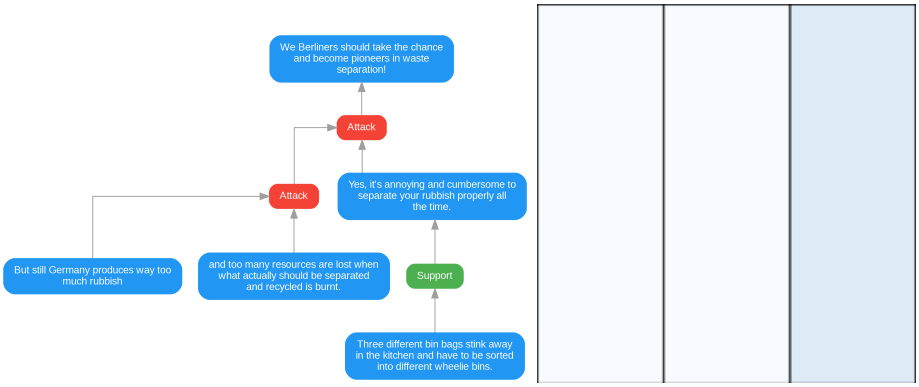

Figure 4.4.: Example of graph structure visualization V3: The AG from the argumentative microtexts corpus (Peldszus and Stede 2015) on the left is transformed into the visualization on the right. Here, there are two I-nodes that refer to the major claim (depth 1; light blue) and one I-node which refers to a child of the major claim (depth 2; darker blue).

V3 is built on the assumption that the overall structure of an AG is determined by the form of its branches. This means, for example, that an AG with two equally long branches has a different structure from an AG with two branches of different lengths or an AG with three branches.

To represent the structures of the argument graph as a treemap, I focus on the containment of the I-nodes (i.e., the parent-child relation between I-nodes). The rectangular area of the nodes is equally divided into smaller rectangles based on the number of children that this node has. The color of the area represents the depth d of this node. This is implemented in Algorithm 2. It works by dividing the original square (with height h, width w, a position defined by $x = 0$ and $y = 0$ and depth $d = 0$) into |children(major_claim)| rectangles of equal size. Recursively, the area of each child is divided into equal-sized rectangles and colored according to their depth. Rectangles are defined by the *Rectangle* method which takes the position, dimensions, and a value that can be mapped to a corresponding color. The type of relation between parent and child nodes (i.e., attack or support) is ignored for V3. As a color palette, I used the tool *ColorBrewer* [3] (Harrower and Brewer 2003), which generates color palettes for human viewers based on best practice from literature.

It should be noted that there are treemap characteristics that might be detrimental to the training success of the ViT. This includes the high reliance on color contrast (as there is otherwise no white-space to separate visual features), as well as the fact that shallow leaf nodes might have a stronger impact on the graph embedding because they occupy a larger area than the leaf nodes of deeper graph branches.

[3] `https://colorbrewer2.org/`

Algorithm 2 Recursive computation of treemap rectangles using an AG's I-Nodes. It takes a node, the position (x, y) and dimension (h, w) of its allocated rectangle, a node depth d and a boolean horizontal, which flips the partitioning direction on every recursion.

```
procedure GET_TREEMAP_RECTS(node, x, y, h, w, d, horizontal)
    parts ← |CHILDREN(node)|
    if parts = 0 then
        return {}
    if horizontal then
        c_width ← ⌊w / parts⌋
        c_height ← h
    else
        c_width ← w
        c_height ← ⌊h / parts⌋
    res ← {}
    for idx, child ∈ CHILDREN(node) do
        calculate child position c_x, c_y according to idx and horizontal
        res ← res ∪ {RECTANGLE(c_x, c_y, c_width, c_height, d)}
        res ← res ∪ GET_TREEMAP_RECTS(child, c_x, c_y, ..., d + 1, ¬horizontal)
    return res
```

Visualization Design V4: Visual Graph Representation Based on Treemaps (I-Nodes, Saturated colors)

After analyzing the ViT training results (see Appendix A), it became apparent that the model is not able to differentiate the different colors used sufficiently and especially struggles with their proximity to white. Because of this, for iteration V4, I chose to still use Algorithm 2 to draw the treemaps and only change the color palette to more saturated colors with a higher contrast. An example for a AG visualization with V4 can be found in Figure 4.5.

Visualization Design V5: Visual Graph Representation Based on Treemaps (S-Nodes)

V5 builds on the assumption that the branching degree of I-Nodes is secondary to the overall graph structure, and instead, the visualization should focus on representing the relations between S-nodes to visually represent serial, linked, or convergent premises (see Section 2.1). For V5, the following deliberations influenced the visualization design:

- Because of the prior explanations, the basis of the representation will remain a treemap.
- The colors of the treemap rectangles indicate the type of the corresponding S-node; in particular, attack nodes are represented by a red area, and support nodes by a green area.

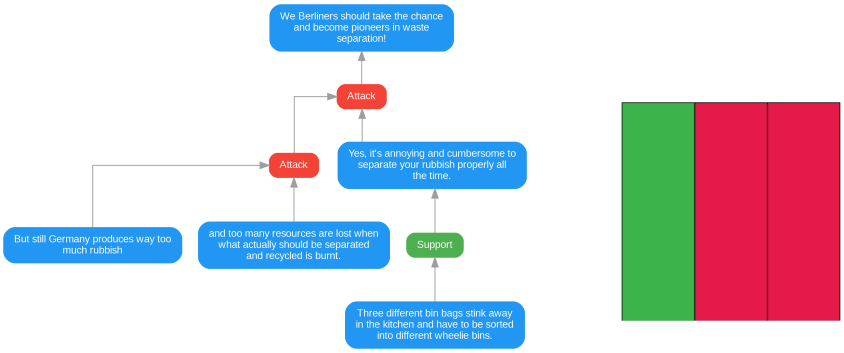

Figure 4.5.: Example of graph structure visualization V4: The AG from the argumentative microtexts corpus (Peldszus and Stede 2015) on the left is transformed into the visualization on the right. Here, there are two I-nodes which refer to the major claim (depth 1; red) and one I-node which refers to a child of the major claim (depth 2; green).

- It is integral that every S-node is represented. Therefore, instead of the previous algorithm, where the area of a parent node is generally fully divided into the rectangles of their children, the parent node claims 10% of the available area, and only the remaining area is equally divided for the children nodes.

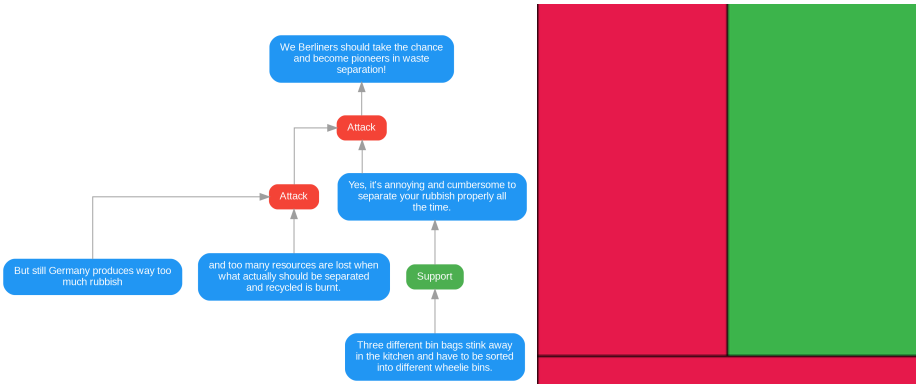

Figure 4.6.: Example of graph structure visualization V5: The Microtexts (Peldszus and Stede 2015) AG on the left is transformed into the image on the right. Here the major claim is attacked by premise 1 (red row at the bottom) which is attacked by premise 2 and 3 (left red column) and supported by premise 4 (right green column).

This is achieved with Algorithm 3. Compared to Algorithm 2, it does not regard the depths of the nodes, as they are not used to color the rectangles. An example can be found in Figure 4.6.

Algorithm 3 Recursive computation of treemap rectangles using an AG's S-Nodes. It takes a node, the position (x, y) and dimension (h, w) of its allocated rectangle and a boolean horizontal, which flips the partitioning direction on every recursion.

procedure GET_TREEMAP_RECTS(node, x, y, h, w, horizontal)
    parts $\leftarrow$ |CHILDREN_SNODES(node)|
    own_width $\leftarrow w$
    own_height $\leftarrow 0.1 \times h$
    if parts = 0 **then**
        return RECTANGLE(x, y, own_width, own_height,LABEL(node))}
    y = y + own_height
    if horizontal **then**
        c_width $\leftarrow \lfloor \frac{w}{\text{parts}} \rfloor$
        c_height $\leftarrow h \times 0.9$
    else
        c_width $\leftarrow w$
        c_height $\leftarrow \lfloor \frac{h}{\text{parts}} \rfloor$
    res $\leftarrow$ { RECTANGLE(x, y, own_width, own_height,LABEL(node))}
    for idx, child $\in$ CHILDREN(node) **do**
        calculate child position c_x, c_y according to idx and horizontal
        res $\leftarrow$ res $\cup$ GET_TREEMAP_RECTS(child, c_x, c_y, . . . , ¬horizontal)
    return res

5. Experimental Evaluation

Using visualizations V1-V5 for argument graphs (described in Chapter 4) and the training procedure described below in Section 5.2.1, the performance of the respective visualization designs and the corresponding vision models will be evaluated and compared to each other, as well as to previous works (Bergmann et al. 2019, Lenz et al. 2019) in a systemic evaluation.

I also performed three small-scale studies to show the effects of extending a model's training, a comparison to the similarities produced by the A* algorithm, as well as the time scaling properties of the vision pipeline in comparison to A* for large graphs.

5.1. Hypotheses

To answer the research question whether vision-based similarity computation presents a faster and more scalable alternative to A* search, and to verify the training methodology, I investigate the following theses in a systematic evaluation:

- **H1**: The retrieval quality (NDCG, correctness, completeness) achieved through vision-based structural similarity computation is similar to the retrieval quality of the A* search from previous work. It does not reduce any of the metrics by more than 10%.
- **H2**: Vision-based structural similarity computation significantly reduces retrieval times, especially for complex AGs.
- **H3**: The graph visualization technique alters the similarity-based ranking and thus directly affects the scores of the evaluation metrics.
- **H4**: The fine-tuned models outperform the models, which are only pre-trained, in terms of retrieval quality (NDCG, correctness, completeness).

5.2. Experiment Setup

The vision models were trained as described below; the training behavior is described in Appendix A. To maintain comparability with the results of previous works, I use the same annotated corpus of argumentative microtexts (Peldszus and Stede 2015) and the same data set of queries which Bergmann et al. (2019) and Lenz et al. (2019) used. It contains 24 argument graphs, belonging to 6 different topics, with accompanying expert rankings. The expert rankings are a relevancy ranking of the argument graphs in the

corpus that share the query's topic, ranging from 1.0 (highly relevant) to 3.0 (mostly irrelevant).

The query data set, called *microtexts-retrieval-simple* in the following, is constructed out of 12 trivial queries, containing only a single I-node, as well as 7 queries with one S-node and 5 queries with two S-nodes.

To evaluate the structural retrieval quality of vision-based similarity computation on more complex graphs, the same evaluation is carried out additionally on *microtexts-retrieval-complex*, a data set containing 12 queries also related to the argumentative microtexts corpus with 3-5 S-nodes per query. For this data set, there are no reference values from previous works.

As the scope of this thesis is to analyze the usefulness of a vision-based structural argument retrieval, an ideal MAC phase will be simulated by only considering argument graphs with the same topic as the query. This guarantees a constant starting point for all ViT models, which is not influenced by any choices related to the MAC phase.

After the simulated MAC phase, the vision-based structural similarity pipeline has the task of ranking the remaining cases in an order that should be as close as possible to the order of the expert rankings. Because the number of cases with the same topics as the respective queries is limited, I deviate from the previous works (Lenz et al. 2019, Bergmann et al. 2019), where the $k = 10$ most similar cases were considered for each query. Instead, only arguments with a corresponding topic (6-8), i.e., only the relevant arguments, are considered, which is why the average precision always amounts to 1.0 and is omitted in the following tables.

The metrics Normalized Discounted Cumulative Gain (NDCG), Correctness (CR), and Completeness (CP) will be computed for each experiment, as described by Lenz et al. (2019). The reported metrics are averaged over all queries. Added to that, the retrieval time in seconds (duration) on an Intel Core i7 8700 is measured, which already provides a large performance improvement. GPUs could be used to further reduce the time needed for the embedding phase and cosine similarity computation, however, as seen in Section 5.6, these phases are already so fast on the CPU that the CPU-exclusive visualization phase becomes the bottleneck for complex AGs.

5.2.1. Training Swin Transformer v2 Models for V1-V4

For the training of the embedding model, I followed the first two steps of the training recipe for C-TEM (Xiao et al. 2023), a contemporary text embedding model for the Chinese language:

1. Pre-training in a self-supervised manor using unlabeled data,
2. General purpose fine-tuning using contrastive learning.

Because the goal of C-TEM was to develop multipurpose embeddings which adapt to their domain, they proposed a third training step of multitask learning through instruction-based fine-tuning. However, this is beyond the scope of this thesis, where the embeddings will only be used in a retrieval context, which is why this third step is

skipped. Apart from that, they also used labeled data for the third step, which I do not have access to.

The model is implemented in *PyTorchLightning* 2.1.1 using the *transformers* library [1]. Due to the compatibility of the visualization designs V4 and V5, the Swin Transformer v2 models trained for V4 (V4-pt and V4-ft) are also used for the evaluation of visualization V5.

Pre-Training

In order to pre-train the Swin Transformer v2 model from scratch, I start with a randomly initialized SwinV2 model, which is pre-trained using an AE approach for 100 epochs on the data set (see Section 5.2.2). I chose not to use a pre-trained model because popular pre-training image data sets like ImageNet-22K (Deng et al. 2009) or COCO (Lin et al. 2014) are developed for photo classification tasks and do not contain any images of graphs. Because of that, pre-training on these data sets might not bring any benefit for this application, and instead might even lead to unwanted bias and unpredictability.

The Swin Transformer v2 model trained for V1-V4 are based on Microsoft's tiny Swin Transformer v2 configuration[2] with a patch-size of 4 and an input window of 256x256px (27.6M parameters). The decoder is a single linear layer (2.4M parameters) that maps the encoder's latent representation (i.e., the embedding) to a 32x32px image space.

During training, an *ImageProcessor* prepares an input image by transforming it into a vector of pixel values (3 channels x 256 x 256). During this step, potential resizing, normalization, and conversion to PyTorch tensors are performed. The encoder then receives these pixel values as input and transforms them into an embedding. For the tiny Swin Transformer v2 model, these are vectors with 768 entries. The decoder then transforms this embedding into an image in a smaller image space (32x32px). The loss of the entire AE is computed by downscaling the original pixel values from 256x256px to 32x32px and calculating the Mean Squared Error between the raw pixel values.

Fine-Tuning

I adapted the approach of Karpukhin et al. 2020 to implement contrastive learning using only in-batch negatives. The training model consists of a previously pre-trained Swinv2 ViT model (27.6M parameters) with a simple MLP head (one hidden layer of 4×768, overall 213K parameters).

I implemented the training process very similarly to T Chen et al. (2020b):

1. Each image x from the training batch is randomly augmented twice, which generates two views of every input which represent each others positive pairs: q, k.

2. q and k are encoded using the encoder network (the pre-trained Swin Transformer v2 model), resulting in the embeddings e_q and e_k.

[1] `https://huggingface.co/docs/transformers/index`
[2] `https://huggingface.co/microsoft/swinv2-tiny-patch4-window8-256`

3. The embedding dimensionalities are reduced by passing them through an MLP projection head.
4. The contrastive loss is calculated between every element's corresponding image view and every other element in the batch (in-batch negatives) on the reduced embeddings.

Specifically, after using an *ImageProcessor* to prepare the images, for a batch of 64 images, 2 contrastive views per image are created. These views are derived from the original images by using the following transformations:

- random horizontal flips,
- random vertical flips,
- Gaussian Blur
- random crop (an area of 40% - 90% of the original image is resized to the original dimensions) and
- dropout to simulate random noise.

The first four transforms are derived from the original SimCLR transforms (T Chen et al. 2020a); dropout is inspired by T Gao et al. 2021. It should be noted that color jitter, as one of the most important transforms (T Chen et al. 2020a) could not be used. This is because a change of color for a node in V1-V2 or a rectangle (V3-V5) might completely change its meaning and therefore represent a different graph structure.

After the Swinv2 model transfers every image view to its corresponding embeddings, they are passed through the MLP head. Afterwards, the loss (see Equation 2.1) between every pair of corresponding image views and every other image in the batch is computed and used to optimize the model parameters. As discussed by T Chen et al. (2020b), the projection head is used during training to prevent the *curse of dimensionality* when computing the loss.

5.2.2. Data sets

The training data sets for every model evaluated in Section 5.3 (pre-training and fine-tuning for a particular visualization design share the same data set) consisted of the visualizations images generated for the AGs from the AG corpora listed in Table 5.1. The respective data sets can be found in Table A.1.

For pre-training of the extended Swin Transformer v2 evaluated in Section 5.4, I created a synthetic data set containing 415640 V4 treemap images after removal of duplicates. The treemaps themselves have a random depth depth $\in \{1, \ldots, 9\}$, with the area for each depth divided noSplits $\in \{0, \ldots, S\}$ times with $S = \max\{2, 7 - \text{depth}\}$. This generates treemaps that represent argument graphs with a maximum depth of 9 where a node has a maximum branching degree of 7. The definition of S ensures that the deeper nodes have a lower branching degree to prevent the deeper nested rectangles in the

resulting image from getting so small that the vision model ignores them. A small subset of the data set images can be found in Figure 5.1. The data set can be found under `https://huggingface.co/datasets/kblw/treemap_sat`. The extended fine-tuning for this model uses the same dataset as the normal V4 fine-tuned model.

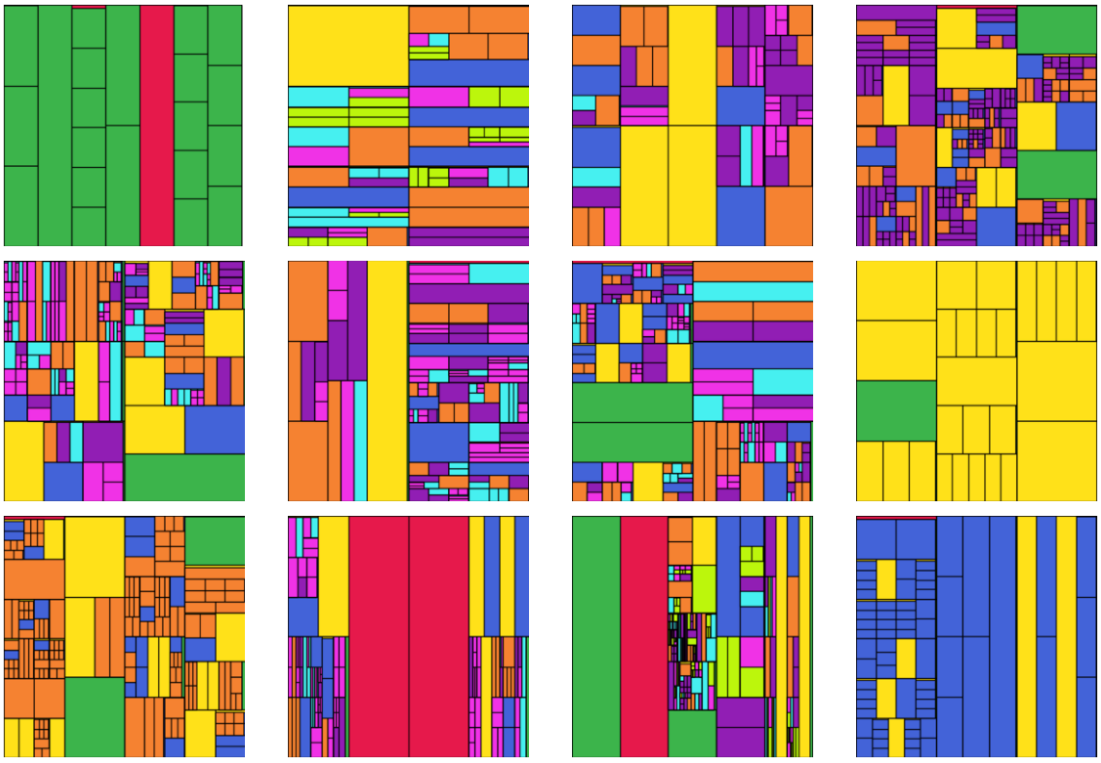

Figure 5.1.: A subset of 12 images from the extended pre-training data set. Some samples only have two or three colors, which indicates that the branches of the corresponding graphs have a very uniform depth. Other samples represent very unbalanced graphs and can have areas with a large concentration of different colored rectangles.

5.3. Results and Discussion

The results for structural argument retrieval using the **simple queries** data set of all visualizations V1-V5, using their respective models, can be found in Table 5.2.

Noticeably, the Completeness of all models and visualizations is 1.0. This is because the simulated MAC phase retrieves all relevant documents and a ViT model with a high enough latent dimension generally maps different inputs on different embeddings, similar to a hashing function. This means that $<$ generally defines a total order on the embeddings of the retrieved argument graphs, because of which every possible pair of the reference ranking is generally included in the produced ranking.

Data set	Source	Description
Kialo Graph-NLI	Agarwal et al. 2022	Graphs model discussion trees on Kialo, an online debates platform
Araucaria	Reed 2006	Corpus of analyzed argumentation, constucted using the Araucaria tool
IAC	Walker et al. 2012	A corpus for research on deliberation and debate
QT30	Hautli-Janisz et al. 2022	Argument and conflict in broadcast debate
US2016	Visser et al. 2020	Television debates and social media reactions to the 2016 US presidential elections
Persuasive Essays	Stab and Gurevych 2017	Annotated persuasive essays

Table 5.1.: Argument graph corpora used to construct my general training data set

However, the values of the Correctness metrics are much worse than those achieved using A*, showing that although the produced ranking includes the required pairs, their order often contradicts the reference ranking.

Regarding Ndcg, the deviations between different visualizations and models are very small, although V5 marginally outperforms the other visualizations. Compared to the values reported by Lenz et al. (2019), the advantage of vision-based similarities does not indicate better retrieval quality, but is explained by the simulated ideal MAC phase. As only the relevant argument graphs are retrieved, the value of Ndcg is skewed upward before the FAC phase even occurs. Because of this, an inverted expert ranking as the worst possible ranking achieves a Ndcg of 0.80644 (simple queries) 0.78402 (complex queries), which sets the lower bounds of Ndcg in this scenario.

Regarding Duration, it is apparent that the vision-based computation is much faster, saving around 91.5% (Exact Scheme Match) or 93.7% (Onto. Sim) of computation time, respectively, in comparison with A* search. When using a case base with more complex argument graphs, the time complexity for A* search scales much worse with graph complexity than the vision-based structural similarity pipeline (see also Section 5.6). V1, using *Graphviz'* dot engine is the slowest visualization design on average.

Comparing the Ndcg and Correctness of **fine-tuned** models to those, which are only **pre-trained**, no general statement about the effects of fine-tuning on retrieval quality can be made. This is because for some visualizations fine-tuning results in an improved retrieval quality (V1, V5), while there are other visualizations where it even hurts quality (V3, V4).

The results for structural argument retrieval using **complex queries** data set of all visualizations V1-V5, using their respective models, can be found in Table 5.3.

For the same reason as with the simple queries data set, the completeness is 1.0 regardless of the visualization or model used. However, there are large gains in regard to

Table 5.2.: Results of structure-based retrieval using V1-V5 on simple queries in comparison to the best results of previous works including argumentation schemes. The vision models reported include only pre-trained models (PT), as well as pre-trained and fine-tuned models (FT).

Experiment	Ndcg	Correctness	Completeness	Duration(s)
V1-PT-simple	0.908	0.064	1.000	1.776
V1-FT-simple	0.911	**0.095**	1.000	2.277
V2-PT-simple	0.913	0.013	1.000	1.608
V2-FT-simple	0.905	0.025	1.000	1.613
V3-PT-simple	0.901	0.027	1.000	1.596
V3-FT-simple	0.901	−0.011	1.000	**1.592**
V4-PT-simple	0.908	0.078	1.000	1.632
V4-FT-simple	0.907	0.043	1.000	1.608
V5-PT-simple^{+}	**0.914**	0.066	1.000	1.617
V5-FT-simple^{+}	**0.914**	0.068	1.000	1.630
DV $\bigoplus$ FT $\bigoplus$ DV (Ex. Scheme Match)*	0.859	**0.252**	0.943	**19.830**
DV $\bigoplus$ FT $\bigoplus$ DV (Onto. Sim)*	**0.861**	0.216	0.943	27.096

* as reported by Lenz et al. (2019)
$^{+}$ evaluation performed using V4 models

Ndcg and especially Correctness. Noticeably, V2-FT outperforms the other visualization in all retrieval quality metrics, while also providing the fastest retrieval time. For this visualization, fine-tuning also brings large quality gains, representing the largest fine-tuning gain of 0.04 for Ndcg and 0.2 for correctness.

Comparing V5 to V3-V4 clearly shows that **S-nodes** treemaps capture the structure of an argument graph much better than **I-nodes**. However, the poor retrieval quality of V4 compared to V3, which encodes exactly the same information using more desaturated colors, is unexpected.

Comparing the Ndcg and Correctness of **fine-tuned** models to those, which are only **pre-trained**, fine-tuning increases both metrics for almost every visualization design (V1-V4) and only the Ndcg value for V5 drops.

Impact of graph complexity

Comparing the simple query results to the complex query results, we see that the retrieval duration scales linearly with the number of requests, while the small increase in query complexity does not have any noticeable effects. It is also apparent that the visualizations and vision models perform much better (with respect to Correctness) on the complex query data set, especially using fine-tuned models, resulting in gains from 0.08 to 0.72. Even considering Ndcg, for all visualizations except V4 retrieval quality is improved on the complex queries with gains from 0.002 up to 0.2.

This is expected as the complex queries carry more information which can be visualized and embedded. Especially for the trivial cases, with 0 S-nodes, V1 and V2 only visualize a single blue circle, which does not enable the derivation of any meaningful graph structure, as every argument graph form the corpus contains at least one I-node. Therefore, the similarity calculated between this visualization and the graphs in the case base is arbitrary and does not carry any meaning.

This problem also occurs for the treemap visualization V3-V5. For the 12 queries with one S-node, V3-V5 produce a blank (white) image, and for the queries with one S-node, they produce a unicolored image, providing no information that may help the vision model to compare the queries to the case base graphs structurally.

This shows that ViT models need a minimum of information encoded in the visualization of the queries to be able to perform a good retrieval. Overall, hypothesis H1 has

Table 5.3.: Results of structure-based retrieval using V1-V5 on complex queries. The vision models reported include only pre-trained models (PT), as well as pre-trained and fine-tuned models (FT).

Experiment	Ndcg	Correctness	Completeness	Duration(s)
V1-PT-complex	0.954	0.535	1.000	1.253
V1-FT-complex	0.958	0.687	1.000	0.948
V2-PT-complex	0.931	0.536	1.000	0.976
V2-FT-complex	**0.977**	**0.751**	1.000	**0.950**
V3-PT-complex	0.904	0.308	1.000	0.966
V3-FT-complex	0.913	0.432	1.000	0.962
V4-PT-complex	0.888	0.155	1.000	0.968
V4-FT-complex	0.888	0.157	1.000	0.963
V5-PT-complex$^+$	0.933	0.381	1.000	0.993
V5-FT-complex$^+$	0.924	0.404	1.000	0.965

$^+$ evaluation performed using V4 models

to be discarded as the vision-based structural similarity pipeline led to a much worse retrieval quality than A* search for simple queries, with even the best correctness value representing a decrease of about 50%. However, it is plausible, when looking at the large gains in retrieval quality for complex queries, that vision-based similarity computation can lead to a retrieval quality that is at least similar to A* search, if the AG and therefore the visualization yield enough information to sufficiently distinguish the AGs.

Hypothesis H2 can clearly be accepted based on the durations reported in Tables 5.2 and 5.3 and the scaling which can be seen in Section 5.6.

Hypothesis H3 can be partly accepted, as the results show that the visualization design significantly impacts the retrieval quality for complex queries, resulting in a Correctness gain of 0.596 and a Ndcg gain of 8.9% between V4-FT and V2-FT as the worst and best fine-tuned models, respectively. However, in the case of simple queries, the Ndcg values

do not differ significantly (maximally by 1.3%), which is why H3 has to be discarded for these queries.

Hypothesis H4 has to be discarded, as there are two experiments, where the pre-trained model clearly outperformed the fine-tuned model for simple queries. However, because almost all visualizations experienced an advantage from fine-tuning on the complex queries, it is plausible that H4 could be accepted for even more complex AGs.

Finally, it should be noted that the information loss that comes from discarding the S-nodes for V3 and V4, or discarding the I-nodes in V5 is clearly detrimental to the retrieval quality for complex queries, which results in V1 and V2 clearly outperforming V3-V5 even though their training behavior was superior (see Appendix A) which shows that the training behavior, even when using identical model architectures, is a bad indicator for downstream task performance in this case.

5.4. Retrieval Performance for Extended Training

To evaluate the improvements from longer training, I trained a Swin Transformer v2 from scratch using a much larger pre-training data set (see Section 5.2.2) and more epochs (1000 instead of 100) for fine-tuning. The model is trained on V4 visualizations and is compatible with V4 and V5 images.

Results

Table 5.4.: Results of structure-based retrieval using V5 after extended training

Experiment	Ndcg	Correctness	Completeness	Duration(s)
V5-ext-PT-simple	**0.919**	**0.109**	1.000	1.801
V5-ext-FT-simple	**0.919**	0.077	1.000	**1.596**
V5-ext-PT-complex	0.945	0.553	1.000	1.057
V5-ext-FT-complex	**0.955**	**0.649**	1.000	**0.957**

It can be clearly seen that the extended training improves the retrieval quality. For simple queries, it improves Ndcg by 0.005 and Correctness by at least 0.009; for complex queries, it improves Ndcg by at least 0.012 and Correctness by at least 0.172. Regarding simple queries, the pre-trained extended model using V5 graph images is able to improve all retrieval quality metrics and even surpasses V1-FT-simple in terms of correctness. However, for simple queries, fine-tuning does not increase the retrieval quality of the extended model, even though it did so for the base model, where V5-FT outperformed V5-PT for simple queries.

For complex queries, the extended training improved on the results of the base model; however, both models (V5-ext-PT and V5-ext-FT) were still not able to overcome the original V2-FT.

5.5. Qualitative Impact of Graph Changes on Vision-based Similarities vs. A* Similarities

To compare the quality of the structural similarity of argument graphs produced by my ViT approach against those produced by the A* computation, I chose a base argument graph from the argumentative microtexts corpus and produced four derivative graphs with a different alteration of a single S-node. The base graph and its derivatives can be seen in Figure 5.2, D1-D3 are produced by swapping one of the 3 S-nodes; D4 is produced by eliminating the only positive S-node and its child. This results in D1 and D3 still having two attack S-nodes and one support S-node, but with different placements, while D2 and D4 now only have attack S-nodes (3 S-nodes for D2, 2 S-nodes for D4).

The structural similarity calculation for ViT similarities is carried out using visualization V5 (the resulting graph visualizations can be found in Figure 5.3) and the extended fine-tuned Swinv2 model from Section 5.4. The similarity values for A* are computed as the normalized sum of node and edge similarities between the derivative graph as query argument (Q) and the base graph as case argument (C) as described by Lenz et al. (2019). In this case, the best mapping between the graphs is the identity: $m : Q \rightarrow C = \mathbb{1}$. The resulting similarity values for both approaches can be found in Table 5.5. Analyzing

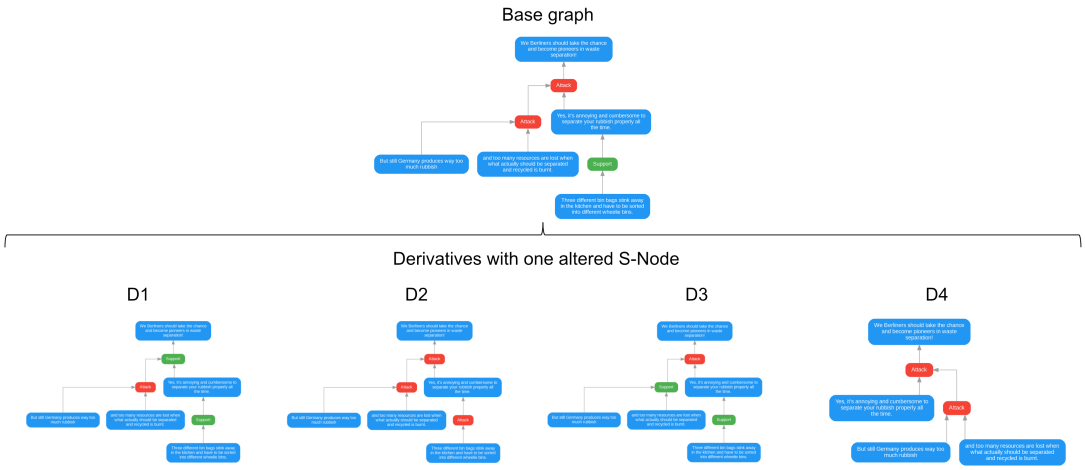

Figure 5.2.: Base graph and selected derivatives with one altered S-nodes for a qualitative study of vision-based similarities

these similarities, it can be seen that the embeddings produced by the ViT model tend to be very similar, hence the computed similarities are all very high and do not differ before the third decimal. Because all derivatives can be produced from the base graph with the alteration of a single S-node, the almost equally high similarity for all derivatives could be conceptually more reasonable than the A* similarity, which clearly separates D4 from D1-D3 in this case. However, when comparing the base graph with the more complex

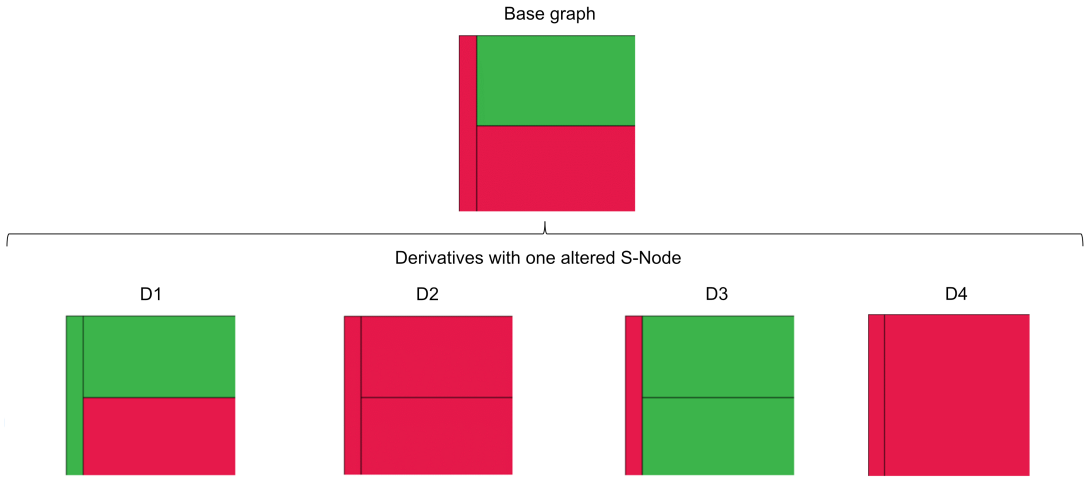

Figure 5.3.: Resulting V5 visualizations of the base graph and its derivatives for a qualitative study of vision-based similarities

Derivative	Sim. to Base Graph (V5)	Sim. to Base Graph (A*)
D1	0.992 75	0.866 67
D2	0.991 83	0.866 67
D3	0.992 84	0.866 67
D4	0.991 823	0.733 33
M (Figure 5.4)	0.991 561	
C (Figure 5.4)	0.991 932	

Table 5.5.: Impact on the similarity when altering one graph node, using the vision-based approach vs. A*

graphs shown in Figure 5.4, the medium graph still produces a similarity of 0.99156 and the complex graph 0.99193. This unexpectedly makes the complex graph more similar to the base graph than the medium graph or D4.

Although the complex graph is probably already too complex for the V5 visualization with only an area of 256×256px (some of the areas already get so small that they form overarching black areas instead of being distinguishable as red or green), this also highlights the problem of unexpected interactions between embeddings produced by a Transformer model.

Together with the assumption that ViT models need a minimum level of information to be able to produce meaningful embeddings from Section 5.3, this suggests that there is an optimal query complexity for vision-based structural similarity computation. Below this optimal complexity, queries do not provide enough information, rendering the computed similarities arbitrary, and above this optimal complexity, the space-confined

visualizations and corresponding ViT models fail to clearly distinguish between different graphs.

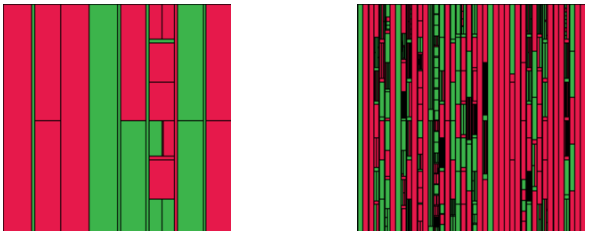

Figure 5.4.: V5 visualization of a medium graph (M) with 29 (left) and a complex graph (C) with 693 (right) S-nodes from the Kialo-GraphNLI data set (Agarwal et al. 2022)

5.6. Processing large Argument Graphs using a Vision-based Approach vs. A*

In this study, I evaluate how the graph complexity (measured by the number of the graph's S-nodes) affects the computation time of structural similarity. Let x, y be argument graphs with $|x|, |y|$ denoting their respective numbers of S-nodes.

To study graph complexity scaling, I chose 53 graphs from the Kialo GraphNLI data set (Agarwal et al. 2022) making up the set of queries Q with $n = |x| \in \{4, \ldots, 57\}$ for $x \in Q$. The set of case base arguments CB consists of a single argument graph y with $|y| = 20$. As the case base graph's complexity is constant, this setup allows studying the impact of increasing graph complexity on retrieval time in isolation. In this study, visualization V5 and the extended fine-tuned Swinv2 model from Section 5.4 are used to represent the vision-based approach. A*-based similarity computation is represented by its implementation in *Arguelauncher*[3]. The MAC phase and evaluation are disabled to only measure the time needed for the graph-based similarity computation.

Vision-based similarity computation requires the 3 steps outlined in Section 4.3: visualization, embedding, and cosine similarity calculation. The scaling behaviors of each of these steps can be seen in Figure 5.5. The embedding step, as well as the cosine similarity calculation, require constant time and are not influenced by the complexity of the input graphs. The visualization time increases linearly with graph complexity, even though there are several outliers. These could be caused by deviations in the size of the AG files, which are completely loaded, although only the information about the S-nodes is considered to visualize the AG.

For a practical implementation of an argumentation machine, the linear scaling of visualization in respect to graph complexity is likely not a problem, as only the query has to be visualized, whereas the case base graphs visualizations and embeddings can be precomputed at store time. Additionally, the visualization time increases relatively slowly,

[3] `https://github.com/recap-utr/arguelauncher`

by about 0.96ms per S-node, which means that even a very large argument graph with 10000 S-nodes should only take about 9.65s, making the system capable of visualizing and processing very complex queries for structural argument retrieval.

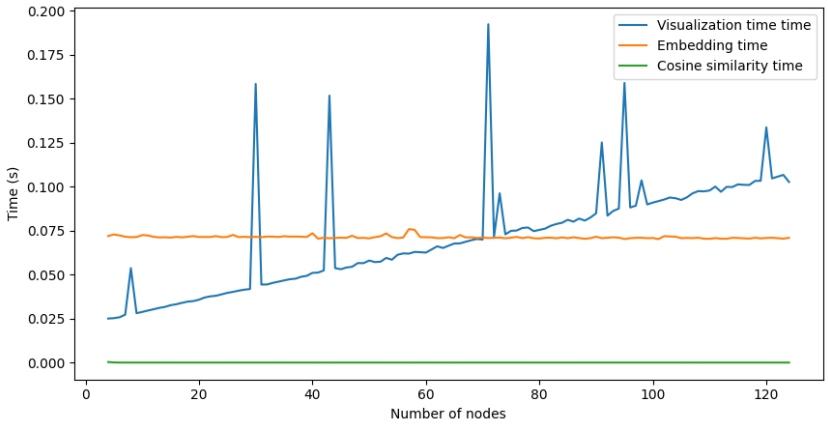

Figure 5.5.: Computation times for the separate steps of vision-based structural graph similarity computation for argument graphs with 4 to 124 S-nodes

For comparison, Figure 5.6 shows the full structural similarity computation for both the vision-based approach and and compares it to the graph-based A* similarity computation. Although both techniques struggle with outliers of longer computation time, the computation time for the vision-based approach is far more consistent, as it has only one noticeable spike, where the argument graph with 8 S-nodes takes 54% longer than its predecessor. On the other hand, the largest of the 4 spikes for A* increases the time needed to process the graph with 37 S-nodes by 4000% over its predecessor.

Regarding the absolute times for both approaches, it is apparent that A* is not viable for retrieval of complex arguments in a production argumentation machine, as a single comparison between an argument graph with 20 S-nodes, and an argument graph with more than 11 S-nodes takes at least 50s. After 30 S-nodes, the time for a single similarity computation explodes to over 1000s.

5.7. Limitations

The following limitations should be noted regarding the results of the evaluation above:

Regarding the compatibility between V4 and V5

Based on the assumption that models trained on V4 should be compatible with V5, I used the models trained with V4 samples for the evaluation of V5 images. However, there are two crucial differences between the visualizations. V4 images use more colors (9 colors, representing node depth) than V5 (2 colors, representing the type of S-nodes) in my implementation; additionally, in visualization design V5 parent nodes always claim

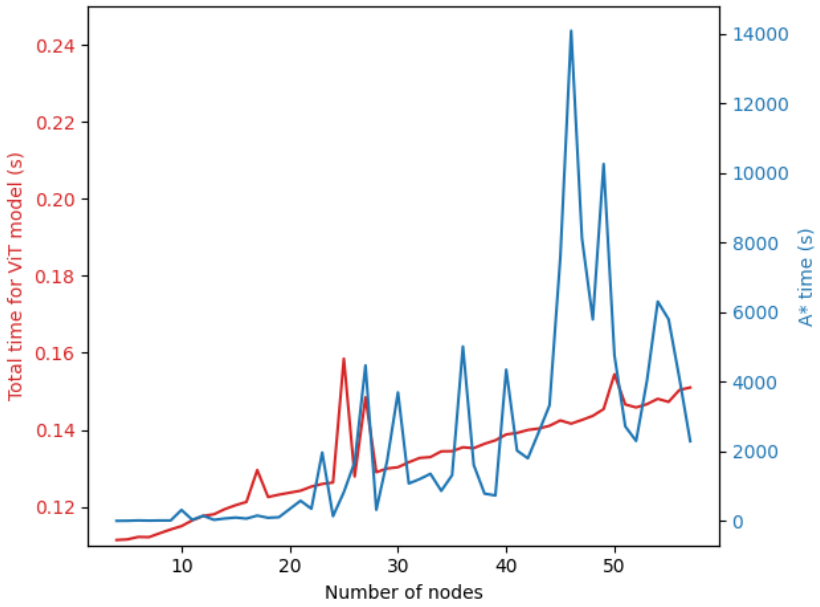

Figure 5.6.: Comparison of structural similarity computation time between vision-based approach and A*. Note: Vision-based time is scaled from 0.1 to 0.25 s, but A* is scaled from 0 to 14000s.

10% of the available space instead of being fully divided into the areas of their children, creating a different overall layout.

Training a vision model for visualization V5 with the bigger color selection of V4 could improve generalization for additional, similar treemap representations, but could also result in the vision model using only a limited part of the entire latent space to represent V5 images, which is known to produce sub-optimal results (El-Nouby et al. 2021). Because of this risk and the slightly different overall layout, it seems probable that a higher retrieval quality could be achieved through pre-training, or at least fine-tuning a vision model on V5 specific images.

Regarding the retrieval quality for large argument graphs

The scaling study in Section 5.6 disregards the quality of the retrieval for these larger AGs. It is probable that the competitive results for small graphs (Section 5.3) do not transfer directly to large AGs. Especially, the increasingly small treemap rectangles for deeper nodes of large argument graphs might make distinguishing between two different AGs using a vision model very hard, and could therefore be detrimental to the retrieval quality achievable in this setting.

Regarding visualization design

As shown in the evaluation, the performance in terms of precision and recall of my approach depends heavily on the visual representation of the argument graphs used. In order to represent the graphs in a compressed format, I made simplifications, for the traditional graph drawing and especially for the treemap visualizations. These simplifications introduce information loss, and the visualizations examined in this thesis might be subpar, depending on the concrete task. Especially, AGs which contain cycles cannot be displayed with the tree-based visualizations V3-V5 that I explored, even if AIF explicitly allows them (Chesñevar et al. 2006). Furthermore, as highlighted in Section 5.6 and Section 5.5, the treemap designs evaluated in this thesis struggle to represent very large AGs in the limited input window of a vision model, because the areas allocated for deeper nodes decreases rapidly.

Regarding model architecture and training data sets

While I already show in Section 5.4 that a longer training period can improve the model's retrieval quality, I expect there to be an additional improvement in increasing the model size and constructing a larger, more comprehensive data set for pre-training and fine-tuning the models. Especially, compared to work focused on contemporary vision models (Z Liu et al. 2021a; Oquab et al. 2024), a model size of only 27.6M parameters and training data sets with around 5000 or 416K samples, respectively, may severely limit the capabilities of the resulting ViT model.

6. Conclusion and Future Work

In this thesis, I proposed a vision-based structural similarity pipeline for argument graph retrieval, which works by visualizing AGs, embedding these visualizations with a vision model, and computing the k-nearest neighbors through cosine similarity. The research question whether vision-based argument retrieval can provide a faster and more scalable alternative to A* search for structural AG retrieval can be affirmed; however, not every use case allows for the effective use of the vision-based approach.

Concretely, this thesis shows that it is possible to conceptualize specialized visualization designs, which are capable of capturing the structure of AGs to some degree. These visualizations can then be used to train a ViT model to deliver a promising retrieval quality for complex argumentative microtexts queries. However, the finding in the evaluation suggests that there may be a limited ideal graph complexity range, defined by the number of S-nodes of an AG, below and above which the computed similarities from the pipeline only have limited significance. Furthermore, the training of the ViT models becomes unpredictable under this threshold, as can be seen in the decreased retrieval quality of some fine-tuned models for the simple microtexts queries.

Regarding efficiency and scaling, the use of embeddings allows storing a uniform, query-independent representation of the original argument, which can be pre-computed to allow for fast comparisons even across large case bases. Because of this, the vision-based pipeline is able to improve computation times for complex graphs (see Section 5.6) by several orders of magnitude. This opens up the new possibility of utilizing the vision-based pipeline as a structural similarity MAC phase to pre-filter the AGs on a structural level, which could then be ranked by the more accurate A* search. Especially in use cases, where no contentual pre-filtering can be performed, because the user does not want to semantically limit the retrieval results, this might prove very useful and could be explored in future work. However, the qualitative study performed as part of my evaluation also showed the pitfall of Transformer models, where the model output may be inexplicable, potentially resulting in poor recall for a subsequent A*-based FAC phase.

Future Work Incorporating Argumentation Schemes

Argument schemes can be seen as local argument structures, representing the relation between two ADUs (see Section 2.1). However, in this thesis, the exact argumentation schemes which might have been applied in particular arguments were not used; instead, only the general type of S-nodes (i.e., support or attack) influenced the visualization of an AG. Through this simplification, potentially important information is lost, which might be highly relevant when trying to retrieve arguments based on a convincing

argumentation structure. For example, a user might search for arguments where an *argument from analogy* is challenged by an *argument from a position to know*.

Although visualizations V1, V2 and V5 could support encoding the labels of S-nodes, e.g., through the use of different colors, the visualization designs and the corresponding visualization codes would have to be adapted to fit this requirement. This also brings new challenges, as encoding each known argumentation scheme using individual, equidistant colors might make distinguishing these colors from each other too challenging for a vision model, similarly to the hues of blue used for V3. Because of this, the use of an ontology to only represent the most important supercategories of argumentation schemes appears more promising. To implement this, the previous work of Lenz et al. (2019) and Walton and Macagno (2015), as well as the literature on visualization design cited in this thesis, can be considered.

Future Work on Visualization Design

As noted in the limitations of the evaluation, the results achievable through vision-based argument retrieval depend heavily on the visualization design used to represent the structure of the argument graphs. Future work on suitable visualization designs should focus on optimally using the input windows of vision models by preventing large amounts of white-space similar to treemaps. However, they should ideally also allocate the same area for each visual feature (similar to how node-link graph drawings use uniform node sizes) to prevent bias caused by overly large areas compared to the decreasing size of the areas for deeper graph nodes, as exhibited by visualizations V3-V5. Finding an optimal visualization design could improve the retrieval quality of the evaluated argumentative microtexts corpus, but would be especially relevant for processing very large AGs.

There exists literature on visualizing large graphs (Hu and Shi 2015; Kornaropoulos and Tollis 2012; J Tang et al. 2016; Wills 1999) and *GrouseFlocks* (Munzner 2014) or similar hierarchical visualizations could also be considered. However, visualization designs are not focused on optimal interpretability for a vision model. Several works that analyze the inner workings of ViT models (Bhojanapalli et al. 2021; Ma et al. 2023; Zhou et al. 2022) may provide additional insights for suitable visualizations beyond those reached in this thesis. Furthermore, alternative layout engines, such as PBF (Lionakis et al. 2023)) could be explored.

Future Work on Optimizing the Vision Model

As mentioned in the limitations of the evaluation carried out in this thesis, the ViT model itself (in terms of the number of training parameters) and the training data set were very small. Due to the impressive scaling of ViT models with respect to both of these properties (Dosovitskiy et al. 2021; Z Liu et al. 2021a), increasing both could result in a noticeable increase in retrieval quality, without changing the training recipe or the operating principle of the vision-based retrieval pipeline. Especially swapping the Swin Transformer v2 configuration from SwinV2-Tiny (27.6M parameters) to SwinV2-L (197M parameters) or SwinV2-G (3B parameters) (Z Liu et al. 2021a) and synthetically

generating a bigger pre-training data set (e.g. similar to the procedure described in Section 5.2.2 for the extended training) would probably be the simplest way to accomplish this. Additionally, argument mining approaches (Lenz et al. 2020) could be used to create a larger training corpus based on real arguments.

However, there are also many techniques to improve contrastive learning performance that were not evaluated in this thesis, including hard-mined and cross-batch negatives (Asai et al. 2022; Khan et al. 2022; Qu et al. 2020; L Wang et al. 2022a) and regularization of the output feature space (El-Nouby et al. 2021). Through these techniques, the training of the ViT model should result in a more effective embedding model and increase the retrieval quality.

Furthermore, contrastive learning holds the following challenges:

- For this approach to be effective, the batch sizes have to be very large (preferably between 512-4096, T Chen et al. 2020a).
- A sample $x^- \in X^-$ may be a false negative, that is, x^- is actually similar to x, however, according to the training objective, it will be further apart from x in the embedding space.

These aspects can make contrastive training unstable and very resource-intensive. As an alternative, model distillation could be explored Grill et al. (2020) and Tian et al. (2021), eliminating the need for contrastive pairs.

Future Work on Optimizing the Retrieval Process

For the evaluations in this thesis, a precise K-NN search based on cosine similarity is implemented. This already helps to alleviate long search times of A* search, especially since the dimensionality is a fixed value (768 in my implementation), and cosine similarity is an operation which can be very efficiently parallelized on highly capable GPUs. However, for very large case bases, containing millions of argument graphs, the cosine similarity computation could become the bottleneck for argument retrieval. The use of fixed-size embeddings allows the use of modern specialized vector databases (e.g. Qdrant[1]) with efficient indexing structures based on approximate nearest-neighbor algorithms. These algorithms are very well studied (Arya et al. 1998; Indyk and Motwani 1998; T Liu et al. 2004; Malkov and Yashunin 2018) and should allow for a noticeable improvement in computation times for very large case bases if additional retrieval inaccuracies can be tolerated. This could be the key factor to enable structural argument retrieval on a very large scale.

[1] `https://qdrant.tech/`

A. Training results

The training for the models was performed as described in Section 5.2.1. Both pre-training and fine-tuning were performed on the same data set (see Table 5.1). The argumentative microtexts corpus was used as the validation data set. The model for every visualization was trained for 100 epochs on a single Nvidia V100 GPU. As V5 is assumed to be compatible to V4, no extra models are trained for this visualization design.

Note: Each data set was deduplicated using *fclones*[1] before being used in training. Because some visualizations contain more simplifications, they are more likely to map different argument graphs on the same image, resulting in their data sets containing more duplicates, which were eliminated before training. The resulting data sets can be found in Table A.1.

Visualization	Data set	Number of images
V1	kblw/graphimages_dot	4809
V2	kblw/graphimages_twopi	4793
V3	kblw/treemap_weak_ft	2323
V4	kblw/treemap_sat_ft	2587

Table A.1.: Data sets used for training the ViT models on visualizations V1-V4

The models are pre-trained using an Adam optimizer with a learning rate of 0.001 and a ReduceLROnPlateau scheduler. For fine-tuning, an AdamW optimizer (learning rate: 0.005) with a CosineAnnealingLR scheduler (t_max : 100, eta_min : 0.0001) is used. Figure A.1 shows the loss throughout the pre-training processes of V1-V4. In particular, the two treemap visualizations are more suitable for training a ViT model. They converge faster and maintain a much smaller reconstruction loss for the training and validation data set throughout the pre-training process. Nevertheless, the training for all models converges and, therefore, succeeds. There also seems to be no overfitting as the losses for the training and the validation data set are very similar. As for the fine-tuning runs, the results are not that clear. While random fluctuations decrease, no clear downward trend, and therefore no clear convergence of the models can be identified. Especially the validation loss stays nearly constant. V3 is the only visualization that shows an upward trend regarding the top-5 accuracy on the validation data set. Nevertheless, none of the fine-tuning runs shows satisfactory training behavior. As the training mechanism has been successfully validated on the CIFAR-10 data set (Krizhevsky 2009), which contains 60000 photos of real-life objects such as airplanes or trucks, this indicates that either

[1] `https://github.com/pkolaczk/fclones`

the visualizations themselves are not suitable for contrastive learning (e.g., because the images are too similar to one another), or the contrastive transformations used for the two different views of each image are not suitable. This is likely to be at least part of the problem, as Xie et al. (2022) have already shown that contrastive learning on images is very sensitive to the transformations used to create the different views and especially color-jitter is important, which cannot be used for my graph visualizations.

In the following sections, I will present the reconstruction behavior of the trained models during pre-training.

A.1. V1

When looking at the model reconstructions in Figure A.3, it is evident that the model successfully reconstructs the original samples (S1-S4). However, as assumed in Chapter 4, visualization features that are much smaller than others (in this case, the edges connecting the graph nodes) seem to be completely ignored. Another interesting observation to point out is that the model draws nodes in turquoise (especially when processing random inputs R1 and R2), even though this color is not present in any of the training samples. Furthermore, the uniform red input (P2) does not result in a reconstruction with mostly red nodes. However, the reconstruction of the color gradient (P4) as part of the image patterns shows that the model seems to develop a good sense of locality and is able to place the colors it is able to use in a similar location to the original. Although the patterns (P1-4) and random inputs (R1, R2) are outside the training distribution and share almost no similarity, most of their reconstructions (apart from PR3 and PR4) resemble graph drawings of densely packed argument graphs.

Challenges faced in Training

Overall generated image. As can be seen in Figure 4.1, the generated image contains the important characteristics of the actual graph structure consisting of graph elements (nodes and edges), while the background is white. Due to the Graphviz configuration (node sizes, node margins) and the layout algorithm, a large portion of the image is taken up by the uni-colored background, which carries no information.
Graph Layout. Depending on the graph, Graphviz' dot engine may produce very imbalanced graphs where one dimension of the resulting image is several times larger than the other (i. e. a very deep or very wide graph). As vision models expect a quadratic input size (e.g. 256x256 pixels), the image has to be resized so it can be processed. This leads to small image features being further compressed, further amplifying the first problem.

A.2. V2

Regarding model's reconstructions in Figure A.4, visualization V2 seems to result in improved reconstructions compared with V1, as more reconstructions for image patterns

are more consistent with expectations. Especially the uniformly red image (P2) results in a mostly red reconstruction, and the color gradient (P4) locality seems to have improved. Noticeably, because *twopi* is a radial layout model, which is apparent for large graphs, its reconstruction for out-of-distribution data (P1-P4, R1, R2) is centered in a circle around the image center. However, the graph edges are still ignored, and the model still produces colors that were not seen in training.

A.3. V3

Regarding model's reconstructions in Figure A.5, it is apparent that the desaturated colors used for small node depths of the original argument graph, together with the thin black lines separating different areas in the treemaps are not suitable to be consistently recognized by the ViT model (see SR1, SR4).

Because of this weakness, the model also fails to grasp which rules have to apply for a valid argument graph visualization (i.e., the visualization is composed only out of rectangles; this means every black line has to divide the entirety of its parent). Although the reconstructions of out-of-distribution images (especially P4, R1, R2) resemble argument graph visualizations, the model already fails to reconstruct 2 vertical lines for S1 and also fails to reconstruct any of the vertical lines for the more complex S4. On the topic of color, the model seems to produce only colors which are reasonably similar to the colors found in training samples and only color areas in hues of blue.

A.4. V4

Regarding model's reconstructions in Figure A.6, V3s problem of attending to the thin black lines separating two equally colored rectangles persists. However, when there is a high contrast (i.e., separation of a red from a green rectangle), the reconstruction is perfect, indicating that this plays a large role in the model's ability to differentiate two areas. Additionally, all reconstructions of the training samples seen in the figure are valid argument graph visualizations. From the argument graphs S1-S4, it also seems, as if visualization V4 reconstructs vertical separation more faithfully than horizontal separations.

On the topic of color, V4 seems to remain largely faithful to the colors found in the original training samples, only producing one image (P1) where turquoise is again produced even though it is not part of any sample.

A.5. V4, Extended Training

For pre-training, the same tiny Swinv2 model as previously is trained from scratch for 100 epochs on the kblw/treemap_sat data set. This data set was built by generating $\tilde{9}$00k random treemaps using the procedure described in Section 5.4, which were then

deduplicated to 415640 samples. 90% of the data set is used for training, while the remaining 10% is used as a validation data set.

The training behavior can be seen in Figure A.7 and shows a fast convergence, similar to the normal pre-training run, and a successful training. The similar losses for the training and the validation data sets are very similar, which shows the absence of overfitting here as well.

The fine-tuning is executed on the resulting model from the previous pre-training step. In contrast to the extended pre-training, the extended fine-tuning uses the same dataset (kblw/treemap_sat_ft) as the normal fine-tuning run, however, it is performed for 1000 epochs instead of 100 epochs.

Like in the normal fine-tuning run, no clear downward trend or convergence is visible. Although there is a spike in training loss at around 22k training steps, this spike does not affect validation loss. When looking at the steady upward trend in top-five accuracy on the validation data set, it is apparent that the training has an effect on the model's performance even though the absolute loss value is no good indication for this.

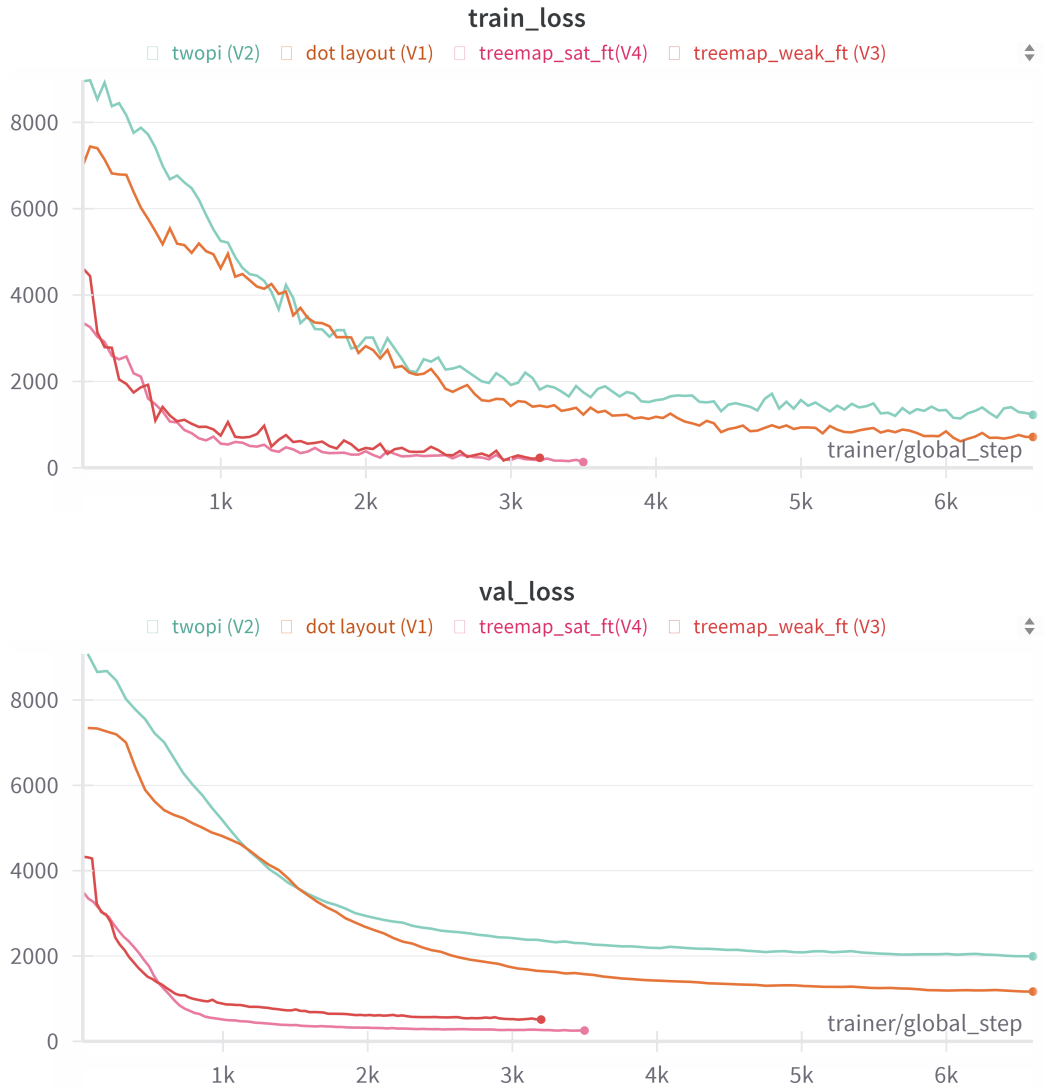

Figure A.1.: Training metrics during pre-training of the 4 Swin Transformer v2 models for V1-V4

Figure A.2.: Training metrics during fine-tuning of the 4 Swin Transformer v2 models for V1-V4

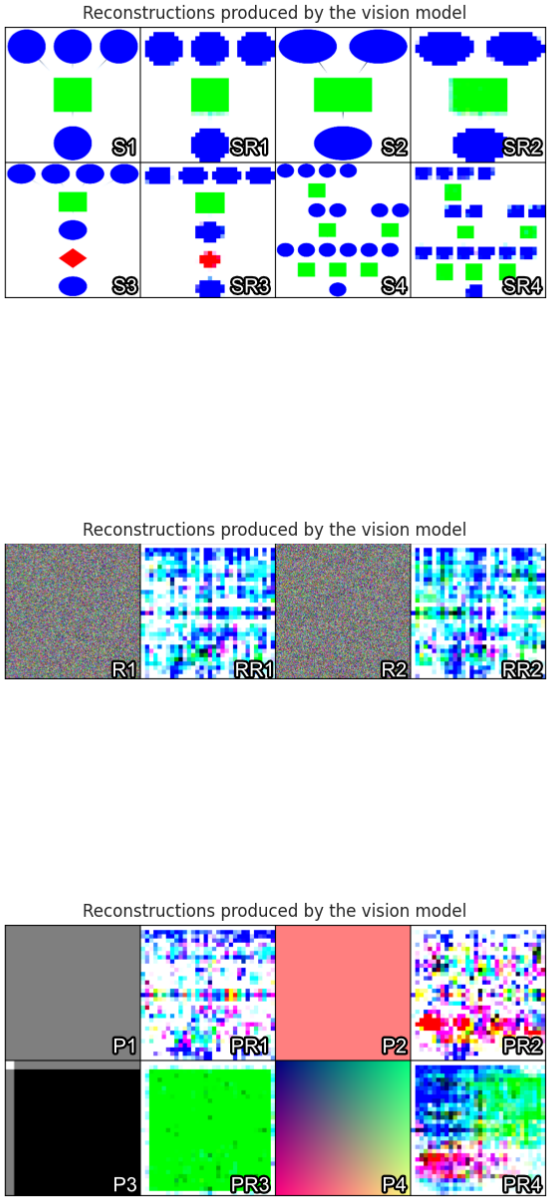

Figure A.3.: Training samples and their reconstructions from the pre-trained V1 Swin Transformer v2 model. Input images: training samples (S), random inputs (R) and patterns (P). SR, RR and PR mark the corresponding model reconstructions.

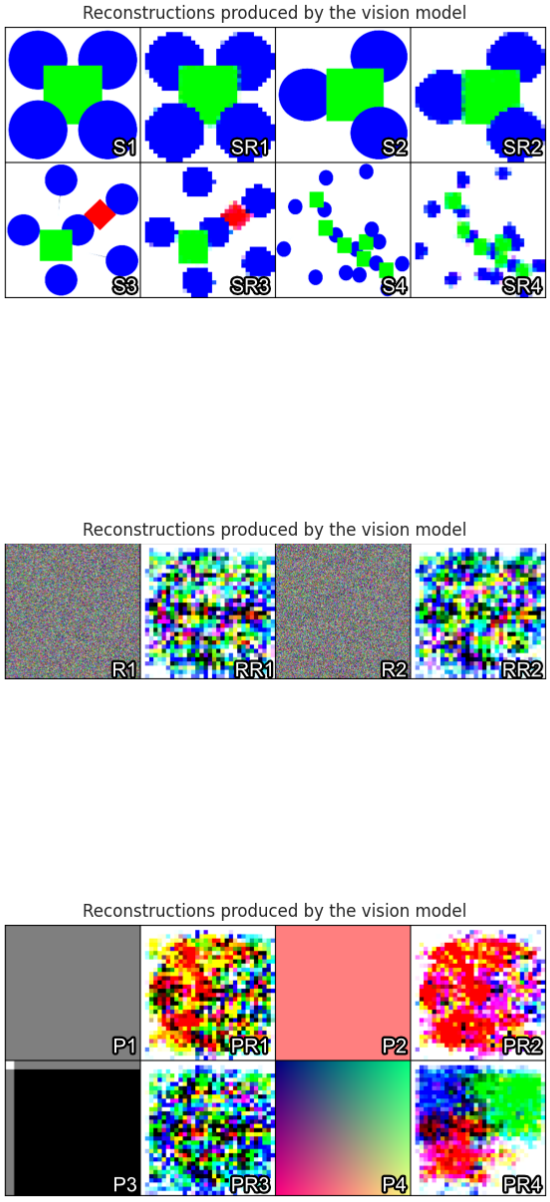

Figure A.4.: Training samples and their reconstructions from the pre-trained V2 Swin Transformer v2 model. Input images: training samples (S), random inputs (R) and patterns (P). SR, RR and PR mark the corresponding model reconstructions.

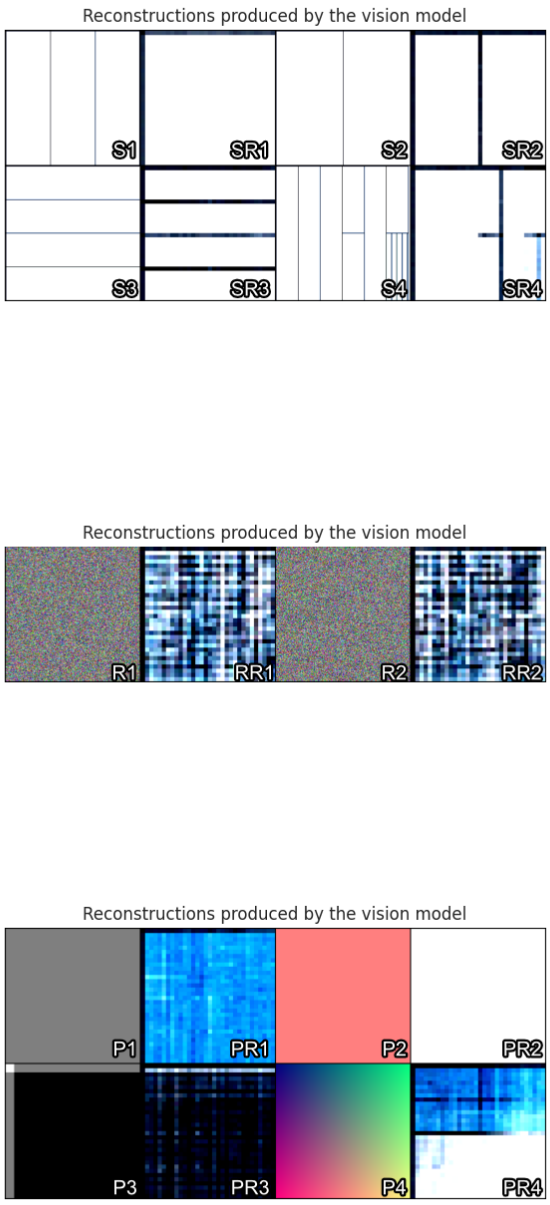

Figure A.5.: Training samples and their reconstructions from the pre-trained V3 Swin Transformer v2 model. Input images: training samples (S), random inputs (R) and patterns (P). SR, RR and PR mark the corresponding model reconstructions.

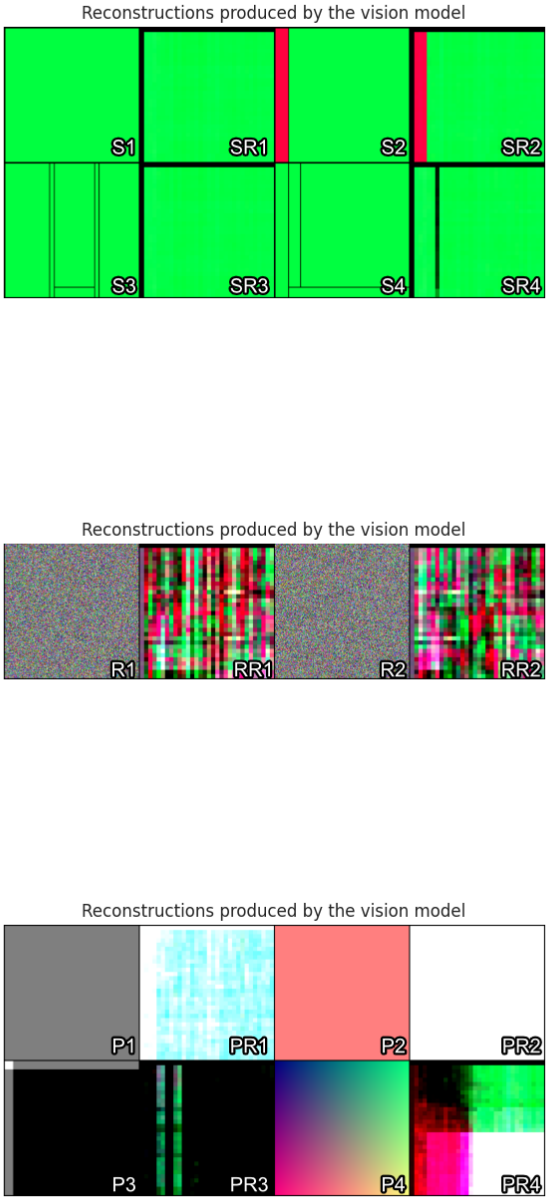

Figure A.6.: Training samples and their reconstructions from the pre-trained V4 Swin Transformer v2 model. Input images: training samples (S), random inputs (R) and patterns (P). SR, RR and PR mark the corresponding model reconstructions.

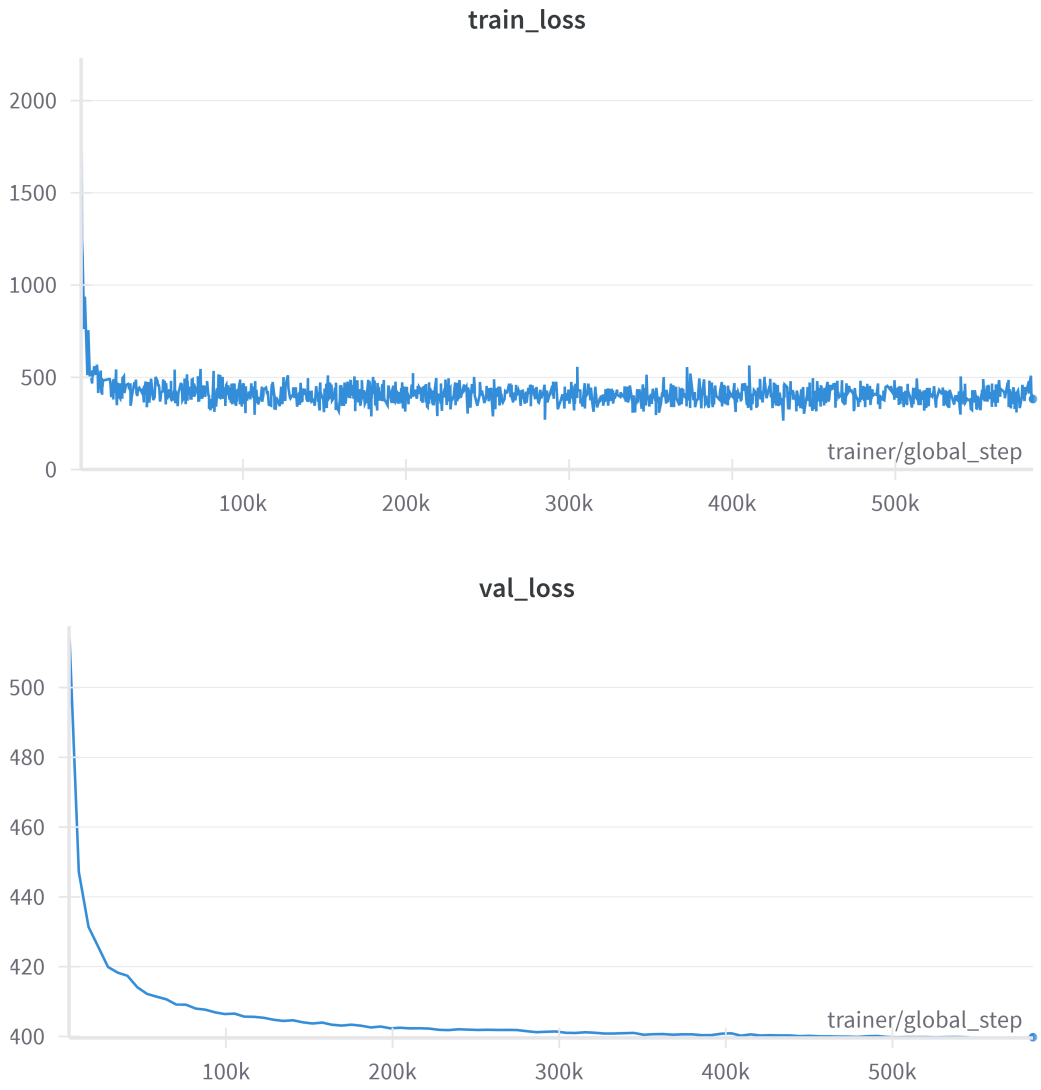

Figure A.7.: Training behavior of extended pre-training for V4

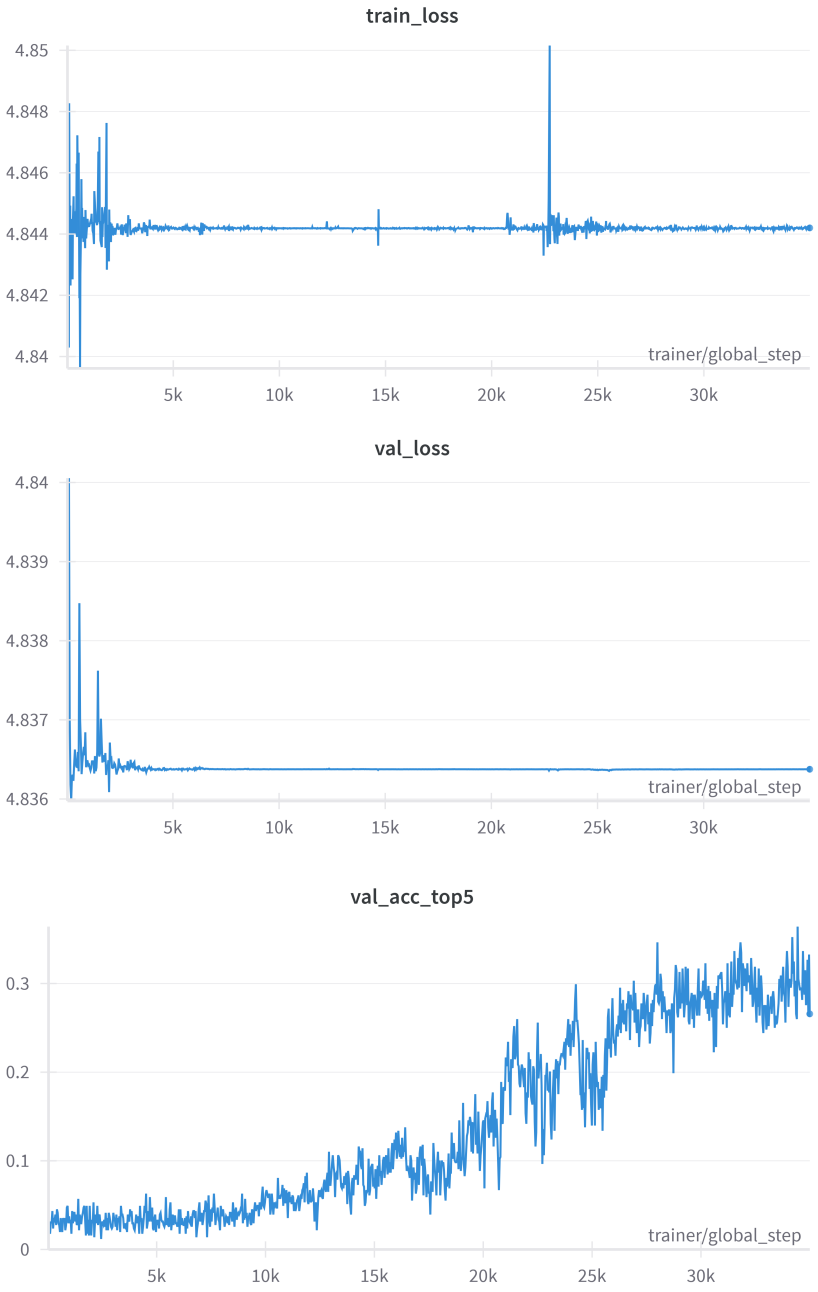

Figure A.8.: Training behavior of extended fine-tuning for V4

Bibliography

Aamodt A and Plaza E (2001). "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches". In: *AI Communications* 7, pp. 39–59. DOI: 10.3233/AIC-1994-7104.

Agarwal V, Joglekar S, Young AP, and Sastry N (2022). "GraphNLI: A Graph-based Natural Language Inference Model for Polarity Prediction in Online Debates". In: *The ACM Web Conference* (*TheWebConf*).

Angelini P and Hanxleden R von, eds. (2023). *Graph Drawing and Network Visualization - 30th International Symposium, GD 2022, Tokyo, Japan, September 13-16, 2022, Revised Selected Papers*. Vol. 13764. Lecture Notes in Computer Science. Springer. DOI: 10.1007/978-3-031-22203-0.

Arya S, Mount DM, Netanyahu NS, Silverman R, and Wu AY (1998). "An optimal algorithm for approximate nearest neighbor searching fixed dimensions". In: *Journal of the ACM* (*JACM*) 45.6, pp. 891–923.

Asai A, Schick T, Lewis P, Chen X, Izacard G, Riedel S, Hajishirzi H, and Yih Wt (2022). *Task-aware Retrieval with Instructions*. arXiv: 2211.09260 [cs.CL].

Bao H, Dong L, and Wei F (2021). "BEiT: BERT Pre-Training of Image Transformers". In: *CoRR* abs/2106.08254. arXiv: 2106.08254.

Bekos MA and Chimani M, eds. (2023). *Graph Drawing and Network Visualization - 31st International Symposium, GD 2023, Isola delle Femmine, Palermo, Italy, September 20-22, 2023, Revised Selected Papers, Part I*. Vol. 14465. Lecture Notes in Computer Science. Springer. DOI: 10.1007/978-3-031-49272-3.

Bergmann R, Althoff KD, Minor M, Reichle M, and Bach K (2009). "Case-Based Reasoning - Introduction and Recent Developments". In: *Künstliche Intelligenz* 1/2009, pp. 5–11.

Bergmann R and Gil Y (2014). "Similarity assessment and efficient retrieval of semantic workflows". In: *Information Systems* 40, pp. 115–127.

Bergmann R, Lenz M, Ollinger S, and Pfister M (2019). "Similarity Measures for Case-Based Retrieval of Natural Language Argument Graphs". en. In.

Bergmann R, Schenkel R, Dumani L, and Ollinger S (2018). "ReCAP-Information Retrieval and Case-Based Reasoning for Robust Deliberation and Synthesis of Arguments in the Political Discourse." In: *LWDA*, pp. 49–60.

Bhojanapalli S, Chakrabarti A, Glasner D, Li D, Unterthiner T, and Veit A (2021). "Understanding robustness of transformers for image classification". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10231–10241.

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S,

Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, and Amodei D (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].

Cai H, Zheng VW, and Chang KCC (2018). "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9, pp. 1616–1637.

Caron M, Touvron H, Misra I, Jégou H, Mairal J, Bojanowski P, and Joulin A (2021). "Emerging Properties in Self-Supervised Vision Transformers". In: *CoRR* abs/2104.14294. arXiv: 2104.14294.

Chen M, Radford A, Child R, Wu J, Jun H, Luan D, and Sutskever I (2020). "Generative pretraining from pixels". In: *International conference on machine learning*. PMLR, pp. 1691–1703.

Chen T, Kornblith S, Norouzi M, and Hinton GE (2020a). "A Simple Framework for Contrastive Learning of Visual Representations". In: *CoRR* abs/2002.05709. arXiv: 2002.05709.

Chen T, Kornblith S, Swersky K, Norouzi M, and Hinton GE (2020b). "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *CoRR* abs/2006.10029. arXiv: 2006.10029.

Chesñevar C, Mcginnis, Modgil S, Rahwan I, Reed C, Simari G, South M, Vreeswijk G, and Willmott S (2006). "Towards an argument interchange format". en. In: *The Knowledge Engineering Review* 21.4, pp. 293–316. DOI: 10.1017/S0269888906001044.

Choy KL, Lee WB, and Lo V (2002). "Development of a case based intelligent customer–supplier relationship management system". In: *Expert systems with Applications* 23.3, pp. 281–297.

D'Zmura M (1991). "Color in visual search". In: *Vision research* 31.6, pp. 951–966.

Deng J, Dong W, Socher R, Li LJ, Li K, and Fei-Fei L (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Devlin J, Chang M, Lee K, and Toutanova K (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, and Houlsby N (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: 2010.11929 [cs.CV].

Forbus KD, Gentner D, and Law K (1995). "MAC/FAC: A Model of Similarity-Based Retrieval". In: *Cognitive Science* 19.2, pp. 141–205. DOI: https://doi.org/10.1207/s15516709cog1902"˙1.

Gao L and Callan J (2021). "Is Your Language Model Ready for Dense Representation Fine-tuning?" In: *CoRR* abs/2104.08253. arXiv: 2104.08253.

Gao T, Yao X, and Chen D (2021). "SimCSE: Simple Contrastive Learning of Sentence Embeddings". In: *CoRR* abs/2104.08821. arXiv: 2104.08821.

Girdhar R, El-Nouby A, Liu Z, Singh M, Alwala KV, Joulin A, and Misra I (2023). *ImageBind: One Embedding Space To Bind Them All*. arXiv: 2305.05665 [cs.CV].

Grill J, Strub F, Altché F, Tallec C, Richemond PH, Buchatskaya E, Doersch C, Pires BÁ, Guo ZD, Azar MG, Piot B, Kavukcuoglu K, Munos R, and Valko M (2020). "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning". In: *CoRR* abs/2006.07733. arXiv: 2006.07733.

Grover A and Leskovec J (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

Harrower M and Brewer CA (2003). "ColorBrewer. org: an online tool for selecting colour schemes for maps". In: *The Cartographic Journal* 40.1, pp. 27–37.

Hautli-Janisz A, Kikteva Z, Siskou W, Gorska K, Becker R, and Reed C (2022). "QT30: A Corpus of Argument and Conflict in Broadcast Debate". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 3291–3300. URL: https://aclanthology.org/2022.lrec-1.352.

Healey CG (1996). "Choosing effective colours for data visualization". In: *Proceedings of Seventh Annual IEEE Visualization'96*. IEEE, pp. 263–270.

Holten D and Van Wijk JJ (2009). "A user study on visualizing directed edges in graphs". In: *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 2299–2308.

Hu Y and Shi L (2015). "Visualizing large graphs". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 7.2, pp. 115–136.

Indyk P and Motwani R (1998). "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613.

Karim RM, Kwon OH, Park C, and Lee K (2019). "A study of colormaps in network visualization". In: *Applied Sciences* 9.20, p. 4228.

Karpukhin V, Oğuz B, Min S, Lewis P, Wu L, Edunov S, Chen D, and Yih Wt (2020). "Dense passage retrieval for open-domain question answering". In: *arXiv preprint arXiv:2004.04906*.

Khan S, Naseer M, Hayat M, Zamir SW, Khan FS, and Shah M (2022). "Transformers in vision: A survey". In: *ACM computing surveys (CSUR)* 54.10s, pp. 1–41.

Kipf TN and Welling M (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

Kolomeets M, Desnitsky V, Kotenko I, and Chechulin A (2023). "Graph Visualization: Alternative Models Inspired by Bioinformatics". In: *Sensors* 23.7. DOI: 10.3390/s23073747.

Kornaropoulos EM and Tollis IG (2012). "DAGView: an approach for visualizing large graphs". In: *International Symposium on Graph Drawing*. Springer, pp. 499–510.

Krizhevsky A (2009). "Learning Multiple Layers of Features from Tiny Images". In: URL: https://api.semanticscholar.org/CorpusID:18268744.

Lenz M, Ollinger S, Sahitaj P, and Bergmann R (2019). "Semantic Textual Similarity Measures for Case-Based Retrieval of Argument Graphs". en. In: *Case-Based Reasoning Research and Development*. Vol. 11680. Series Title: Lecture Notes in Computer Science.

Cham: Springer International Publishing, pp. 219–234. DOI: 10.1007/978-3-030-29249-2˙15.

Lenz M, Sahitaj P, Kallenberg S, Coors C, Dumani L, Schenkel R, and Bergmann R (2020). *Towards an Argument Mining Pipeline Transforming Texts to Argument Graphs*. en. arXiv:2006.04562 [cs]. DOI: 10.3233/FAIA200510.

Lewiński M and Mohammed D (2016). "Argumentation Theory". en. In: *The International Encyclopedia of Communication Theory and Philosophy*. 1st ed. Wiley, pp. 1–15. DOI: 10.1002/9781118766804.wbiect198.

Li B, Zhou H, He J, Wang M, Yang Y, and Li L (2020). "On the sentence embeddings from pre-trained language models". In: *arXiv preprint arXiv:2011.05864*.

Li J, Li D, Savarese S, and Hoi S (2023). *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. arXiv: 2301.12597 [cs.CV].

Li J, Li D, Xiong C, and Hoi S (2022). *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. arXiv: 2201.12086 [cs.CV].

Lin T, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, Perona P, Ramanan D, Dollár P, and Zitnick CL (2014). "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312. arXiv: 1405.0312.

Lionakis P, Kritikakis G, and Tollis IG (2023). "Experiments and a User Study for Hierarchical Drawings of Graphs". In: *IEEE Access*.

Liu H, Li C, Wu Q, and Lee YJ (2023). *Visual Instruction Tuning*. arXiv: 2304.08485 [cs.CV].

Liu T, Moore A, Yang K, and Gray A (2004). "An investigation of practical approximate nearest neighbor algorithms". In: *Advances in neural information processing systems* 17.

Liu Z, Hu H, Lin Y, Yao Z, Xie Z, Wei Y, Ning J, Cao Y, Zhang Z, Dong L, Wei F, and Guo B (2021a). "Swin Transformer V2: Scaling Up Capacity and Resolution". In: *CoRR* abs/2111.09883. arXiv: 2111.09883.

Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, and Guo B (2021b). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *CoRR* abs/2103.14030. arXiv: 2103.14030.

Liu Z and Shao Y (2022). "Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder". In: *arXiv preprint arXiv:2205.12035*.

Ma J, Bai Y, Zhong B, Zhang W, Yao T, and Mei T (2023). "Visualizing and understanding patch interactions in vision transformer". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Malkov YA and Yashunin DA (2018). "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs". In: *IEEE transactions on pattern analysis and machine intelligence* 42.4, pp. 824–836.

Munzner T (2014). *Visualization analysis and design*. CRC press.

El-Nouby A, Neverova N, Laptev I, and Jégou H (2021). "Training vision transformers for image retrieval". In: *arXiv preprint arXiv:2102.05644*.

Nowell L, Schulman R, and Hix D (2002). "Graphical encoding for information visualization: an empirical study". In: *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.* IEEE, pp. 43–50.

Oquab M, Darcet T, Moutakanni T, Vo H, Szafraniec M, Khalidov V, Fernandez P, Haziza D, Massa F, El-Nouby A, Assran M, Ballas N, Galuba W, Howes R, Huang PY, Li SW, Misra I, Rabbat M, Sharma V, Synnaeve G, Xu H, Jegou H, Mairal J, Labatut P, Joulin A, and Bojanowski P (2024). *DINOv2: Learning Robust Visual Features without Supervision*. arXiv: 2304.07193 [cs.CV].

Peldszus A and Stede M (2015). "An annotated corpus of argumentative microtexts". In: *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation, Lisbon*. Vol. 2, pp. 801–815.

Penedo G, Malartic Q, Hesslow D, Cojocaru R, Cappelli A, Alobeidli H, Pannier B, Almazrouei E, and Launay J (2023). *The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only*. arXiv: 2306.01116 [cs.CL].

Perozzi B, Al-Rfou R, and Skiena S (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

Qu Y, Ding Y, Liu J, Liu K, Ren R, Zhao WX, Dong D, Wu H, and Wang H (2020). "RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering". In: *arXiv preprint arXiv:2010.08191*.

Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, and Sutskever I (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *CoRR* abs/2103.00020. arXiv: 2103.00020.

Radford A, Narasimhan K, Salimans T, Sutskever I, et al. (2018). "Improving language understanding by generative pre-training". In.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, and Liu PJ (2020). "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *The Journal of Machine Learning Research* 21.1, pp. 5485–5551.

Reda K, Salvi AA, Gray J, and Papka ME (2021). "Color nameability predicts inference accuracy in spatial visualizations". In: *Computer Graphics Forum*. Vol. 40. 3. Wiley Online Library, pp. 49–60.

Reed C (2006). "Preliminary results from an argument corpus". In: *Linguistics in the twenty-first century*, pp. 185–196.

Reimers N and Gurevych I (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084*.

Silva S, Santos BS, and Madeira J (2011). "Using color in visualization: A survey". In: *Computers & Graphics* 35.2, pp. 320–333.

Stab C and Gurevych I (2017). *Argument Annotated Essays (version 2)*. URL: https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2422.

Steck H, Ekanadham C, and Kallus N (2024). "Is Cosine-Similarity of Embeddings Really About Similarity?" In: *arXiv preprint arXiv:2403.05440*.

Su H, Kasai J, Wang Y, Hu Y, Ostendorf M, Yih Wt, Smith NA, Zettlemoyer L, Yu T, et al. (2022). "One embedder, any task: Instruction-finetuned text embeddings". In: *arXiv preprint arXiv:2212.09741*.

Sugiyama K, Tagawa S, and Toda M (1981). "Methods for visual understanding of hierarchical system structures". In: *IEEE Transactions on Systems, Man, and Cybernetics* 11.2, pp. 109–125.

Tang H, Ji D, Li C, and Zhou Q (2020). "Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification". In: *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 6578–6588.

Tang J, Liu J, Zhang M, and Mei Q (2016). "Visualizing large-scale and high-dimensional data". In: *Proceedings of the 25th international conference on world wide web*, pp. 287–297.

Thongtan T and Phienthrakul T (2019). "Sentiment classification using document embeddings trained with cosine similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pp. 407–414.

Tian Y, Chen X, and Ganguli S (2021). "Understanding self-supervised Learning Dynamics without Contrastive Pairs". In: *CoRR* abs/2102.06810. arXiv: 2102.06810.

Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, and Jégou H (2020). "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877. arXiv: 2012.12877.

Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, and Lample G (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Van Eemeren FH (2018). *Argumentation theory: A pragmadialectical perspective*. Springer.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, and Polosukhin I (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Visser J, Konat B, Duthie R, Koszowy M, Budzynska K, and Reed C (2020). "Argumentation in the 2016 US presidential elections: annotated corpora of television debates and social media reaction". In: *Language Resources and Evaluation* 54.1, pp. 123–154.

Walker MA, Tree JEF, Anand P, Abbott R, and King J (2012). "A Corpus for Research on Deliberation and Debate." In: *LREC*. Vol. 12. Istanbul, Turkey, pp. 812–817.

Walton D and Macagno F (2015). "A classification system for argumentation schemes". In: *Argument & Computation* 6.3, pp. 219–245.

Walton D, Reed C, and Macagno F (2008). *Argumentation schemes*. Cambridge University Press.

Wang J, Yang Z, Hu X, Li L, Lin K, Gan Z, Liu Z, Liu C, and Wang L (2022). *GIT: A Generative Image-to-text Transformer for Vision and Language*. arXiv: 2205.14100 [cs.CV].

Wang L, Yang N, Huang X, Jiao B, Yang L, Jiang D, Majumder R, and Wei F (2022a). "Simlm: Pre-training with representation bottleneck for dense passage retrieval". In: *arXiv preprint arXiv:2207.02578*.

Wang L, Yang N, Huang X, Jiao B, Yang L, Jiang D, Majumder R, and Wei F (2022b). "Text embeddings by weakly-supervised contrastive pre-training". In: *arXiv preprint arXiv:2212.03533*.

Wang W, Bao H, Dong L, Bjorck J, Peng Z, Liu Q, Aggarwal K, Mohammed OK, Singhal S, Som S, et al. (2022). "Image as a foreign language: Beit pretraining for all vision and vision-language tasks". In: *arXiv preprint arXiv:2208.10442*.

Wheeler R and Aitken S (2000). "Multiple algorithms for fraud detection". In: *Applications and Innovations in Intelligent Systems VII: Proceedings of ES99, the Nineteenth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge, December 1999*. Springer, pp. 219–231.

Wills GJ (1999). "NicheWorks—Interactive Visualization of Very Large Graphs". In: *Journal of Computational and Graphical Statistics* 8.2, pp. 190–212. DOI: 10.1080/10618600.1999.10474810.

Xiao S, Liu Z, Zhang P, and Muennighof N (2023). "C-pack: Packaged resources to advance general chinese embedding". In: *arXiv preprint arXiv:2309.07597*.

Xie Z, Zhang Z, Cao Y, Lin Y, Bao J, Yao Z, Dai Q, and Hu H (2022). "Simmim: A simple framework for masked image modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663.

Xu M (2021). "Understanding graph embedding methods and their applications". In: *SIAM Review* 63.4, pp. 825–853.

Yu L, Yazici VO, Liu X, Weijer Jvd, Cheng Y, and Ramisa A (2019). "Learning metrics from teachers: Compact networks for image embedding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2907–2916.

Zhou D, Yu Z, Xie E, Xiao C, Anandkumar A, Feng J, and Alvarez JM (2022). "Understanding the robustness in vision transformers". In: *International Conference on Machine Learning*. PMLR, pp. 27378–27394.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Trier, 4. April 2024