ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Scene-Graph Attention for Visual Grounding in Indoor Environments

Master's Thesis

Amalie Kjaer
Department of Computer Science

Advisors: Dr. Zuria Bauer, Dr. Mihai Dusmanu
Supervisor: Prof. Dr. Marc Pollefeys
Computer Vision and Geometry Group
Department of Computer Science D-INFK

August 30, 2024

Abstract

We address the challenge of Visual Grounding (VG) in 3D indoor environments, which involves identifying objects in a scene corresponding to a natural language question. Existing solutions often struggle with complex reasoning tasks that require a deep understanding of both visual and textual information. Many current approaches rely on language priors, rather than an encoded understanding of the scene structure, and few have explored the potential of scene-graphs for this task. To tackle these challenges, we propose a novel approach for VG that leverages the strengths of Scene-Graphs, Graph Neural Networks (GNN), and Contrastive Language-Image Pre-trained Models (CLIP). Specifically, 1) we developed a Scene-Graph dataset for ScanNet, which we call ScanSG, using CLIP to embed images of objects into graph nodes, and a simple heuristics-based method to generate graph edges. Our dataset can be paired with Question-Answer datasets about ScanNet, we refer to this dataset pairing as ScanSG+ScanQA. 2) We proposed an end-to-end VG model, AttSQ+GNN, which integrates graph and text modalities using a Graph Neural Network to encode contextual scene information and cross-attention mechanism to identify relevant objects within a scene based on a query. 3) We conducted a comprehensive evaluation of the model, its components, and the input data.

Our experiments demonstrate that while the proposed AttSQ model improves over the baseline on the full ScanSG+ScanQA dataset (43.9% against 28.1% instance accuracy). We also show that the addition of a cross-attention mechanism paired with learnable question-embeddings significantly improves perfomance across all test settings. However, the addition of a GNN-component to the model decreases model performance: the AttSQ+GNN model scored 16.4 percent-points lower than its no-GNN counterpart, AttSQ when evaluated on the full ScanSG+ScanQA dataset. We show that the model's performance is highly dependent on the quality of the scene graph node features, and on the difficulty of the queries. These findings highlight the need for further refinement of scene graph generation and embedding techniques to enhance model performance, and offer insights into the strenghts and limitations of GNN-methods for VG.

Contents

Chapter 1

Introduction

Background and Motivation Visual Understanding, the ability of a computer system to interpret and derive meaning from visual data, is a critical task in the field of computer vision, with numerous applications. The importance of visual understanding lies in its potential to automate and enhance tasks that traditionally require human perception and interpretation. For example, in robotics, visual understanding enables navigation and path planning by allowing robots to perceive, understand and reason about their environment. In augmented reality, visual understanding enables overlaying digital content onto the physical world, creating immersive experiences in gaming, education, and remote collaboration. In human-assistive technologies, such as assistive devices for the visually impaired, visual understanding can help interpret surroundings and provide real-time feedback to users.

More specifically within the field of visual understanding, the tasks of Visual Grounding (VG) and Visual Question Answering (VQA) are critical as they enable systems to reason about visual data. VG and VQA are closely related tasks at the intersection of vision and language. On one hand, the VG task [8] consists in identifying objects in a scene that best correspond to a given natural language query. For example, given a scene representation and the question "Where did I leave my keys?", a VG model should locate the object of interest and return the object instance, such as {`object_instance_id: 42, object_label: keys, position: (x, y, z)`}. On the other hand, the VQA task [2] goes a step further by requiring the model to generate a natural language answer to questions that can pertain not only to object identification but also to attributes, relationships, and functions within the scene. For example, given the same scene representation and question, a VQA model should return an accurate and cohesive natural language response, such as `"You may have left your keys on top of the white cabinet, at location (x, y, z)"`. The key difference between VG and VQA thus lies in the output produced by the model.

While these reasoning tasks are straightforward for humans, they present significant challenges for AI, requiring complex reasoning across multiple domains. VQA involves generating textual answers based on visual input, demanding a deep understanding of both the image and the question. To achieve this, VG is essential, as it allows models to accurately localize and interpret objects within a scene, rather than relying on superficial language patterns or spurious correlations. By focusing on VG, this thesis aims to build a foundation for VQA systems that truly understand scene structure, moving closer to human-like interpretation of visual data.

Problem Statement and Objectives Current approaches to VG have explored the use of Transformers, Large Language Models and Graph Neural Networks to solve these tasks. However, despite advancements in VG, existing models often struggle with complex reasoning tasks that require a deep understanding of both visual and textual information. Moreover, while Graph Neural Networks (GNNs), scene-graph structures and pre-trained models like Contrastive Language-Image Pretraining (CLIP) [33] have shown promise in

various tasks, there has been little exploration of their combined potential in tackling VG tasks. This thesis aims to fill this gap by proposing a novel approach that leverages the strengths of scene-graphs, GNNs and CLIP for visual grounding in indoor environments. We evaluate our models on a mixed VQA and VG dataset [4] rather than a classical VG dataset, as this better captures the complex, real-world challenges posed by difficult questions.

In summary, this thesis focuses on developing an end-to-end model for VG that combines the power of GNNs with the robustness of CLIP embeddings. Specifically, the contributions of this work include:

- **Dataset Generation**: Generation of a paired dataset of Scene-Graphs and Questions for 3D indoor environments, referred to as ScanSG+ScanQA. This dataset is the first of its kind, providing a valuable resource for future research in this area.
- **VG Model**: Introduction of a novel VG model, AttSQ+GNN, which integrates graph and text modalities to accurately identify relevant objects within a scene based on a query. The model leverages the structural understanding provided by scene-graphs using GNNs, and the semantic richness of CLIP embeddings.
- **Experimental Evaluation**: Comprehensive evaluation of the proposed model on the ScanSG+ScanQA dataset, including analysis of model performance, robustness, and the impact of various design choices.

Figure 1.1 illustrates the contributions of this thesis.

Structure of the Thesis The remainder of this thesis is structured as follows: Chapter 2 reviews the related work in 3D visual understanding, scene graphs, and deep learning on graphs, providing context for the proposed approach. Chapter 3 details the methods used for scene graph generation and visual grounding model, including the development of the ScanSG+ScanQA dataset and the architecture of the AttSQ+GNN model. Chapter 4 presents the experimental setup and results, offering insights into the effectiveness of the proposed model and dataset, showing areas of strength and weakness, useful to future work. Chapter 5 discusses the implications of the findings, their significance, and potential avenues for future research. Chapter 6 concludes the thesis, summarizing the contributions and reflecting on the broader impact of this work.

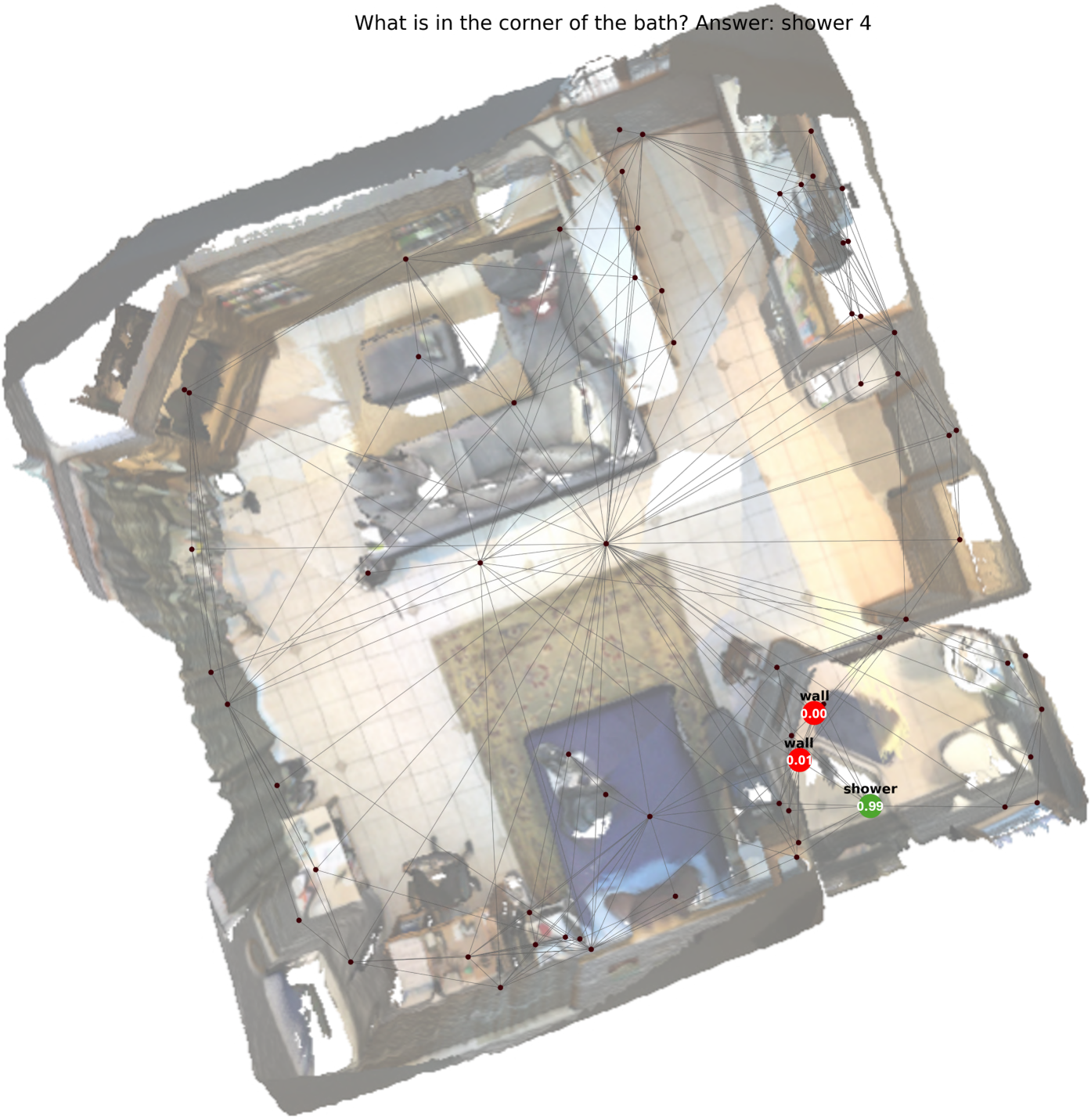


Figure 1.1: Output of the proposed VG model, overlaid on the model input. The scene-graph generated is shown overlaid its corresponding point-cloud. The text at the top of the figure shows an example input question. The large nodes show the model's top-3 predictions. The model correctly identified and located "shower" as the correct answer to the query (green node), with near-1 probability.

Chapter 2

Related Work

In the following sections, we present previous works in the 3D Visual Understanding field, and bring to light the relevant state-of-the art methods and results. Of particular interest are methods using 3D scene graphs or Graph Neural Networks to some avail to solve the VQA or VG tasks. We also briefly outline the "building blocks" of our proposed model, namely Graph Neural Networks, Image-Text Pretrained Models and Transformers. Lastly, we present the datasets relevant to this project, ScanNet and ScanQA, and provide some useful dataset statistics and insights.

2.1 3D Visual Understanding

The field of 3D Visual Understanding has gained significant attention in recent years, encompassing tasks such as instance segmentation, visual grounding (VG), and visual question answering (VQA). VG and VQA are closely related tasks at the intersection of vision and language. While the terms are sometimes used interchangeably in the literature [20], they differ in the output produced:

- The VG task [8] consists in identifying objects in a scene that best correspond to a given natural language query. As such, the input is an scene-question pair (the scene can have any representation, such as RBG-D, point-cloud, graph), and the output is an object instance present in the scene. The VG task requires a model to understand and align both visual and textual information, providing a foundation for tasks that require an understanding of the spatial and contextual relationship between objects.
- On the other hand, the VQA task [2] goes a step further by requiring the model to generate a natural language answer to questions that can pertain not only to object identification but also to attributes, relationships, and functions within the scene.

This difference means that VG and VQA datasets differ slightly: VQA datasets typically consist of question-answer pairs, where the answer is a natural language text [4]. On the other hand, VG datasets typically consist of query-instance pairs, where the query is a natural language text description, and the answer is a specific object instance in the scene [8]. We note that generally speaking, VQA datasets encompass a more complex, broader range of questions that pertain not only to objects but also to their attributes, relations, and functions [4].

Both the VG and VQA tasks have been applied to a broad range of fields, such as video question-answering [43], medical VQA [25] or chat-bots [10]. Other than the area of application, we further distinguish 2D from 3D applications of VG and VQA. 2D applications take 2D data as input (eg. an image or a video), and only requires 2D information to solve the query. 3D applications take 3D data as input (eg. a point-cloud or RGB-D) and requires 3D information to solve the queries about distances, sizes, paths, etc. While most research efforts have focused on the 2D task, recent strides have been made towards generalising

this task to 3D [40]. The 3D VG and VQA tasks requires even more robust structural understanding and reasoning capabilities that its 2D counterparts.

Previous works have highlighted the importance of VG for solving the VQA task [8], [31]. More specifically, VG facilitates the alignment of objects within a scene with their corresponding natural language descriptions, a prerequisite for accurately answering questions in the VQA task. VG is thus a foundational step allowing the model to develop a deeper understanding of spatial and contextual relationships, which are critical for generating coherent and contextually appropriate responses in VQA. We therefore focus on the VG task.

Traditionally, VG follows a either a single-stage or a two-stage pipeline. On one hand, two-stage methods involve detecting object proposals first (or using ground truth segmentation), and then matching them with the text description. These models output an object instance. They are highly dependent on the quality of the object detectors [34]. On the other hand, one-stage methods eliminate the need for object proposals by taking the entire 3D scene as input and embedding point-level features in an end-to-end manner. These methods directly regress the coordinates of the object bounding box rather than outputting an object instance. This method may overlook contextual information, often leading to relatively coarse predictions [28]. As such, one-stage methods try to quickly determine the object location, but often miss relational information. Two-stage methods, on the other hand, first identify potential objects throughout the scene, and then match the objects with the text description, allowing for better spatial understanding, at the cost of increased complexity and processing time. Focusing on the two-stage pipeline, it typically considers some (or all) of the following steps:

1. **Scene Feature Extraction**: The scene is encoded at the object-level, using the segmentation.
2. **Text Feature Extraction**: Typically using GloVe [32], RoBERTa [26] or more recently CLIP [33].
3. **Scene and Text Context Encoder**: Scene context encoders are used to capture dependencies within the 3D scene. These can be based on graph neural networks [11] or self-attention mechanisms [15], [42]. GNN-based encoders represent the relationships between object proposals explicitly, while self-attention mechanisms automatically learn semantic correlations without predefined structures. Text context encoders are typically based on recurrent neural networks [8] or self-attention mechanisms [15].
4. **Multi-Modal Interaction**: Early VG methods employed simple concatenation to combine scene and text features, followed by MLPs or GNNs to refine the fused representation [8], [1]. More recent methods use transformer-based cross-attention mechanisms to align and reason about the scene and text information [28], [17]. Some methods also integrate additional modalities, such as 2D images, multi-view, or human guidance [39], [5], [30].
5. **Grounding Head**: The grounding head ranks object proposals based on their matching score with the text, typically using a classifier implemented as an MLP [6], [44]. There is a growing trend of integrating LLMs into VG to improve semantic understanding and reasoning. These models can decompose complex queries, enrich context, and generate new descriptions to assist the grounding process. Some methods also use LLMs by leveraging their commonsense reasoning capabilities [13], [23].

To the best of our knowledge, no existing methods have combined both CLIP and GNNs to tackle the VG task.

The main VG dataset used to evaluate VG models are ScanRefer [8], Nr3D and Sr3D [1]. In Table 2.1, we compare some results from previous VG papers. We note that the accuracy of current methods range from 35.6% to 65.2% on the full Nr3D dataset, although in all methods, the accuracy varied per query type

Table 2.1: Overall accuracy of the VG task on the NR3D dataset for some previous papers. We note that he accuracy on this task ranges from 35.6% to 65.9%.

Method	Accuracy (%)
3DRP-Net [73]	**65.9**
MA2TransVG [90]	65.2
MiKASA [86]	64.4
3DRefTransformer [34]	39.0
TGNN [56]	37.3
Referit3D [12]	35.6

("Easy", "Hard", "View-dependent" or "View-independent") [1]. Currently, both two-stage and one-stage methods face challenges in accurately reasoning about complex spatial relationships in 3D scenes [18].

2.2 3D Scene Graphs

Formal Definition A 3D Scene Graph is a structured graph representation of a 3D environment that encodes the spatial, semantic, and relational information of objects within that environment [3]. In their simplest form, scene graphs extend instance segmentation by adding relational information between segment-nodes. More specifically, to construct a graph representation $G = (V, A, E)$, the following components must be defined:

1. **Node features,** $V \in \mathbb{R}^{n \times D_V}$**:** the graph contains n nodes of D-dimensional embedding, which encodes information about the object it represents. Nodes could encode information about the location, size, appearance, function, or any other attribute.

2. **Adjacency,** $A \in \mathbb{R}^{n \times n}$**:** the graph edges are defined in an $n \times n$ adjacency matrix, which encodes node connectivity. We note that node connectivity can be chosen to represent any type of similarity, such as proximity, visual similarity, semantic similarity or functional similarity.

3. **Edge features,** $E \in \mathbb{R}^{n \times n \times D_E}$**:** Optionally, D_E-dimensional edge features E can also be defined. Edge features could for example encode the relation between nodes, such as "next to", "inside", "above", "same color", or "same function".

Scene Graph Generation Chang et. al. [7] summarize existing scene graph generation methods. Of particular interest are CNN-based methods, which primarily focus on extracting features from images using CNNs and then classifying the relationships between objects based on these features [38]. These methods generally involve three key steps: region proposal, feature learning, and relationship classification. However, as Lorenz et. al [27] point out, the lack of formal definitions in Scene Graph Generation metrics hinders effective comparison of methods.

Use of Scene Graphs in VQA and VG Scene graphs have been adopted in VQA and VG with varying degrees of success. On one hand, some works have failed to utilize edge information within the graph, often reducing the graph to a simple collection of objects [12]. On the other hand, other works propose models for fully connected graphs [16], [22], once more not without leveraging the rich relational data encoded in the edges. Some methods do utilize edge information, but relies on additional data such as "instructions" or "reasoning patterns" on how to solve the query [24], which complicates its application across different domains.

Strengths and Limitations Scene graphs offer several advantages: they provide a semantically meaningful and interpretable high-level representation of a 3D space, making it easier to understand and analyze. They also incorporate contextual awareness, and 3-dimension information. Additionally, scene graphs are a highly compact representation, abstracting point clouds into a more memory-efficient format. This leads to a smaller memory footprint and faster inference times.

However, scene graphs also present some drawbacks. Compressing the 3D point cloud into a scene graph might result in information loss. Moreover, scene graph generation involves many design choices, and there is no such "ground truth" scene graph representation. This raises concerns about model robustness to scene graph representation variability.

2.3 Deep Learning on Graphs: Graph Neural Networks

Graph Neural Networks (GNNs) are a family of neural networks designed to operate on graph-structured data, where data points (nodes) are connected by edges representing relationships or interactions [14]. Unlike traditional neural networks that work on grid-structured data like images or tables, GNNs capture the dependencies and context of nodes within a graph, making them invaluable for tasks like node classification.

Input and Output: The input to a GNN is a graph $G = (V, E)$, with V and E being the set of nodes and edges respectively. Each node v in V is a feature vector. The output of a GNN is a graph with the same number of nodes and the same edges. However, the output node feature vectors have been updated to incorporate contextual information form its neighbouring nodes.

Forward Pass: This is achieved with a message-passing framework consisting of two main steps: the AGGREGATE function, followed by the COMBINE function. As such, A GNN layer consists of a AGGREGATE and COMBINE function and GNN layers are stacked to increase the receptive field of each node. More specifically, for each node v, the AGGREGATE function gathers information from its neighboring nodes. This is done using a permutation-invariant function (e.g., summation, mean, or max) that aggregates the embeddings of neighboring nodes $N(v)$. The COMBINE function then updates the node's embedding by combining its previous embedding $h_v^{(l-1)}$ with the aggregated features $a_v^{(l)}$. The most general formulation is:

$$a_v^{(l)} = \text{AGGREGATE}^{(l)}\left(\left\{h_u^{(l-1)} \mid u \in N(v)\right\}\right), h_v^{(l)} = \text{COMBINE}^{(l)}\left(h_v^{(l-1)}, a_v^{(l)}\right)$$

where $h_u^{(l-1)}$ is the embedding of neighbor u from the previous layer, $N(v)$ denotes the set of one-hop neighbors of node v. For the specific application of GCN,

$$a_v^{(l)} = \sum_{u \in N(v) \cup \{v\}} \frac{1}{\sqrt{d_u d_v}} h_u^{(l-1)},\ h_v^{(l)} = \sigma\left(W^{(l)} a_v^{(l)}\right), \tag{2.1}$$

d_u and d_v are the degrees of nodes u and v, respectively, $W^{(l)}$ is the learnable weight matrix at layer l, σ is a non-linear activation function such as ReLU.

Training: GNNs are trained end-to-end using gradient-based optimization methods, such as stochastic gradient descent. The loss function depends on the specific task; for example, cross-entropy loss is used for classification tasks.

The Oversmoothing Problem: The oversmoothing problem in GNNs arises when stacking too many layers. This causes node embeddings to converge to similar values, losing their distinctiveness. This occurs because

embeddings are repeatedly averaged over larger neighborhoods, leading to 'averaged out' features across the graph. This is problematic for tasks that rely on preserving node-specific information. To combat oversmoothing, strategies like limiting layers, using residual connections, or employing skip connections are often used. These techniques help maintain a balance between local and global feature aggregation in GNNs.

2.4 Image-Text Pretrained Models

Image-Text Pretrained Models are neural networks trained on large datasets containing both images and corresponding text descriptions. These models learn to map images and text into a shared embedding space where semantically related pairs (e.g., an image of a chair and the text "a chair") are positioned close to each other. This approach allows the model to perform tasks such as image captioning, image search, and zero-shot classification by understanding the relationship between visual and textual content.

Contrastive Language-Image Pretraining (CLIP): One of the most prominent models in this domain is CLIP [33]. It uses a contrastive learning approach to train two separate encoders: one for images and one for text. During training, the model maximizes the similarity between the image and text embeddings of matching pairs while minimizing the similarity for non-matching pairs.

Alternatives to CLIP: Several alternatives to CLIP have emerged, such as:

- **ALIGN (Contrastive Learning of Image and Text)** [21]: ALIGN uses a larger dataset and a slightly different training procedure to achieve high performance on various tasks.
- **LiT (Learning Image and Text Embeddings)** [37]: A variation that fine-tunes an image model on a frozen text model to improve zero-shot performance.
- **OWL-ViT** [29]: OWL-ViT extends CLIP by attaching a lightweight detection head to each Transformer output token. This allows the model to detect objects in images based on textual queries without needing pre-defined categories during training.

2.5 Transformers

Transformers [36] are a type of neural network architecture that has become the foundation for many state-of-the-art models in natural language processing and computer vision. The Transformer architecture consists of an encoder-decoder structure, as shown in Figure 2.1. We focus on the encoder, and on the cross-attention mechanism from the decoder.

Transformer Encoder The encoder processes the input sequence and generates a sequence of hidden representations. Each encoder contains two key components: self-attention, which allows each position in the sequence to attend to all other positions, capturing dependencies irrespective of their distance in the sequence; and a fully connected feed-forward network, applied to each position separately.

Cross-Attention Cross-attention is the main mechanism in the Transformer decoder. Unlike self-attention, where each position attends to other positions within the same sequence, cross-attention allows each position in the target sequence to attend to all positions in the source sequence. This mechanism enables the model to align and focus on relevant parts of the input sequence when generating each part of the output sequence. Isolating the cross-attention mechanism from the decoder can be done as in [41], and illustrated in Figure 2.2.

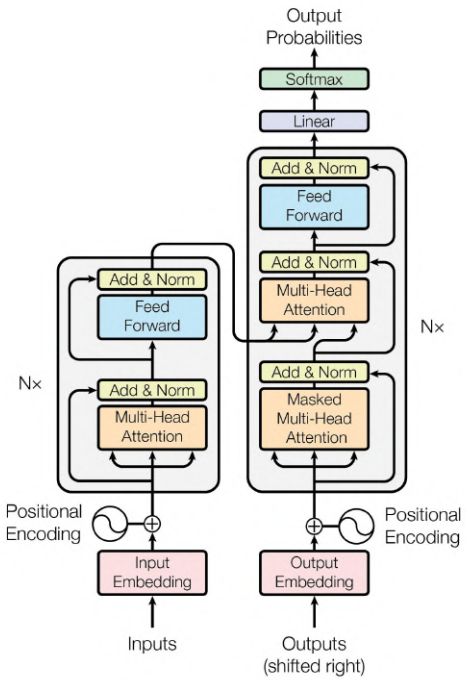


Figure 2.1: Transformer encoder and decoder architecute, as presented in [36].

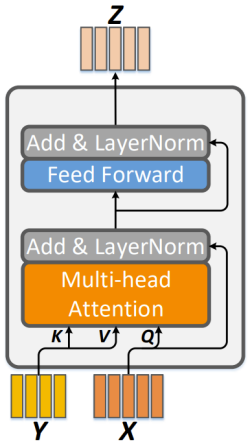


Figure 2.2: Cross-attention mechanism, as presented in [41].

2.6 Datasets

This section presents the two datasets relevant to this project, namely ScanNet and ScanQA.

2.6.1 ScanNet

The ScanNet dataset [9] consists of 1,513 segmented point clouds of indoor environments, generated from 2,492,518 segmented RGB-D frames. The dataset includes a range of spaces e.g. offices, apartments, bathrooms, libraries, and spans both small spaces and large spaces. The dataset offers both instance-level and semantic-level segmentations. In the following analysis, we only consider the set of ScanNet scenes for which the ScanQA dataset provides question-answer pairs.

We report some useful dataset statistics for the scenes for which ScanQA question-answer pairs exist. Figure 2.3 show the type of distribution of scene types. The three most common scene type are 'Bedroom', 'Living Room' and 'Bathroom'. Figure 2.4 shows the distribution of object labels. We note that the most common label is 'wall' (as expected, as all scenes contain multiple walls), followed by 'chair'. Figure 2.5 shows the inverse object density per scene. This describes how clustered or sparse the scene is. This is a useful metric for understanding how similar we should expect the graphs of ScanSG to be. We note that the object density is fairly consistent across scenes.

2.6.2 ScanQA

ScanQA [4] is the only question-answer dataset about 3D indoor scenes to date. We therefore focus on the ScanQA dataset, a VQA and VG dataset with challenging questions, to address the visual grounding (VG) task.

The ScanQA dataset consists of 30,238 question-answer pairs about 633 ScanNet scenes. Each question-answer pair corresponds to exactly one scene. While most questions have a unique answer, some questions have multiple possible answers. To simplify the task at hand, we only consider questions with a unique answer.

Questions Sorting out multi-answer entries, we are left with a simplified dataset of 21,324 questions about 632 scenes (on average 33.7 questions per scene). 17,321 of these questions are unique, while 4003 are repeated one or more times. Figure 2.6 shows the most common question types by wording. We note that the most common question phrasings are "What is...", "What color...", "What type..." and "Where is...". These phrasings represent more than half of the dataset. The average question contains 8.8 words.

Answers The ScanQA dataset provides two types of answers for each question: a long answer, and a short answer. The long answer is a free-form textual answer, while the short answer is simply a single object instance. The short answer can be interpreted as answering "where should one look to answer the question?". In other words, the short answer does not always directly answer the question, but rather specifies the most relevant object in the scene to the question. Table 2.2 provides a few examples of this. We also note that in 61% of questions, the answer label is directly contained in the question. For example, looking at Table 2.2, the question "What color table is on the left side of the cabinet?" already contains the short answer "table" within the question. In this project, we only consider the short answers, as this allows us to solve the easier and more interpretable task of classification rather than generation.

Table 2.2: Examples of long and short answers from the dataset.

Question	**Long Answer**	**Short Answer**
What is in the right corner of room by curtains?	brown cabinet with tv sitting in it	cabinet
What color table is on the left side of the cabinet?	light brown	table
Where is the beige wooden working table placed?	right of tall cabinet	cabinet

We note that the quality of question-answer pairs varies, based on the following observations:

- Some questions are vague, and do not specify which object instance is referred to. For example, "The trash can is to the left of what?" is unclear, as there are 3 trash cans in the scene and the question does not specify which trash can it is referring to.
- Some questions have multiple possible answers, but only specify a single answer. For example, the question "What is placed next to the fridge?" has multiple possible answers as many objects are next to the fridge in the scene. However, the answer reported is "door".

- The short answers provided do not follow a consistent logic: sometimes, the short answer is the subject of the question, while other times it is the object of the question. For example, ”Where is the coffee table kept?” is answered by ”coffee table” while ”What is the coffee table in front of” is answered by ”couch” rather than ”coffee table”.
- Some questions are poorly phrased, for example ”What is the white wooden door?” or have typos, such as ”Where is the rectangular keyboard piano lcoared?”.
- Some questions are view dependent, such as ”What color table is on the left side of the cabinet?”.

These observations imply that any model answering the question with a valid but different answer will be penalized as much as if it predicted a completely incorrect answer.

Furthermore, we note that many of the questions do not require much (if any) 3D information to answer. In other words, they could easily be answered only from 2D RGB images. This is likely because the images were generated from the 2D images of the scenes. The dataset contains no questions about distances, paths, dimensions or 3D geometry.

However, the questions in the dataset are tailored specifically to the ScanNet scenes, and cannot be answered without knowledge of the relevant scene. The dataset does not contain generic questions such as ”Where can I watch a film?” but rather scene-specific questions such as ”What color towel is on the left of a white towel?”.

Figure 2.7 shows the answer distribution in the simplified dataset. We note that the answer distribution is non-uniform, and questions are most commonly answered by the label ”chair”.

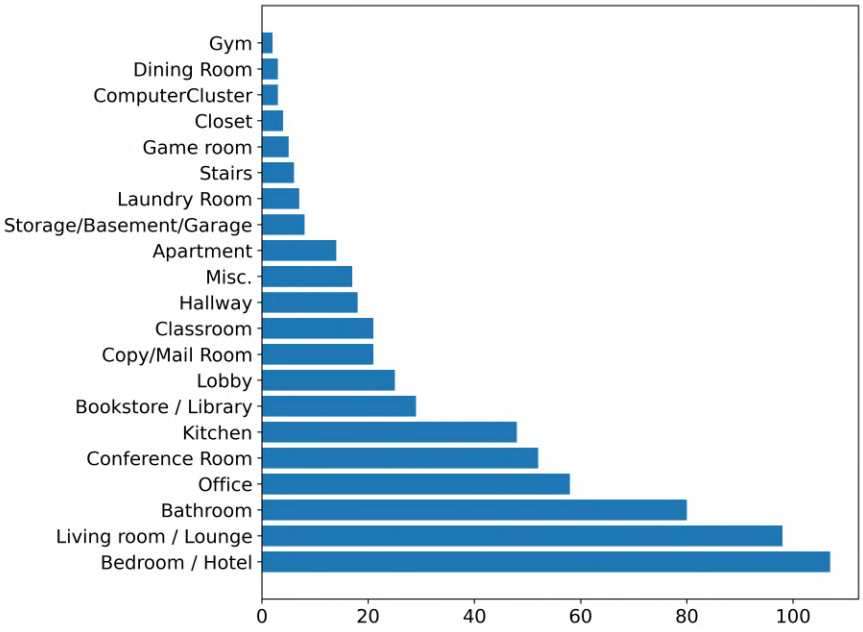


Figure 2.3: Scene type distribution in ScanNet.

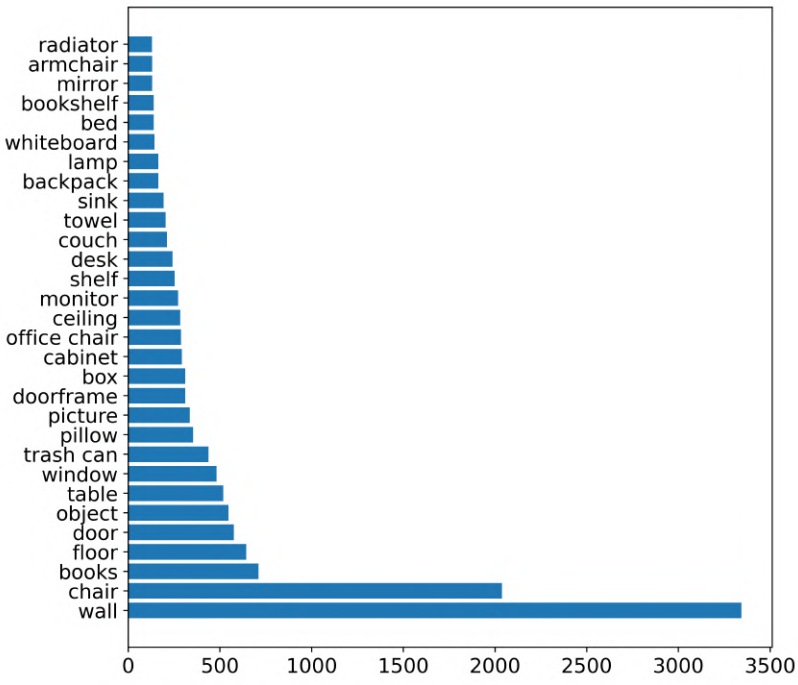


Figure 2.4: Object label distribution for the 30 most common objects in ScanNet.

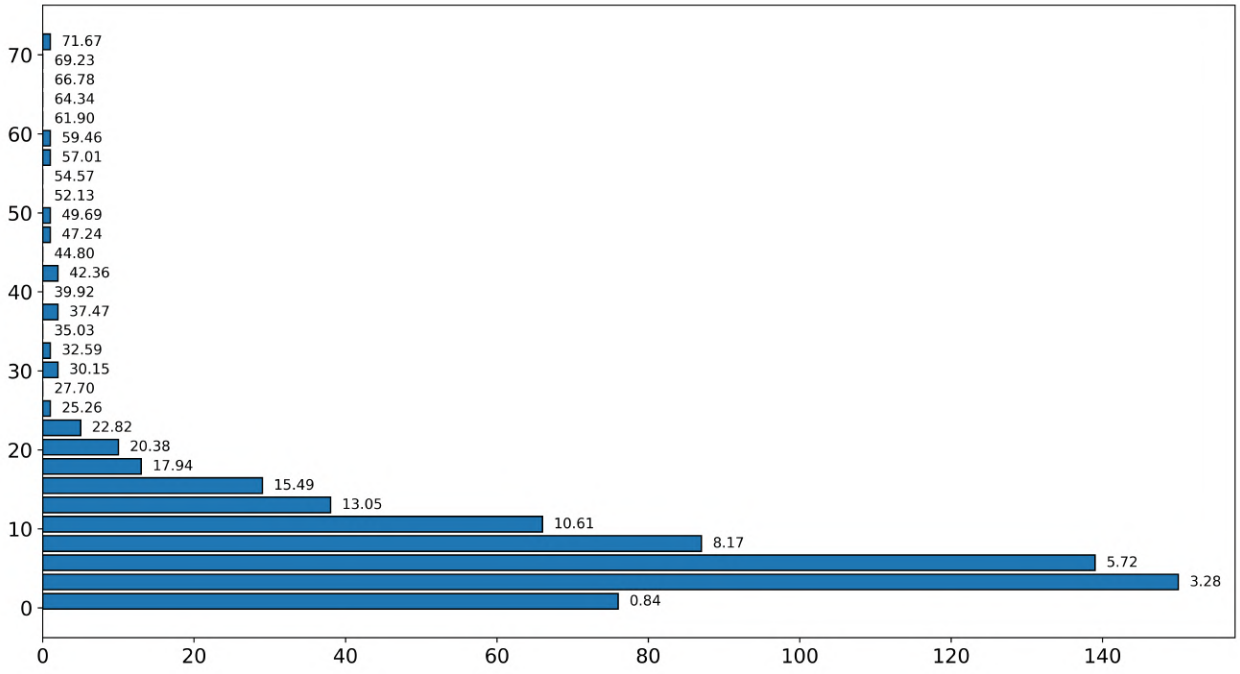


Figure 2.5: Scene (inverse) density distribution (m^2 per object) in ScanNet.

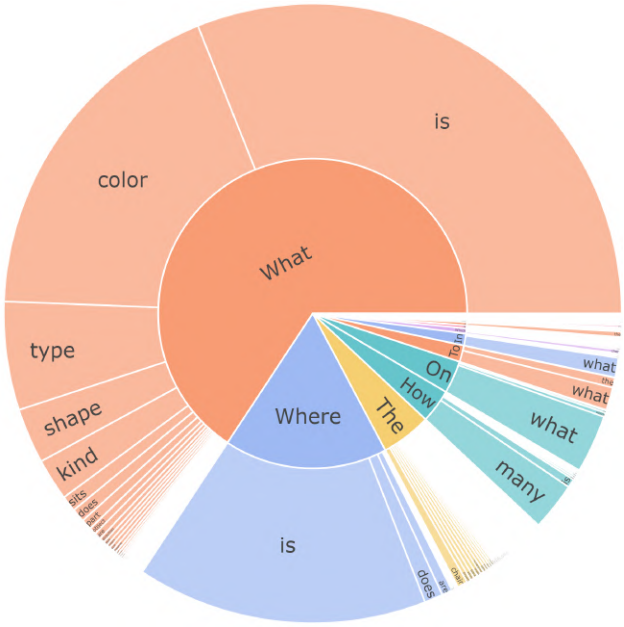


Figure 2.6: Most common question types in ScanQA by wording.

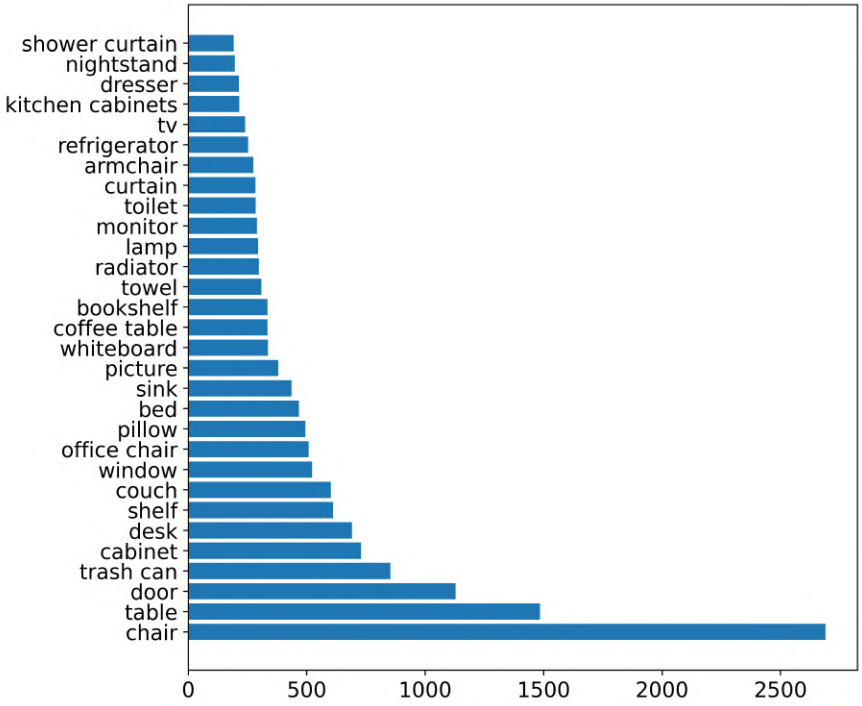


Figure 2.7: Answer distribution of 30 most common answers in the ScanQA dataset.

Chapter 3

Methods

The two main challenges of the VG task are 1) to create rich scene and query representations, often starting from RGB-D frames and free-form natural language text, and 2) to combine scene and text queries in a multi-modal model, in order to obtain an object prediction given question-scene input pairs. To address these challenges, we propose to develop:

1. A dataset of Scene-Graphs for the ScanNet dataset (ScanSG), which can be paired with the questions from the ScanQA dataset. This is, to our knowledge, the first dataset of paired Scene-Graphs and Questions for 3D indoor environments. We refer to this paired dataset as ScanSG+ScanQA.
2. An end-to-end 3D-VG model (AttSQ+GNN), which combines graph and text modalities to identify the most relevant object in the scene.

The following sections detail the Scene Graph Generation (SGG) method used to obtain the ScanSG dataset, as well as the 3D-VG classification models explored. Figure 3.1 provides a brief overview of methods used, which we describe in detail below.

3.1 Scene Graph Generation

We aim to produce semantically rich graph-based representations for indoor scenes captured in the ScanNet dataset. These representations will transform the raw, complex point-cloud data into a more memory-efficient and interpretable scene graph format. The scene graphs should retain critical 3D structural details, which are essential for distinguishing between different scene layouts, even when they contain the same objects arranged differently. By preserving this 3D spatial information, we hope that our approach will facilitate more advanced spatial reasoning and understanding of the relationships between objects within the environment.

To create this scene graph representation, we define the graph nodes as segmented objects within the scene, while the edges between these nodes indicate spatial proximit. As such, nodes representing objects close to each other are connected by an edge. It is important to note that there is no predefined ”ground truth” for such a representation, as SGG involves making design decisions and assumptions.

The following sections detail the simple heuristics-based SGG method used in this paper to generate ScanSG. We first describe the node feature generation (Section 3.1.2), and then the adjacency generation (Section 3.1.3). Together, these components form the ScanSG dataset.

3.1.1 Preprocessing

The only preprocessing step is to remove scenes with faulty segmentations. Specifically, five scenes in ScanNet present segmentation artefacts (some segmented objects do not appear in any frame).

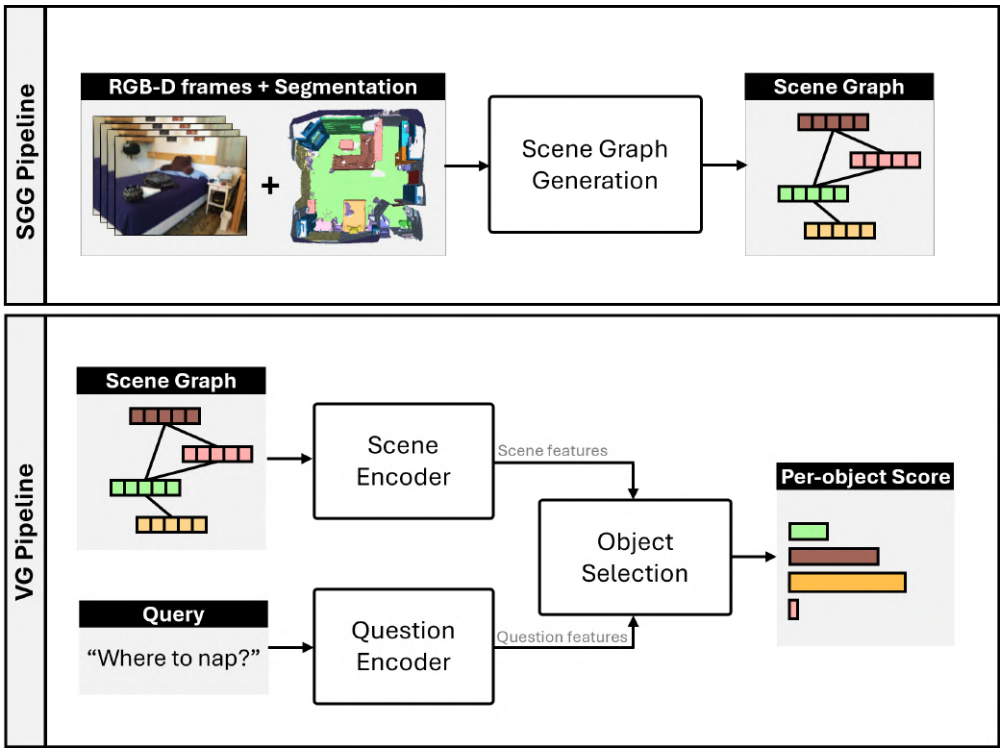


Figure 3.1: Overview of the Scene Graph Generation (SGG) and Visual Grounding (VG) methodologies. The SGG takes as input segmented RGB-D frames of a scene, and outputs a Scene-Graph representation of the scene. The heuristics-based SGG method is detailed in Section 3.1. The VG takes as input a Scene-Graph - Query pair, and outputs a relevance score for each object in the scene. This is achieved by combining encoded scene and question features in an Object Selection Head. Each component of the VG pipeline is further detailed in Section 3.2.

3.1.2 Node Features

While existing SGG methods segment the scene as a first step, we use the ground truth segmentation provided by ScanNet to get a more accurate SG dataset. This approach aligns with findings from previous research, which emphasize the importance of high-quality segmentation in VG model performance. By using oracle segmentation, we aim to minimize errors introduced during the segmentation, allowing our model to focus more effectively on understanding spatial relationships within the scene.

We initialize the node embeddings using CLIP-embedded images of the the objects. CLIP offers a robust starting point for semantically rich node embeddings, as it aligns images and text within the same embedding space. This allows for easy comparison between visual and textual data. Furthermore, CLIP captures both the appearance of objects (including attributes like color and shape), and their semantic meaning (including their function). This is valuable because it ensures that the embedding of a textual description will be closely aligned with the embedded corresponding image.

However, since each object may be captured across multiple RGB-D frames from different viewpoints, we face the challenge of determining which views to embed (a process we refer to as **View Selection**) and how to integrate these various views into a single node embedding (refer to **Cropping + Embedding**).

View Selection

Given the high frame rate of the image capture, we discard every other RGB-D image in each scene to speed up the data processing. We wish to filter out poor object views, and to only consider the top-k views

of each object. In ScanSG, $k = 10$ (the effect of different k-values on embedding quality are compared in Section 4.1.1). The top-k views for each object are selected by ranking views based on their view score s_f, which we define as:

$$s_f = \frac{n_f}{w \times h} \times \frac{b_f}{8},$$

where n_f is the number of pixels occupied by the object in the frame f; w and h are the width and height of the frame and b_f is the number of bounding box corners visible in the frame (out of 8 possible corners). b_f is obtained by projecting the 3D-coordinates of a tight, axis-aligned object bounding box (from the ScanNet oracle point-cloud segmentation) into each RGB-frame f. $b_f = 8$ implies that the entire object is visible in the frame, while $b_f < 8$ implies a partial view of the object.

We refer to the first term of s_f as the *pixel score*, since it represents how much space the object takes up in the frame (close-up views are preferred). The second term is the *corner score*, which represents how much of the total object is visible in the frame (full views are preferred to partial views). A view score $s_f \approx 1$ implies a close-up view of a fully visible object, whereas $s_f \approx 0$ implies either a distant view, or a partial view, or both.

Cropping For each object, we then crop its top-k views around the object of interest. In the ScanSG dataset, a square crop centred around the object, and resized to 224×224 is used. This is because CLIP was trained on this input size. The effect on embedding quality of different cropping methods are compared in Section 4.1.1.

Embedding For each object, the top-k crops are then CLIP-embedded (using the HuggingFace pre-trained model `clip-vit-base-patch32`) and aggregated using the s_f-weighted average. This yields a single 512-dimensional CLIP-embedding for each object. This aggregated embedding captures view variability, giving the object representation a richer embedding.

3.1.3 Adjacency

The adjacency matrix is the graph component encoding 3D information. Edges in the ScanSG dataset represent spatial proximity of nodes. In other words, an edge should connect two nodes if the distance between them is less than some predetermined heuristic threshold. This thresholding method is justified because the object density is mostly consistent across scences, as shown in Figure 2.5. Note that for datasets with more varied object density, using a single threshold value for all scenes may not work well.

To define the adjacency matrix, the shortest distance between all objects in a scene should therefore be calculated. The naive approach to this problem is to compute the distance between all 3D points of 2 object point-clouds, and to retain the shortest distance. Although this is most accurate method to calculate the shortest distance between two objecs, it is too computationally expensive, and also sensitive to outliers.

We therefore approximate the shortest distance between two objects by the shortest distance between their axis-aligned 3D bounding boxes, A and B. This can be calculated as:

$$d = \sqrt{\sum_{i=1}^{n} \max(0, A_{\text{min},i} - B_{\text{max},i})^2 + \sum_{i=1}^{n} \max(0, B_{\text{min},i} - A_{\text{max},i})^2},$$

where A_{min} and B_{min} are the vectors of minimum coordinates, A_{max} and B_{max} are the vectors of maximum coordinates, and i indexes the dimensions of the space. The max function ensures that any negative values (indicating overlap) are clamped to zero. We sum the two terms because only one will be non-zero, depending on the relative positioning of the bounding boxes.

A threshold of 0.2 was selected heuristically (see Section 4.1.2). The adjacency matrix was symmetrised to ensure bidirectional relationships between nodes. Additionally, self-loops were added.

We pair ScanSG with ScanQA and refer to this dataset of Scene-graph - Question pairs as ScanSG+ScanQA.

3.2 Visual Grounding Models

Figure 3.1 illustrates the general 3D-VG method. The models take as inputs a question and a scene graph (generated as described above), and outputs a distribution of the most relevant objects in the scene relative to the question. In other words, the model outputs "Where one should look to answer the question". Regardless of architecture, all the proposed VG model must have the following components:

1. A **scene encoder**, which takes as input a scene graph and outputs embeddings for each object in the scene,
2. A **question encoder**, which takes as input a query and outputs embeddings for the query,
3. An **object selection head**, which combines scene and question encodings to predict the most relevant object.

We explore different solutions for each of these components, and compare the performance of the following three distinct model architectures:

- **Baseline** model (no learning),
- **Baseline + GNN** model (learning only for scene encoding),
- **AttSQ-Net**: Attention-based Scene-Graph Query model (learning for question encoding and selection head),
- **AttSQ-Net + GNN** (learning for scene encoding, question encoding and selection head).

We incrementally build up the AttSQ-Net + GNN model, starting from the simple baseline, and adding complexity to each new model, as shown in Figure 3.2. This helps us identify which components of the final proposed model AttSQ-Net+GNN, are actually beneficial. The following sections detail the architecture of each model.

3.2.1 Baseline

The baseline model is used to establish a lower bound on the performance of the VG task, and is illustrated in Figure 3.3. The baseline does not require any learning, and relies solely on the pre-computed CLIP features of the objects in the scene and the question. More specifically, the model consists of the following components:

1. **Scene Encoder:** the scenes are encoded as the ScanSG nodes (without edges). As a reminder, in ScanSG, the node representing an object contains the score-weighted average CLIP-embedding of its top-k image crops in the RGB-D dataset (as described in Section 3.1.2). As such, the embedding dimensions at the scene encoder output are `[number of nodes in batch, embedding dimension]`.
2. **Question Encoder:** the query is first tokenized. Then, the question encoding is the sentence-level CLIP feature of the tokenized question. In CLIP, the hidden state of the end-of-sentence `[EOS]` token from the last layer of the transformer is used as the representation of the entire sentence. This embedding is then layer-normalized and linearly projected into the multi-modal embedding space

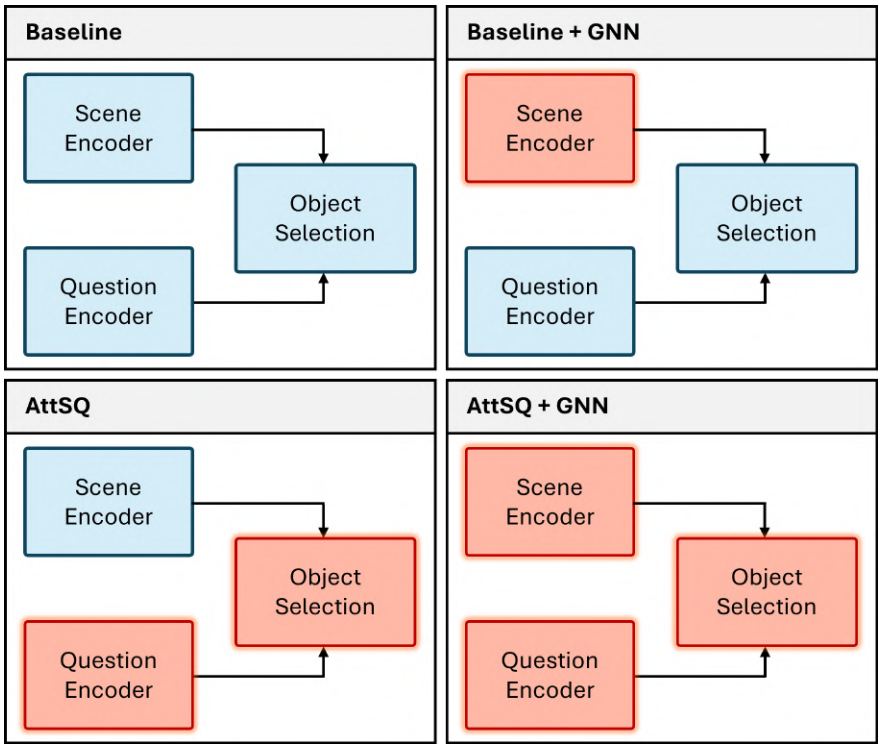


Figure 3.2: Overview of the proposed models. All models consist of a scene encoder, a question encoder, and an object selection head. For each model, we show which components are trainable (red) and which components are fixed (blue). In the baseline model, all components are pre-computed. In the Baseline+GNN model, only the scene encoder is trained. In the AttSQ model, the question encoder and object selection head are trained. In the AttAQ+GNN model, all model components are trained. The specifics of each model architecture are further described in the subsequent sections.

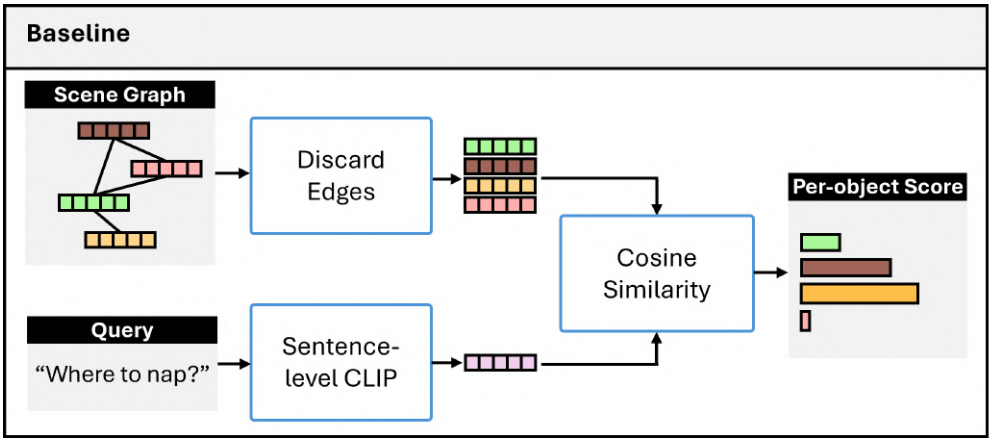


Figure 3.3: Summary of the Baseline model architecture. The model uses the pre-trained CLIP model to embed images of objects in the scene (stored in scene nodes) and the query, and compare these with cosine similarity to retrieve the most similar object to the query.

shared with the image encoder. We use the pretrained `clip-vit-base-patch32` model. As such, the embedding dimensions at the question encoder's output are `[batch size, embedding`

`dimension]`.

3. **Object Selection Head:** the object selection head computes the cosine similarity between the 512-dimensional question and node features. The node with the highest similarity to the query n^* is selected as the output object:

$$n^* = \arg\max_{n \in V} \hat{\mathbf{q}} \cdot \hat{\mathbf{n}},$$

where V is the set of all nodes in the scene; $\hat{\mathbf{q}}$ is the normalized 512-dimensional embedding vector of the query (i.e., the sentence-level CLIP feature of the question); $\hat{\mathbf{n}}$ is the normalized 512-dimensional embedding vector of a node n in the scene; and $\hat{\mathbf{q}} \cdot \hat{\mathbf{n}}$ is the cosine similarity between the query embedding $\hat{\mathbf{q}}$ and the node embedding $\hat{\mathbf{n}}$. To obtain a probability distribution over the nodes, a softmax function is applied to these similarity scores, ensuring that the output reflects the most relevant object in relation to the query. To obtain a probability distribution over the nodes, a softmax function is applied to these similarity scores, ensuring that the output reflects the most relevant object in relation to the query. The model output dimensions is `[batch size, 1]`.

In summary, the baseline model uses the pre-trained CLIP model to embed images of objects in the scene and the query, and compare these to retrieve the most similar object to the query. This is expected to work well for descriptive queries, such as "I'm looking for a large, soft, yellow couch" as CLIP was trained on descriptive captions. However, this is not expected to work well on queries which do not directly describe the sought object, such as "Select any object *but* the large, soft, yellow couch".

While this baseline model offers a simple and commonly used approach to the visual grounding (VG) task [35], [12], it has notable limitations. Firstly, the scene encoder incurs significant information loss by discarding edges, thereby omitting structural information about the scene. As a result, the model fails to capture the overall room layout, and the nodes lack contextual awareness. Consequently, the scene encoding remains unchanged even if objects are rearranged, and objects with similar appearances may have highly similar features, regardless of their context or neighboring objects. This suggests that, while the model may achieve high semantic accuracy, its instance accuracy is likely to be low. Secondly, the model lacks any learning mechanisms, making it highly generalizable but unable to learn or adapt from the training data.

3.2.2 Baseline+GNN

To address the limitations of the baseline model, we explore the Baseline+GNN model, which uses a Graph Neural Network to learn context-aware node embeddings. The model architecture is illustrated in Figure 3.4. The model consists of the following components:

1. **Scene Encoder:** The ScanSG graphs are passed through a dimension-preserving GCN. The GCN updates node embeddings iteratively by aggregating information from neighbouring nodes, such that each node becomes a learned, pooled representation of itself and its 1-hop neighbourhood. This ensures that each node gains context awareness. More specifically, the scene encoder consists of a single 512×512 GCN layer, stripped of its activation function. Note that we omit the activation function as this preserves the original embeddings' scale and distribution, which is crucial for maintaining the integrity of features when using metrics like cosine similarity. However, the trade-off is that the model's non-linearity is removed, which might limit its ability to capture more complex relationships in the data. In Equation 2.1, we therefore set the activation function $\sigma = I$.

2. **Question Encoder:** Same in the Baseline model.

3. **Object Selection Head:** Same as in the Baseline model.

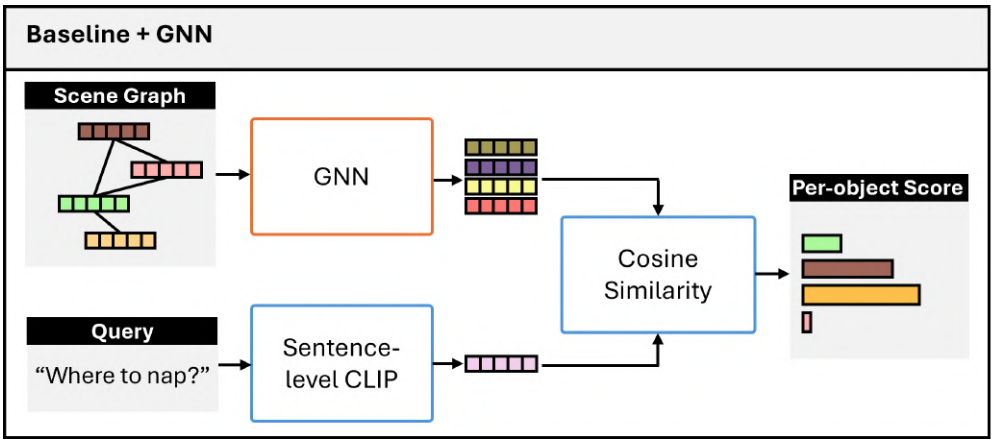


Figure 3.4: Summary of the Baseline+GNN model. CLIP-embedded images of objects are stored in the graph nodes. These node embeddings are then passed through a GCN to propagate information, allowing each node to become context-aware of its neighboring nodes. The query is encoded using sentence-level CLIP embeddings. The model computes the cosine similarity between the query embedding and the context-enriched node embeddings to identify the object most relevant to the query, producing per-object relevance scores.

We note that unlike the Baseline model, the Baseline + GNN model can align node features to query features in a *learned* way. The model can therefore be trained to better adapt to the dataset. However, the Baseline+GNN model still has some limitations. Using the cosine similarity ultimately requires the query and node features to be aligned. However, the query embeddings are fixed, meaning that the model may struggle with questions that are not well-aligned with the learned node embeddings. An alternative method would be to use node and query features to inform a separate *relevance score* for each node. We further explore this idea in the following sections.

3.2.3 AttSQ

We introduce AttSQ, an attention-based model designed to enhance both the question encoding and object selection processes, as shown in Figure 3.5. Unlike the Baseline model, which relies on aligning fixed question embeddings with learned scene embeddings via cosine similarity, AttSQ employs a cross-attention mechanism to infuse the node embeddings with query-specific information. The resulting 512-dimensional question-aware node embeddings are then passed through an MLP to generate a *relevance score* for each object in the scene, reflecting its relevance to the query. More specifically, the model consists of the following components:

1. **Scene Encoder:** Same as in the in Baseline model.
2. **Question Encoder:** Questions are encoded using word-level CLIP embeddings, producing embeddings of shape `[batch size, max sequence length, embedding dimension]`. These embeddings, which include padding tokens to standardize sequence length, are further processed by a transformer encoder, with the same architecture as illustrated in Figure 2.1. This allows the model to refine the question embeddings through training, adapting them specifically to the dataset.
3. **Object Selection Head:** The object selection head consists of a Multi-Layer Cross-Attention block, followed by a two-layer downsizing MLP. Each Cross-Attention block includes the following: a Multi-Head Attention module, concatenation with Layer Normalization, Dropout, a 2-layer feed-forward MLP, and another concatenation with Layer Normalization, followed by Dropout. This ar-

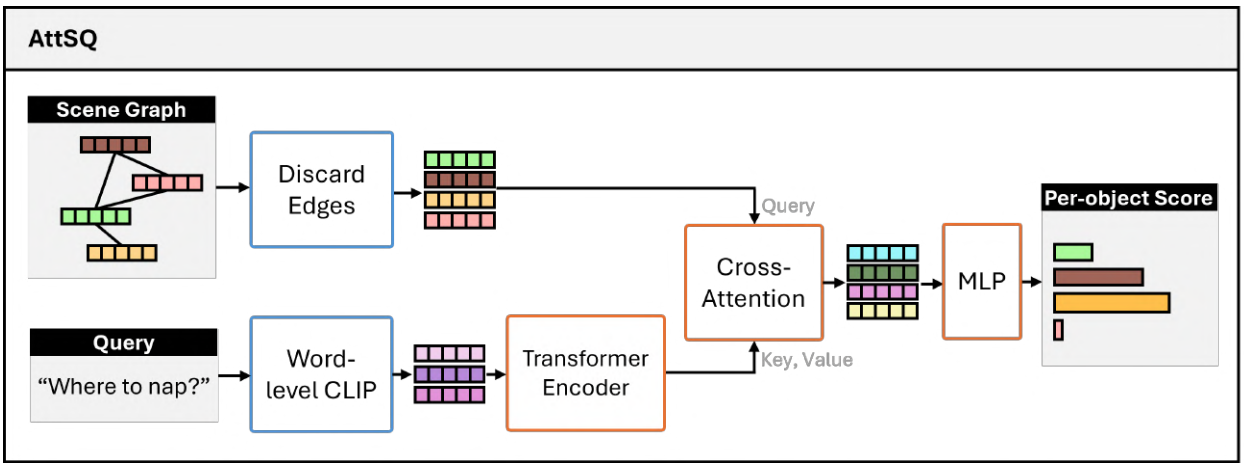


Figure 3.5: Summary of the AttSQ model. CLIP-embedded images of objects are stored in the graph nodes. The scene graph edges are discarded, resulting in raw node embeddings (with no additional context-awareness). The query is encoded using word-level CLIP embeddings, followed by further refinement through a transformer encoder. The resulting node features and question embeddings are then passed into a cross-attention mechanism, where each node embedding (query) attends to the question features (key-value pairs), incorporating query-specific information into the node representations. Finally, these enriched node embeddings are passed through a multi-layer perceptron (MLP) to produce a 1-dimensional relevance score for each object in the scene, reflecting its alignment with the query.

chitecture is inspired by the transformer model outlined in [41], and illustrated in Figure 2.2. Multiple cross-attention blocks are stacked to enhance the model's capacity. In this setup, the cross-attention query comprises the node features, while the key-value pairs are the question features. This design ensures that each node attends to all words in the question, excluding interactions with other nodes. As such, each node embedding should contain some question-awareness (but no context awareness). A key padding mask is applied to prevent nodes from attending to padding tokens, ensuring that only meaningful parts of the question influence the node embeddings. The output shape at this stage is `[number of nodes in batch, embedding dimension]`. To convert this into a 1-dimensional *relevance score*, the cross-attended node features are passed through a two-layer MLP. Note that we also collect the attention weights (averaged across heads) for further attention-pattern analysis.

Overall, the AttSQ model's node features lack context awareness due to the absence of edge information, meaning the model does not directly learn the structural relationships within the scene from the scene encoding. Instead, it relies on language priors or patterns derived from previous questions. Despite this limitation, the cross-attention mechanism facilitates the calculation of a relevance score, adding non-linearity to the object selection process, unlike the direct cosine similarity-based approach of the Baseline model. This design represents an intermediate step towards more sophisticated models, such as AttSQ+GNN, enabling a more nuanced evaluation of feature significance across different model architectures.

3.2.4 AttSQ+GNN

The AttSQ-Net + GNN combines all elements previously discussed, aiming to leverage their combined strengths, as illustrated in Figure 3.6.

1. **Scene Encoder:** Same as in Baseline+GNN.

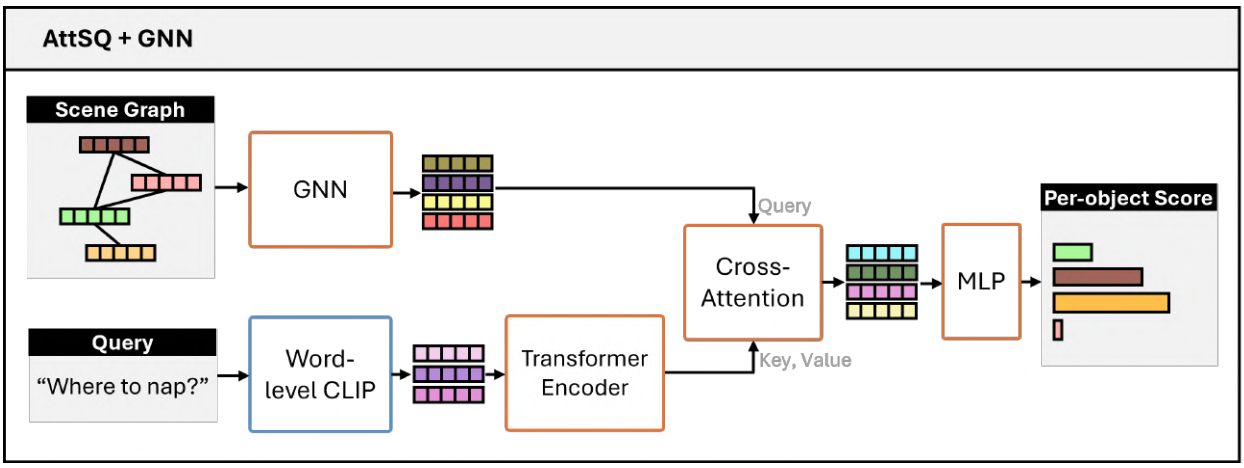


Figure 3.6: Summary of the AttSQ+GNN model. The figure illustrates the AttSQ+GNN model, which integrates components from both the AttSQ and Baseline+GNN models. The scene graph is processed through a GCN to generate context-aware node embeddings. The query is encoded using word-level CLIP embeddings refined by a transformer encoder. These context-aware node embeddings are then enhanced with query-specific information via a cross-attention mechanism. Finally, the enriched node features are passed through a MLP to produce per-object relevance scores, indicating the alignment of each object with the query.

2. **Question Encoder:** Same as in AttSQ.
3. **Object Selection Head:** Same as in AttSQ.

The AttSQ+GNN model integrates the context-awareness provided by the GCN with the query-specific focus introduced by the cross-attention mechanism, leading to more nuanced and informed relevance scoring. However, the model's complexity results in a larger number of parameters. This increased model size may lead to a higher risk of overfitting, particularly if not adequately regularized or trained on sufficiently large datasets.

Specific implementation details (such as number of layers, feed-forward dimensions, etc.) for each model are detailed in Section 4.

3.2.5 Graph Batching and Masking

In PyTorch Geometric, when handling multiple graphs in a batch, the individual graphs are effectively merged into a single large graph. The node feature matrices from each graph are stacked along the first dimension. The adjacency matrices of each graph are combined into a large block-diagonal matrix. Each block along the diagonal represents the adjacency matrix of one graph, while the off-diagonal blocks are filled with zeros, ensuring no interconnections between the nodes of different graphs in the batch. This batching strategy allows the operations designed for a single graph, such as graph convolution, to be efficiently applied to all graphs in the batch simultaneously.

However, this batching method removes the batch dimension, which is necessary for aligning graphs with batched questions. To retain the batch dimension, we use PyTorch Geometric's `to_dense_batch`, which converts graphs into dense tensors while preserving batch dimension. We mask fake nodes, introduced during padding, by setting their logits to `-inf` to ensure they do not impact the model's outputs.

3.2.6 Loss Function

All models are trained using the cross-entropy loss, which is commonly employed for multiclass classification tasks. The cross-entropy loss for a batch of size N and C classes is defined as:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}), \tag{3.1}$$

where $y_{i,c}$ (0 or 1) indicates whether class c is the correct class for the i-th sample. The term $p_{i,c}$ denotes the predicted probability that the i-th sample belongs to class c, as output by the model. The loss function sums over all samples and classes. It is minimized during training to improve accuracy.

Chapter 4

Experiments

In this section, we detail experimental methods and results for the ScanSG dataset generation (Section 4.1) and for the proposed 3D-VG models (Section 4.2).

4.1 Scene Graph Generation

In order to train a successful 3D-VG model, we must ensure that its input data, namely scene graphs and questions, are of high quality. This section empirically evaluates the impact of different design choices on the quality of the generated scene graphs. We evaluate the design choices for scene graph nodes and edges separately, and choose the best performing parameters for the proposed scene graph dataset. For scene graph nodes, we evaluate the effects of different cropping methods, k-values, and visibility scores on the quality of the node embeddings. For scene graph edges, we evaluate the impact of different threshold values on the quality of the edges in the scene graphs.

All experiments in this section are performed on a smaller subset of the dataset, consisting of 71 scenes, with a total of 2467 objects across scenes. This subset is also used for evaluating model performance in Section 4.2.

4.1.1 Node Features

We evaluate the impact of different cropping methods, k-values and visibility scores on the quality of the generated node CLIP-embeddings.

Cropping method: We compare the impact of the following cropping methods on node embedding quality (illustrated in Figure 4.1):

A. A tight, rectangular crop around the object with no further changes. With this method, the object is centered in the image. However, the background is not removed, meaning other surrounding objects could contribute to the embedding, and the image is not square. The rectangular crops are resized as in the original CLIP paper: the crop is resized to 224 in its minimum dimension, and then randomly cropped to a 224×224 square.

B. A tight, rectangular crop around the object with all background pixels masked out. With this method, the object is centered in the image, and the background is removed. However, CLIP was trained on square images with no background removal. To adapt rectangular crops to CLIP, we use the resizing method described in the original CLIP paper.

C. A square crop centred around the object, resized to 224×224. With this method, the crop is square and correctly sized, and no further processing is needed. However, the background is not removed and the non-tight crop means other surrounding objects might contribute to the node embedding.

D. A tight, rectangular crop, resized to 224×224. With this method, the crop is tight around the object (fewer surrounding objects included in the crop). However, the resized crops distort the image, and CLIP was not trained on distorted images.

Figure 4.1: Comparison of cropping methods for object-centric node embeddings. From left to right: (A) Tight rectangular crop with background masked out, (B) Tight rectangular crop with background included, (C) Square crop centered on the object, (D) Tight rectangular crop resized to a square. Each method impacts the quality of CLIP-generated embeddings by varying the inclusion of background elements and image distortion.

k-value: We also evaluate the impact of different k-values on the quality of the scene graph nodes. The k-value determines how many images of the object will be considered when generating the scene graph node. A higher value of k could result in a more accurate representation of the object, but could also introduce noise or inconsistencies as different views may have different embeddings. We note that increasing the k-value highly increases the computational cost. It is therefore important to find a k-value which balances these trade-offs. In this analysis we consider the following k-values: $k = 1$, $k = 3$, and $k = 10$.

To evaluate the quality of the cropping methods and k-values, we conducted the following experiment: Scene graphs were generated using three cropping methods with k values of 1, 3, and 10, resulting in nine different node generation approaches. Then, we CLIP-embed all the object **labels** in the scene, and calculate the similarity between the embedded image crops and labels. For each object, we select the label with the highest cosine similarity, and compare it to its ground truth label. Note that we use the semantic accuracy as an evaluation metric, since there is no context-awareness mechanism or learning in this experiment. The accuracy represents the percentage of nodes that are embedded closest to their label in the CLIP embedding space. Table 4.1 shows the experimental results.

The square crop method (C) with k=10 yielded the highest semantic accuracy. In this setting, the embeddings were closest to their correct labels in the CLIP embedding space 62.2% of the time. For ScanSG, we therefore used the square crop method with k=10.

Table 4.1: Semantic accuracy (%) of scene graph nodes for different cropping methods and k-values. The semantic accuracy is defined as the percentage of nodes whose embeddings were closest to their correct text label in CLIP embedding space. The highest accuracy, 62.2%, was achieved using the square crop method with $k = 10$. The letters A-D refer to the cropping methods in Figure 4.1.

	k-value		
Cropping method	1	3	10
Tight crop (A)	57.6	58.9	60.3
Tight masked crop (B)	49.5	50.1	52.1
Square crop (C)	57.1	58.3	**62.2**
Tight resized crop (D)	56.2	57.4	60.2

Limitations Figure 4.2 shows some examples of misclassified node embeddings. The table shows the crop of the object, the ground truth label, and the predicted labe (label with the highest cosine similarity to the object crop). We note that inaccurate node embeddings are most often caused by one of the following reasons:

- other objects occlude the object of interest (see Fig. 4.2 A, C, D)
- other objects are part of the object of interest (see Fig. 4.2 B)
- other labels are similar or synonymous to the object of interest's label (see Fig. 4.2 E)

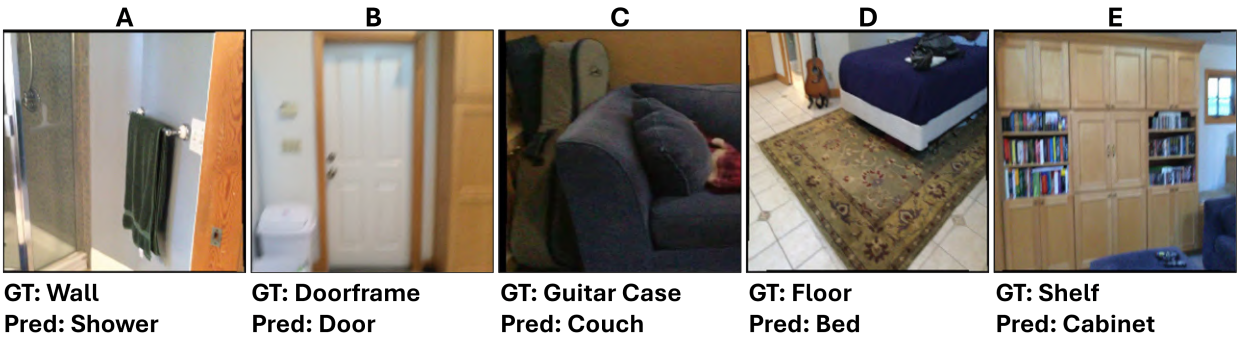


Figure 4.2: Examples of misclassified node embeddings due to limitations of the CLIP-based embedding method. The figure shows object crops with their ground truth (GT) labels and predicted (Pred) labels. Misclassifications often occur due to occlusion (A, C, D), inclusion of other objects as part of the target object (B), or label similarity/synonymy (E), highlighting inherent limitations in using CLIP for object embedding.

These examples highlight some inherent limitations of the CLIP method. However, we believe CLIP embeddings are still a good starting points as they offer a common embedding for the images in text, which is accurate in most cases. However, this also emphasizes the necessity of including some learning into the model, because the initial object CLIP-embedding can be distant from its corresponding text-label embedding.

4.1.2 Adjacency

To determine the optimal edge threshold for scene graph construction, we conduct experiments using the AttSQ+GNN model. The node embeddings and model architecture were kept constant while varying the edge threshold values at 0.2, 0.5, 0.8, and random edges. We trained the model for 200 epochs (with early

stopping) and evaluated the model performance on the test set, for each dataset. Table 4.2 presents the instance accuracy of AttSQ for each threshold value.

Table 4.2: AttSQ+GNN instance accuracy for different edge threshold values. The table shows that sparser graphs, with a lower threshold of 0.2, achieve the highest accuracy (20.1%), while increasing the threshold significantly reduces performance. All threshold values perform better than random edges, indicating that appropriate edge assignment adds value to the model.

Threshold alue (m)	**Instance Accuracy (%)**
0.2	**20.1**
0.5	14.6
0.8	9.8
Random edges	4.0

The results indicate that sparser graphs, with a lower threshold of 0.2, achieve the highest accuracy of 20.1%, significantly outperforming the more densely connected graphs (9.8% accuracy at a 0.8 threshold) and random edges (4.0% accuracy). The experiments demonstrate that a lower threshold (which results in fewer edges) is more effective for model performance, while excessively dense graphs decrease accuracy. No experiments were conducted without any edges, as this would eliminate the graph's structural information. Overall, sparse edge connections enhance the model's ability to learn from the scene graph.

4.2 Visual Grounding Models

In this section, we assess the performance of each model described in Section 3.2 using the proposed ScanSG scene graph dataset. The evaluation metrics and hyperparameters common to all experiments are described below, along with the specific implemention details of each experiment, such that the results can be reproduced. Table 4.3 summarizes all experiments conducted and qualitative results. Detailed description of methods and quantitative results can be found in the relevant sections.

4.2.1 Experimental Setup

Dataset The test/train split from the original ScanQA paper [4] is used. Multi-answer questions are removed such that each question has exactly one answer. Unless otherwise stated, the training set consits of 92% of the dataset with 19,500 questions about 555 scenes. The test set consists of the remaining 8% with 1,581 questions about 71 scenes. The scenes in the training and test set are kept separate, such that during testing, the model sees new questions about new scenes.

Hyperparameters For the baseline model, no training is required, hence none of the following hyperparameters are relevant. For all other models, a batch size of 64 was used for all experiments. The Adam optimizer was used with a learning rate tuned to each model. The loss function used was the cross-entropy loss, except if stated otherwise. The models were trained for 200 epochs, with early stopping.

Evaluation metrics We compare the performance of each model on the test set of the ScanSG dataset, and report the results in terms of best EM@1 (same as instance accuracy) and EM@5. EM@K is the percentage of questions for which the correct answer is in the top-K predicted answers. We report two measures of EM@K to understand the performance at different levels of accuracy, and monitor these metrics during training to understand how the models learn. An increasing EM@5 score indicates that the model is learning to predict the correct answer with higher confidence, while an increasing EM@1 score indicates that the

Table 4.3: Summary of experiments conducted.

Experiment	Research Question	Method	Qualitative Results
1	What is the performance of Baseline, Baseline+GNN, AttSQ, AttSQ+GNN on the generated ScanSG+ScanQA dataset?	Train and evaluate the models.	AttSQ scores the highest, followed by Baseline, AttSQ+GNN and finally Baseline+GNN.
1	Does the GNN component increase model performance?	Compare the performance of Baseline against Baseline+GNN, and the performance of AttSQ against AttSQ+GNN.	No, adding a GNN component decreases model performance.
1	Do the Cross-Attention and Question Transformer components increase model performance?	Compare the performance of Baseline against AttSQ.	Yes, the Cross-Attention mechanism and Question Transformer components increase model performance.
2	How does the quality of ScanSG node features impact model performance?	Store CLIP-embedded text-labels rather than images in the graph nodes (see Figure 4.3); train and evaluate the models. Compare model performance with results from Experiment 1.	Using CLIP-embedded images decreases model performance compared to using CLIP-embedded labels. Hence poor node features decrease model performance.
3	How does the difficulty of ScanQA questions impact model performance?	Train and evaluate the models on easy questions (see Table 4.5). Compare model performance with results from Experiment 2.	Training and evaluating on easy questions yields higher performance across models. Hence hard questions decrease model performance.
3	What words does the Cross-Attention mechanism in the highest performing model attend to?	Train and evaluate the AttSQ on easy questions and CLIP-embedded text labels. Visualize attention weights per word.	The model learns to attend to the subject of the question.
4	Does the GNN component increase model performance on hard questions?	Train and evaluate the models on hard questions (see Table 4.5). Compare the performance of Baseline against Baseline+GNN and AttSQ against AttSQ+GNN.	No, adding a GNN component decreases model performance, even on hard questions.
5	Can we increase model performance (and mitigate overfitting) using regularisation?	Apply regularisation methods to the AttSQ+GNN model and compare model performance.	No, regularization does not improve model performance.
6	Can we increase model performance (and mitigate overfitting) by increasing the dataset size?	Train the AttSQ+GNN model on increasing dataset sizes, and compare model performance for each dataset size.	Yes, increasing dataset size slightly increases performance.

model is learning to predict the correct answer with higher accuracy. We also report the semantic accuracy, Sem-EM@1. The semantic accuracy indicates correct label prediction, regardless of predicted instance. A high discrepancy between EM@1 and Sem-EM@1 indicates that the model correctly identifies the answer label, but struggles to identify the correct instance of this label. Table 4.4 summarizes the results achieved by each model for Experiments 1-4.

Loss function The cross-entropy loss is used across all experiments, unless stated otherwise.

Implementation Details The architectures are described in Section 3.2. More specifically, the experiments were conducting using the following setups:

- **Baseline** The Baseline model has no hyper-parameters.
- **Baseline+GNN** The GNN used is a 1-layer dimensionality-preserving GCN, with no activation fuction. The learning rate was 0.001.

- **AttSQ** The question transformer encoder uses 2 encoder layers, with 4 heads each. The feed-forward dimension is 512. The cross-attention decoder uses 2 encoder layers with 4 attention heads each, and a dropout of 0.01. The feed-forward dimension is 512, and the hidden dimension of the downsizing MLP is 64. The learning rate was 0.0001.
- **AttSQ+GNN** The same parameters are used as in the AttSQ implementation. The GNN used is a 1-layer dimensionality-preserving GCN, with no activation function. The learning rate was 0.0001.

4.2.2 Model Performance on ScanSG + ScanQA

Table 4.4: Test results of each model (%). The grey cell on top of each experiment shows which dataset the models were evaluated on. Easy ScanSG is the ScanSG dataset, where the graph nodes have been swapped out for the CLIP-embedded text-labels. Easy ScanQA is a subset of ScanQA, where only questions which contain the answer are kept. Bold values indicate highest score within the experiment. Overall, the AttSQ model performs best in all experiments, and higher accuracies can be seen across all models for easier datasets (Experiments 2 and 3).

	Model	**EM@1**	**EM@5**	**Sem-EM@1**
Experiment 1	ScanSG + ScanQA			
	Baseline	28.1	59.7	36.4
	Baseline + GNN	13.9	44.8	20.4
	AttSQ	**43.9**	**73.9**	**56.9**
	AttSQ + GNN	27.5	65.6	36.3
Experiment 2	Easy ScanSG + ScanQA			
	Baseline	50.7	74.4	62.7
	Baseline + GNN	20.0	53.5	29.5
	AttSQ	**59.1**	**84.4**	**73.1**
	AttSQ + GNN	35.0	74.6	45.3
Experiment 3	Easy ScanSG + Easy ScanQA			
	Baseline	67.3	91.7	83.3
	Baseline + GNN	22.4	56.3	36.0
	AttSQ	**75.7**	**94.2**	**95.3**
	AttSQ + GNN	42.5	83.5	56.3
Experiment 4	Easy ScanSG + Hard ScanQA			
	Baseline	14.9	36.7	18.1
	Baseline + GNN	14.1	42.8	18.3
	AttSQ	**34.9**	**62.4**	**43.0**
	AttSQ + GNN	23.1	56.0	28.7

Table 4.4 shows the performance of each model on the generated ScanSG+ScanQA dataset, on all 4 experiments.

Experiment 1: Model performance on the full ScanSG + ScanQA dataset

In Experiment 1, we report the EM@1, EM@5 and Sem-EM@1 achieved by each model on the test set of the full ScanSG+ScanQA dataset.

What is the performance of Baseline, Baseline+GNN, AttSQ, AttSQ+GNN on the generated ScanSG+ScanQA

dataset? The best performing model is AttSQ, achieving 43.9% EM@1, 73.9% EM@5 and 56.9% Sem-EM@1.

The Baseline and AttSQ+GNN display similar performances, with EM@1 of 28.1% and 27.5% respectively. The AttSQ+GNN model does however have a slightly higher EM@5 than the Baseline (59.7% against 65.5%), indicating higher-confidence predictions. Higher confidence for AttSQ+GNN compared to the Baseline is to be expected since the AttSQ+GNN model learns from the data, unlike the Baseline. We note that the performance of AttSQ+GNN is unexpectedly low, and investigate this in the following experiments.

The Baseline+GNN model has the lowest performance. This can be explained by the fact that the Baseline+GNN model computes the cosine similarity between a dynamic, learnable embedding (nodes) and a static, pre-computed embedding (question). This method incentivises the model to update node features such that they better align with the questions they answer. However, as some questions have the same answer but very different CLIP-embeddings, this creates conflicting training examples, and the noisy data impedes model learning.

Finally, for all models, the Sem-EM@1 is higher than the corresponding EM@1. This is expected as 1) the model may confuse instances of the same semantic type or 2) the questions may not be specific about which instance of the object it refers to (e.g. the question "What color is the chair?" does not specify which chair it is referring to).

Does the GNN component increase model performance? Comparing the Baseline against Baseline+GNN, and AttSQ against AttSQ+GNN, we observe that in both cases adding a GNN decreases the EM@1 score (by 14.2 and 16.4 percent points respectively). We note that while the no-GNN model counterparts have no context awareness, and can only learn the scene structure through previous questions about the scene, perform these do perform better. Unfortunately, our attempts to enhance the models' reasoning capabilities by incorporating a GNN component did not yield the desired improvement.

Do the Cross-Attention and Question Transformer components increase model performance? Comparing the Baseline against the AttSQ model, we see that adding cross-attention and learnable question embeddings significantly improves performance (EM@1 is increased by 15.8 percent points). The baseline is expected to score relatively low since it relies on static, pre-computed text and image embeddings and cannot learn from the data. While this makes the baseline a highly generalisable model, it also means that the model performs poorly on samples where CLIP(question) has a dissimilar pre-computed embedding than CLIP(answer). Since CLIP was trained on descriptive captions rather than questions about an image, it is expected that question embeddings may not align with their associated answer.

Overall, Experiment 1 shows that the AttSQ model performs the best, indicating that the cross-attention mechanism with the learned word-level embeddings works better than the cosine similarity of fixed sentence-level embeddings. This also indicates that the GNN contributes to poorer performance.

Experiment 2: Model performance on Easy ScanSG + ScanQA

How does the quality of ScanSG node features impact model performance? To understand the impact of the node feature quality on the model performance, we replace object image crops with CLIP-embedded object labels in the graph nodes, creating the 'Easy ScanSG' dataset. This modification provides 'ground truth object embeddings' for comparison with the original ScanSG embeddings. The only difference between Experiment 1 and Experiment 2 is thus the ScanSG node features used to train the model. Figure 4.3 illustrates this setup. We train all models on this new Easy ScanSG+ScanQA dataset, and report the results of this experiment in Table 4.4.

We note that compared to Experiment 1, all models perform better in this easy ScanSG setting. This indicates that poor node features contribute to decreased performance. The limitations of CLIP causing this were illustrated in Figure 4.2, and are further discussed in Section 5. These results highlight the importance of high quality node embeddings. AttSQ is once more the best performing model with EM@1 of 59.1%.

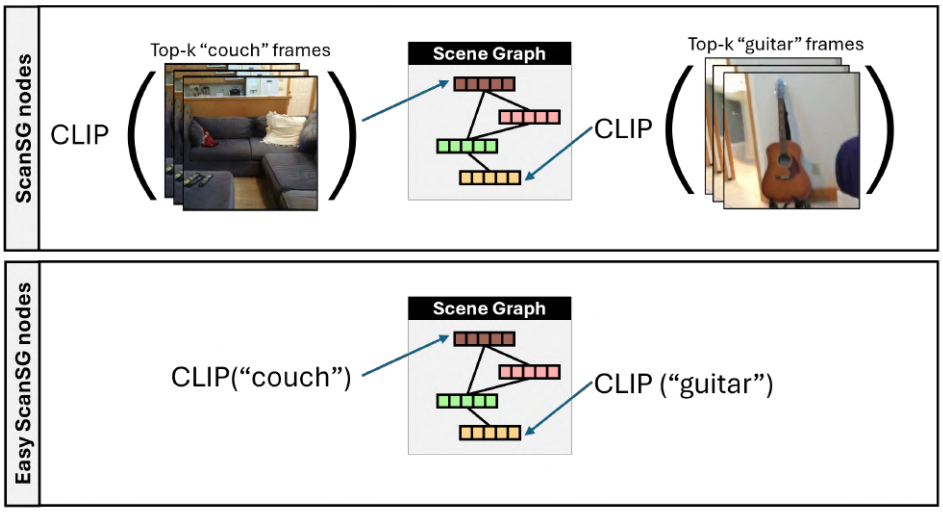


Figure 4.3: Difference between ScanSG and Easy ScanSG node features. ScanSG graphs are constructed from the weighted average CLIP embedding of the top-k object image crops, as previously described. In Easy ScanSG, on the other hand, we CLIP-embed each object's text label into the graph nodes.

Experiment 3: Model Performance on Easy ScanSG + Easy ScanQA

Experiments 3 and 4 investigate the impact of the question dataset on model performance. For the following experiments, we split the ScanQA dataset into the following smaller datasets:

- Easy ScanQA dataset: contains only questions which contain the answer, such as "What color is the chair?", where the most relevant object to identify is "chair". Train set: 11806 questions about 555 scenes. Test set: 1083 questions about 71 scenes.
- Hard ScanQA dataset: contains only questions which do not contain the answer. Train set: 7644 questions about 552 scenes. Test set: 498 questions about 71 scenes.
- Full ScanQA dataset: contains all questions, same dataset as used in previous experiments.

Table 4.5 shows examples of easy and hard questions.

Table 4.5: Examples of easy and hard ScanQA data. The easy data contains the answer within the question, whereas the hard questions refer to another object, which is not mentioned in the question.

Question	**Short Answer (Instance ID)**	**Long Answer**	**Type**
What color is the couch?	couch (1)	brown couch	Easy
Where is the white wooden door?	door (54)	kitchen door	Easy
What is in the right corner of the room by the curtains?	cabinet (8)	cabinet with tv sitting in it	Hard
What is in the corner of the bath?	shower (4)	shower	Hard

How does the difficulty of ScanQA questions impact model performance? In Experiment 3, we assess the

model performance on an easy dataset consisting of the easy ScanQA dataset, and the easy ScanSG datasets. Table 4.4 shows that the accuracy is significantly higher in this setting than in all other experiments, with the AttSQ model achieving an EM@1 accuracy of 75.7%. This indicates that easier questions contribute to higher model performance.

This experiment also achieves a near-perfect semantic accuracy of 95.3%. We note that while the answer appears within the question in the easy dataset, only the object label is obvious, and not the specific instance. The model must often randomly select an instance of the correct object type, which explains the large discrepancy between the instance and semantic accuracies.

What words does the Cross-Attention mechanism in the highest performing model attend to? Figure 4.4 shows an example of the attention pattern of the AttSQ model on an easy test question. It can be seen that the model has succesfully learned to attend to the subject of the question. More examples of this attention pattern are provided in the Appendix A.

Experiment 4: Model Performance on Easy ScanSG + Hard ScanQA

Does the GNN component increase model performance on hard questions?

In all previous experiments, we noted that the GNN component decreased the performance. However, since easy questions are self-contained and do not require contextual information, we hypothesized that the GNN may have introduced unnecessary complexity to the model. However, hard questions do need contextual information to be answered and may therefore benefit from a GNN component. In Experiment 4, we investigate whether the GNN component is beneficial in the setting of hard ScanQA dataset.

Table 4.4 shows that the addition of a GNN component still decreases performance (but less so than in previous experiments, with a 11.8 percent point difference between AttSQ and AttSQ+GNN). We offer three possible explanations for this:

- Propagating CLIP embeddings of different objects with a GNN may not effectively encode contextual information, possibly resulting in embeddings that poorly align with the intended context.
- The difficulty of the questions may largely contribute to the reduced performance. For example, questions like ”What is next to the fridge?” are challenging because multiple objects may fit the description, but the dataset specifies only one correct answer. This highlights a limitation in the dataset’s question-answer structure.
- GNNs typically require large amounts of data to perform well, and the limited number of hard questions in this dataset may not be sufficient to fully leverage the benefits of contextual information provided by the GNN.

4.2.3 Addressing Overfitting

Experiment 5: Model Regularisation

Can we increase model performance (and mitigate overfitting) using regularisation? We note that all models (except AttSQ in the Easy ScanSG + Easy ScanNet setting) display some level of overfitting, as shown in Figure 4.5. Overfitting is commonly caused by a model that is too complex, not regularized enough, not trained on enough data, or trained on poor quality data. To address this issue, we experiment with the following corrective approaches:

- Weight decay (L2): we vary weight decay between 1e-1 and 1e-4 in .1 increments and report the highest accuracy achieved.
- Dropout: we vary dropout between 0 and 0.3 in 0.05 increments and report the highest accuracy achieved.

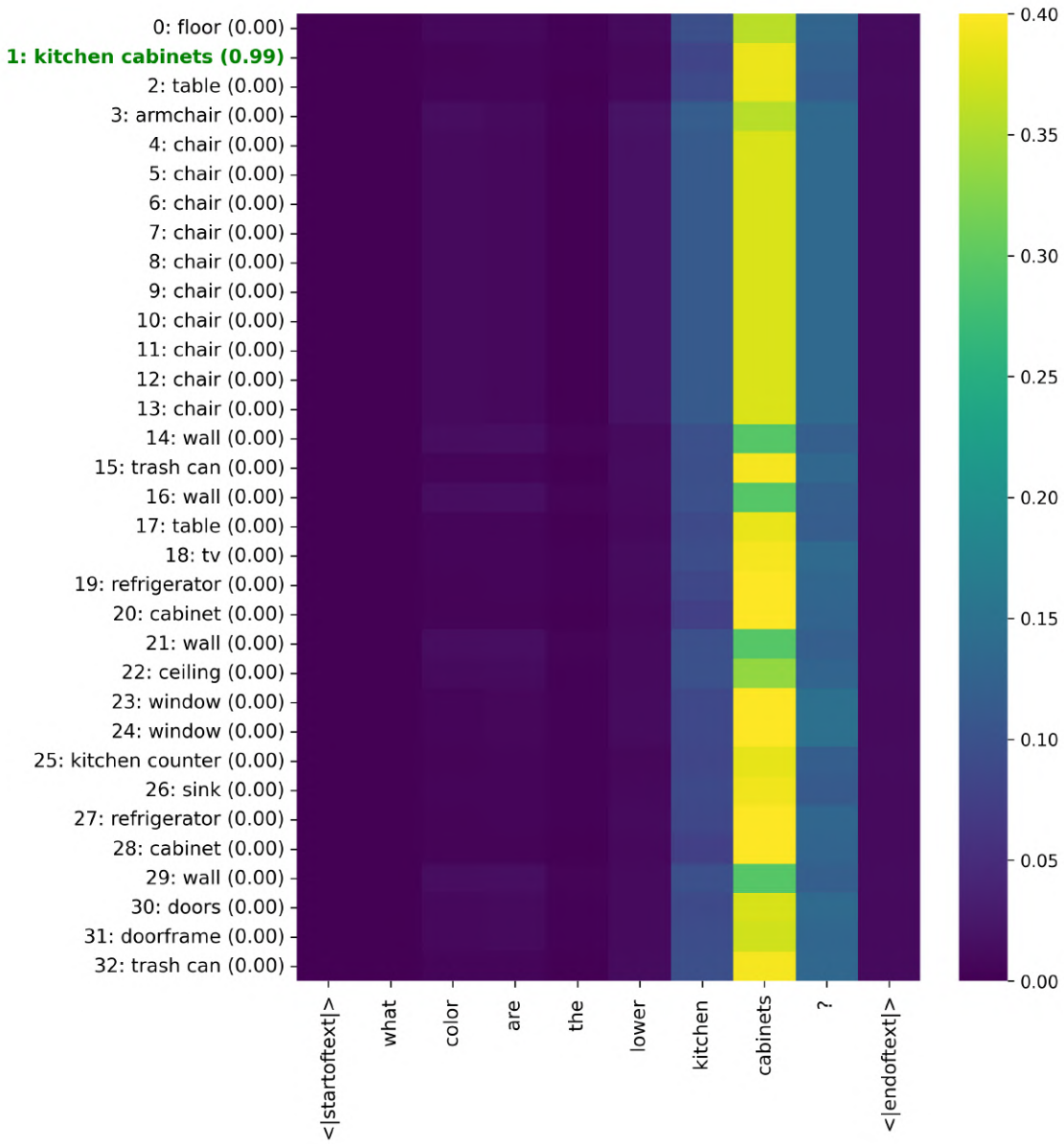


Figure 4.4: Example attention pattern of the AttSQ model on Easy ScanSG + Easy ScanQA test data. The x-axis shows word-level tokens, including the start of sentence and end of sentence tokens. The y-axis shows all object instances present in the scene, and the prediction probability in brackets. Green font indicates the GT label matched the model prediction. The attention pattern shows that the trained model successfully identifies the subject of the question.

- Increased batch size: we increase batch size to 128, 256 and 512 and report the highest accuracy achieved.
- Decreased LR: we vary LR between 1e-4 and 1e-6 in 0.05 increments and report the highest accuracy achieved.
- Focal loss: we change the training loss to focal loss with $\gamma = 2, 5, 10$ and report the highest accuracy achieved.
- Decreased model size: for both the question encoder and cross-attention block, we set number of heads and layers to 2, feed-forward dimension to 128. For the cross-attention block, we set the

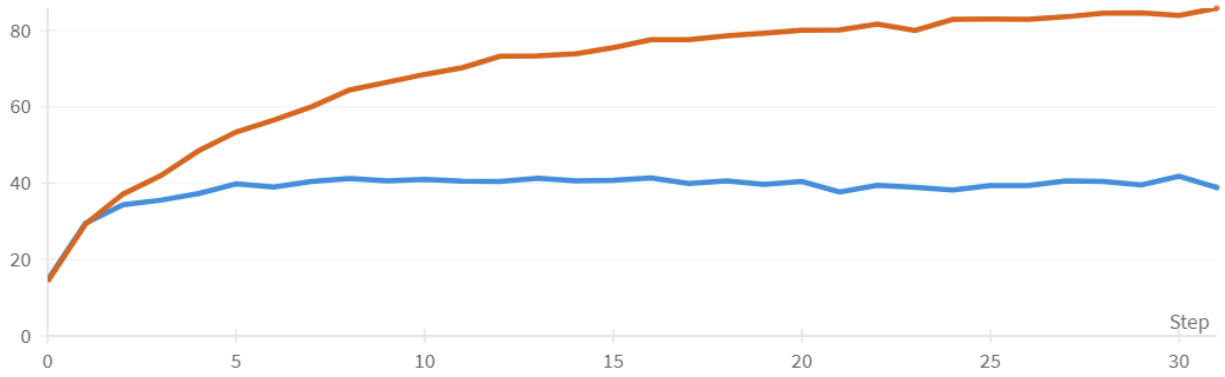


Figure 4.5: Training curve of AttSQ+GNN on the Easy ScanSG + Easy ScanQA dataset. Orange is train accuracy, and blue is test accuracy. This figure indicates that the model is overfitting.

downsizing MLP feed-forward dimension to 32. We leave the LR unchanged (LR=1e-4)

We retrain the ScanSG-GNN model with these changes on a smaller, 25% of the dataset (5070 questions about 236 scenes). We evaluate the performance of each model with these changes on the test set of the ScanSG dataset, and report the best results in Table 4.6.

Table 4.6: Instance accuracy of the AttSQ+GNN model on a small subset of ScanSG + ScanQA (25% of the full dataset). None of the regularization methods improve model performance.

	EM@1
No changes	**24.6**
Batch size = 512	21.6
Weight decay = 1e-3	20.0
LR = 1e-5	22.6
Focal Loss	23.3
Dropout = 0.2	21.9
XS model	22.6

We note that none of the regularization methods described above improved generalization. As a last experiment, we therefore investigate the impact of data quality on the model's ability to generalize.

Experiment 6: Dataset Size

Can we increase model performance (and mitigate overfitting) by increasing the dataset size? To understand the effect of dataset size on model performance, we train AttSQ+GNN on ScanSG+ScanQA datasets of sizes 5000, 10,000, 15,000 and 20,000. We report the obtained accuracies in Figure 4.6.

Increasing dataset size does improve model performance. By quadrupling the dataset size (from approx 5,000 to 20,000), the accuracy increased by 10.4 percent points. Increasing dataset size could therefore also be a promising avenue for improved model performance. However, it is difficult to predict exactly how many data points are required.

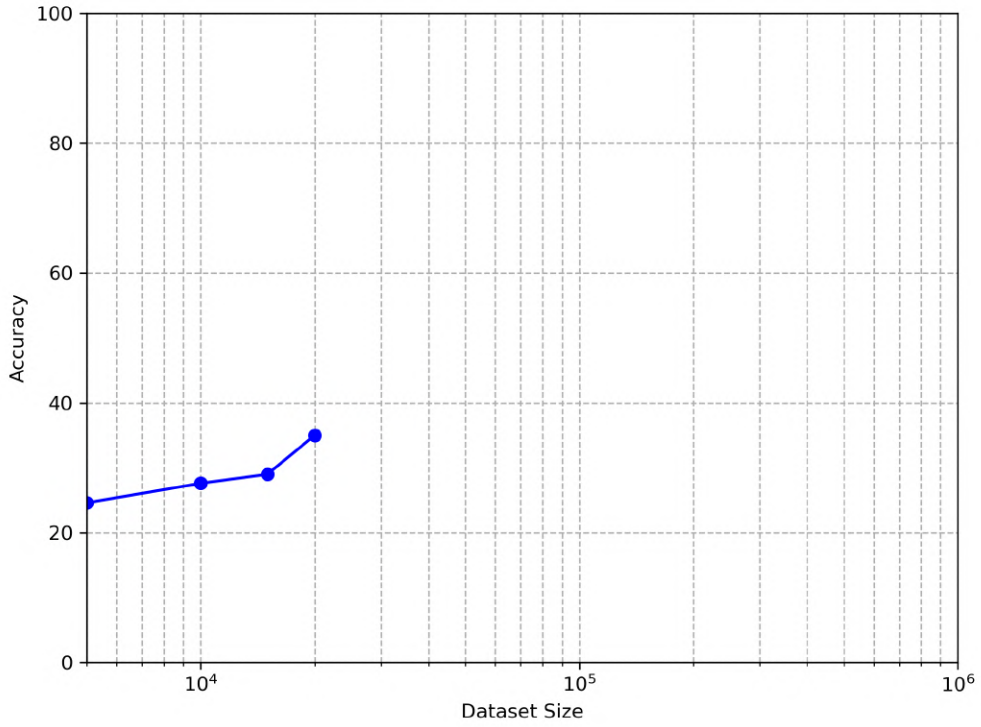


Figure 4.6: Accuracy of the AttSQ+GNN model as a function of dataset size (log scale).

Chapter 5

Discussion

The project was divided into two main parts: first, we developed a scene graph dataset for ScanNet, referred to as ScanSG, designed to be used alongside the ScanQA question dataset. Second, we proposed a graph-text VG model, along with a baseline and investigated the effectiveness of components like GNNs and cross-attention mechanisms. The current section discusses the meaning of our results and addresses limitations.

Scene Graph Generation In this project, we developed a heuristics-based scene graph dataset for ScanNet, which we termed ScanSG. This dataset leverages the multimodal capabilities of CLIP to enable effective comparison between graph nodes and natural language queries.

Firstly, we note that scene graphs do not possess a ”ground truth” as the scene graph generation from scene graphs requires design choices regarding node features and connectivity. To evaluate the scene graphs' quality, we studied several design choices, such as the k-value and cropping method, by assessing the alignment between image embeddings and object label embeddings when varying these parameters. To select the threshold value, we evaluated the performance of the AttSQ+GNN model on graphs with varying thresholds. However, we note that this method comes with the risk of fitting the data to the task, rather than fitting the task to the data, and could lead to overfitting.

Secondly, we used oracle segmentation to produce high quality graphs, however we note that this method relies on high-quality pre-existing segmentation data, and may therefore be less generalizable to real-world applications where such data is unavailable.

Additionally, we observed certain limitations in CLIP's performance: its embeddings are 512-dimensional, requiring large models for processing, and it struggles with occlusions, crowded spaces, and context-dependent objects (e.g., embedding a doorframe often results in it being labeled as a ”door”). These challenges highlight that, while CLIP offers substantial advantages, obtaining representative features for objects using CLIP remains difficult. Future work could explore alternative encoding methods, such as the pre-trained models presented in Section 2.4, integrating ”context nodes” (separate nodes which encode context) or incorporating additional types of information like position or edge labels. Our findings are consistent with other studies that emphasize the critical role of node features in determining the quality of VG results.

QA dataset Our models encounter difficulties in managing complex relational reasoning. A significant challenge lies in creating a question dataset that is both diverse and well-structured. The size of the QA dataset used in this study is relatively small, comprising 20,000 questions. This limited dataset size may negatively impact the model's ability to generalize to unseen data, potentially leading to overfitting. For comparison, larger 2D-VQA datasets such as VQAv2 and GQA contain 1.1 million and 22 million questions, respectively [19]. The challenge of generating a high-quality question-answer dataset is further compounded by the difficulty of crafting complex questions with unique answers, as well as the wide variety of question types (e.g., negation, relational, descriptive). To improve dataset quality, future efforts could focus on creating a dataset

with a broad range of questions, ensuring unique (or fully listed) solutions, consistent reasoning patterns, and a classification system for questions. This would enable a more nuanced understanding of the model's strengths and weaknesses across different types of questions.

VG Models We proposed an attention-based model for visual grounding, incorporating scene graphs to introduce contextual knowledge. To our knowledge, this is the first model to apply GNNs for propagating CLIP embeddings. Unfortunately, contrary to our expectations, the inclusion of GNNs negatively impacted model performance, suggesting that the addition of contextual information might not always be beneficial in this context. We offer three potential explanations for this outcome: (1) the inherent difficulty of the questions posed by the dataset, (2) the insufficient data to fully leverage the GNN structure, and (3) the possibility that propagating CLIP embeddings through GNNs alters them to the extent that they no longer accurately represent the original concepts. To better understand this transformation, future research could generate text based on the propagated node features, thereby clarifying what the GNN-modified embeddings actually represent.

Our experiments identified several factors contributing to lower model performance, including the use of GNNs, suboptimal node features, challenging questions, and the small size of the dataset. However, we also showed that the use of a cross-attention mechanism and learnable question embeddings significantly improved model performance.

Although direct comparisons with other methods are difficult due to differences in evaluation datasets (we used a VG+VQA dataset while others use VG datasets), our model's performance (43.9% for AttSQ on the full ScanSG+ScanQA dataset) falls within the range reported by other papers (35.6-65.9%). Future work should consider evaluating the model on a standard VG dataset for benchmarking. Pretraining the model on other VG datasets, such as ScanRefer, could also be a promising avenue to further enhance performance. Adding a decoder to our existing model could solve the VQA task, and could be used to investigate the impact of training the VG task alongside the VQA task on this dataset.

Chapter 6

Conclusion

In this work, we proposed and evaluated a novel approach to scene graph generation and visual grounding using CLIP embeddings. Our contributions can be summarized in two main points:

Firstly, we developed a simple, heuristics-based scene graph dataset for ScanNet, termed ScanSG, which effectively leverages the mixed-modal capabilities of CLIP. This allows for more accurate comparisons between graph nodes and natural language queries. Through our experiments, we determined that the square cropping method (square crop centred around the object of interest, no masking, distorting or sampling) worked best for CLIP-embedding images (achieving a similarity score to object text-labels of 62.2% for k=10). We observed that increasing the k-value enhances the similarity between image and label embeddings, albeit with a trade-off in computational cost. The relatively low overall scores could be attributed to factors like occlusion, large crops, synonyms, and the presence of objects within objects. Additionally, we found that threshold-based edges outperformed random edges in improving model performance.

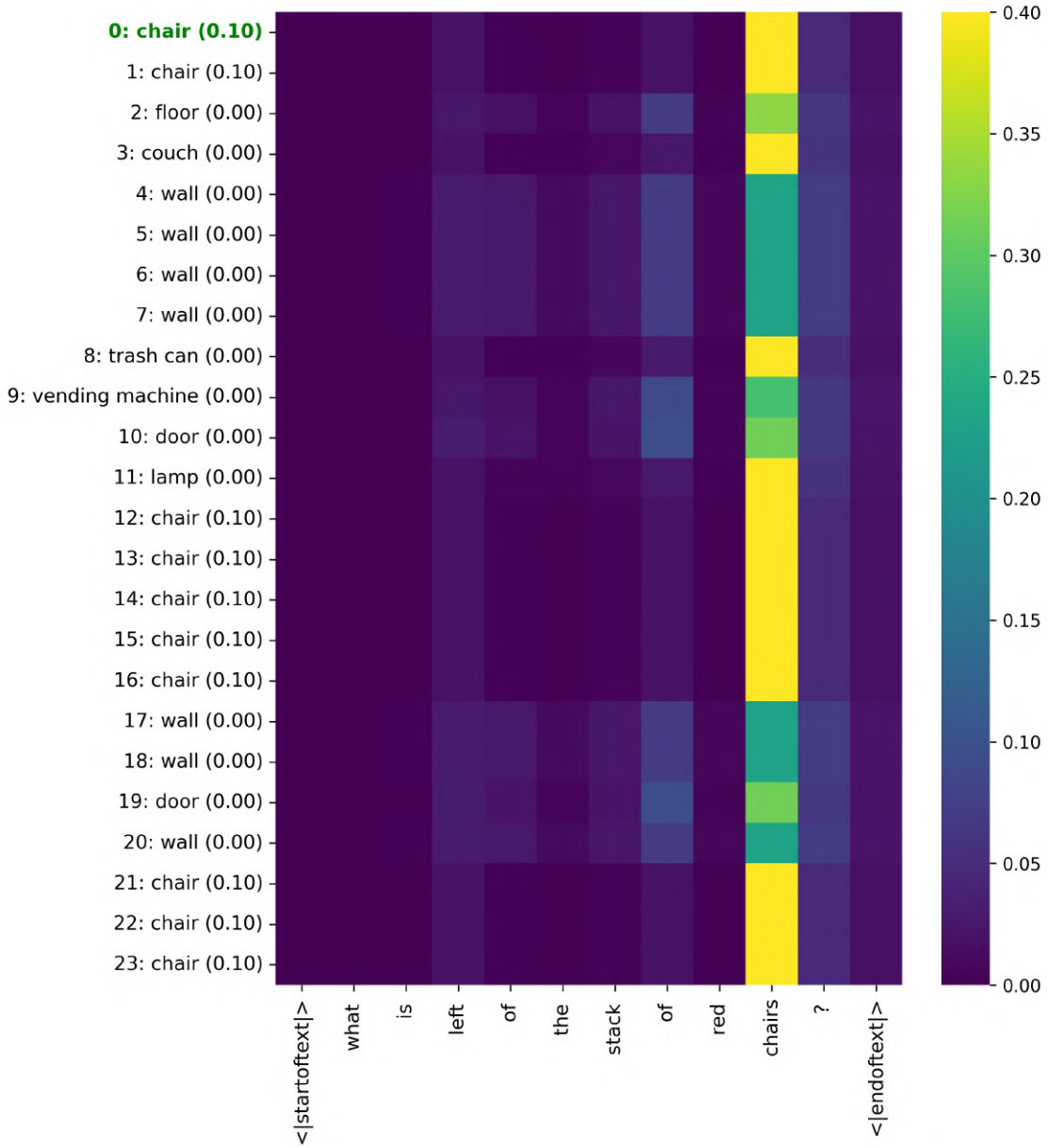
Secondly, we introduced a novel method that integrates scene graphs with queries within a visual grounding pipeline. While previous studies have used CLIP or GNNs independently, to our knowledge, no other work has combined CLIP embeddings with a GNN for propagation. Our best model achieved a performance score of 43.9, significantly outperforming the baseline, which scored 28.1%. However, we identified that poor node embeddings and the inherent difficulty of the questions are major factors limiting performance. When node embeddings were improved, performance increased notably, and under the easiest settings (with improved node embeddings and simpler questions), the performance reached 75%. This highlights the importance of question complexity and the need for different ”reasoning paths” or attention patterns, making the task particularly challenging. Despite our efforts to enhance the model using a GNN, we found that retaining the original CLIP embeddings yielded better results than propagating them through the GNN. We propose three potential reasons for this: (1) some question data might be too challenging to answer, (2) the dataset size may be insufficient, or (3) this method may not be effective for encoding context awareness, and alternative approaches could potentially offer better results.

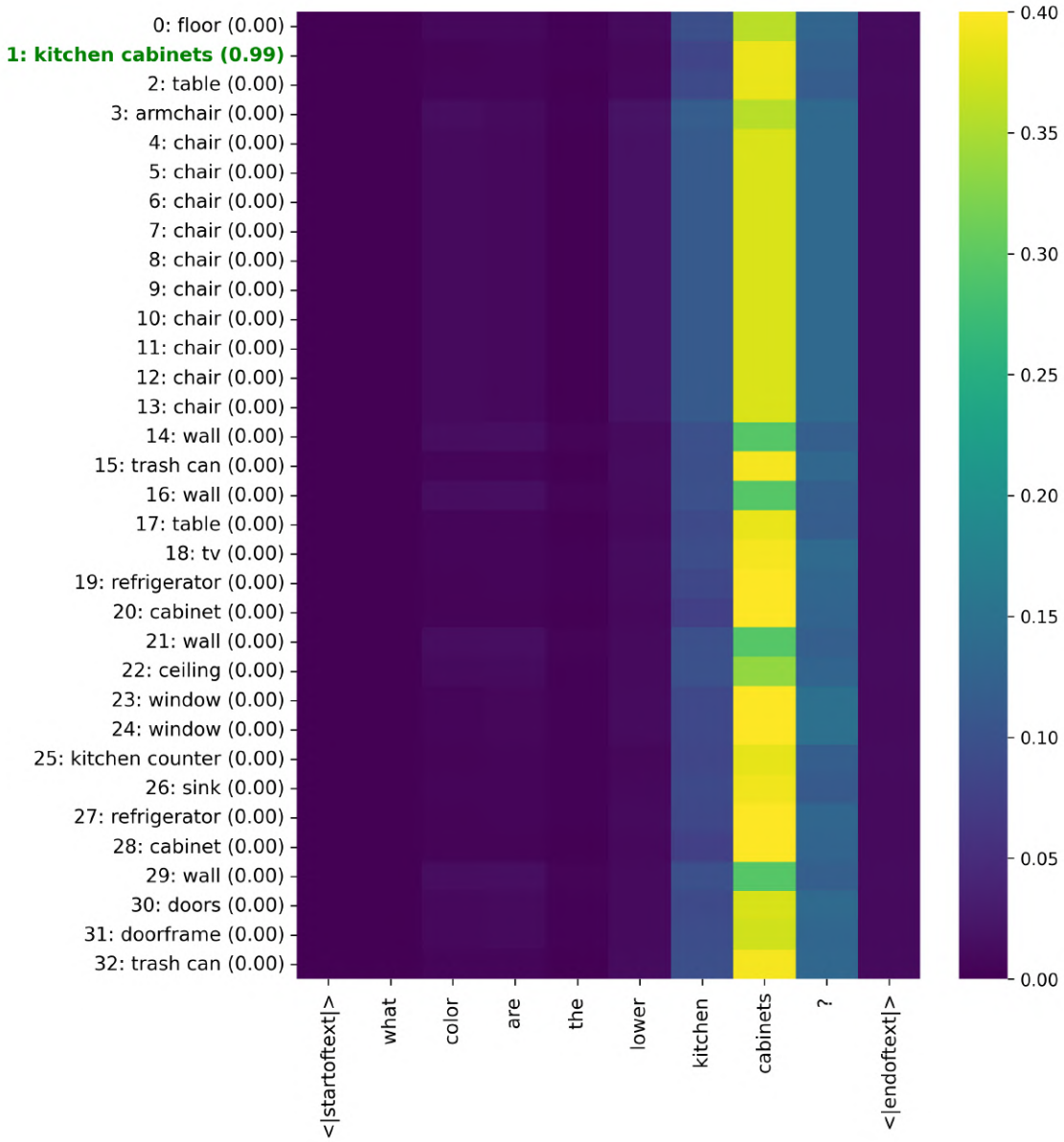
Overall, we have thoroughly explored and assessed the performance of integrating scene graphs with CLIP embeddings in visual grounding. Future work should focus on refining question datasets and improving node embeddings to enhance model performance further. Additionally, exploring alternative methods for encoding contextual information in visual grounding tasks could lead to more effective and robust models.

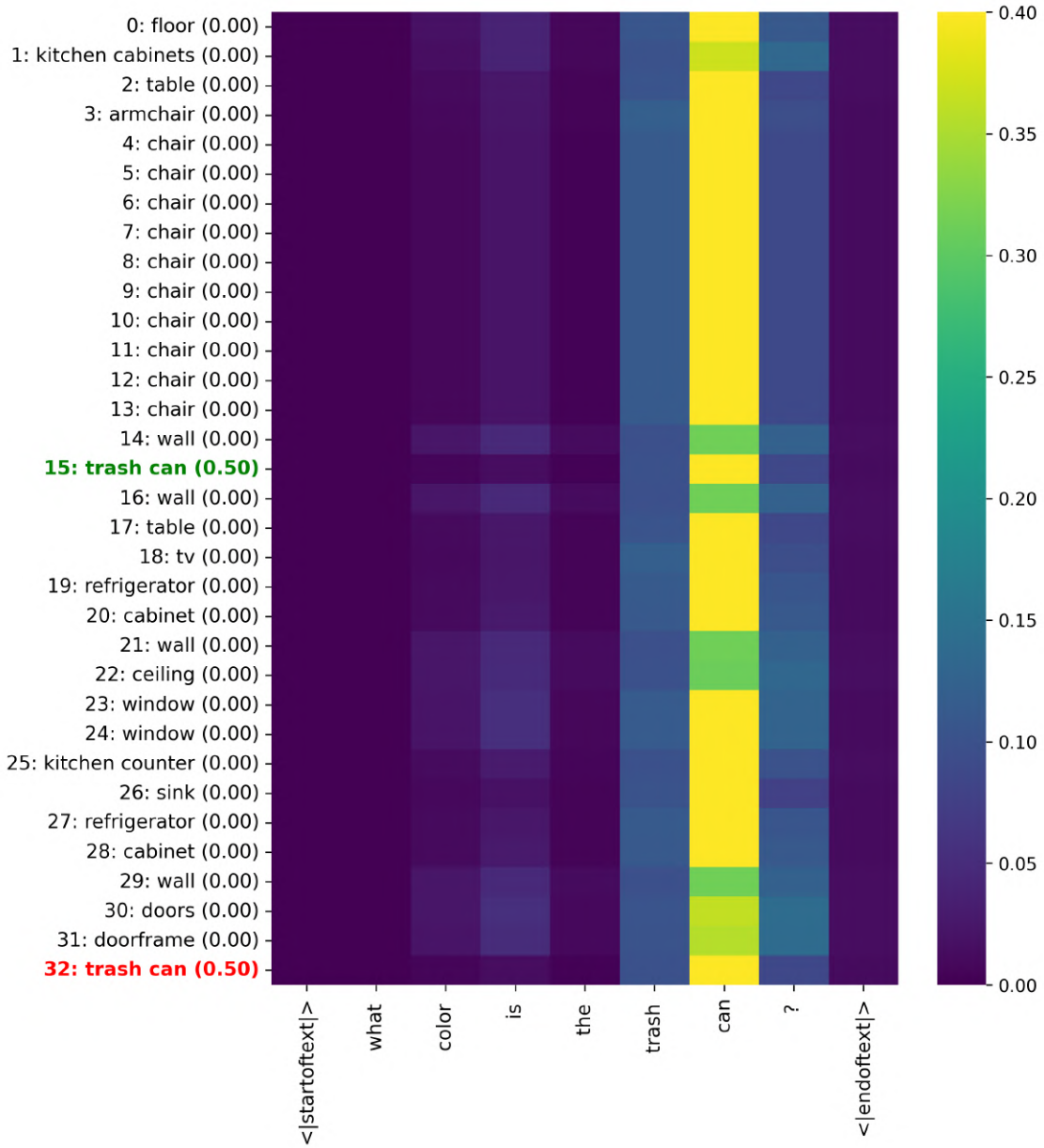
Appendix A

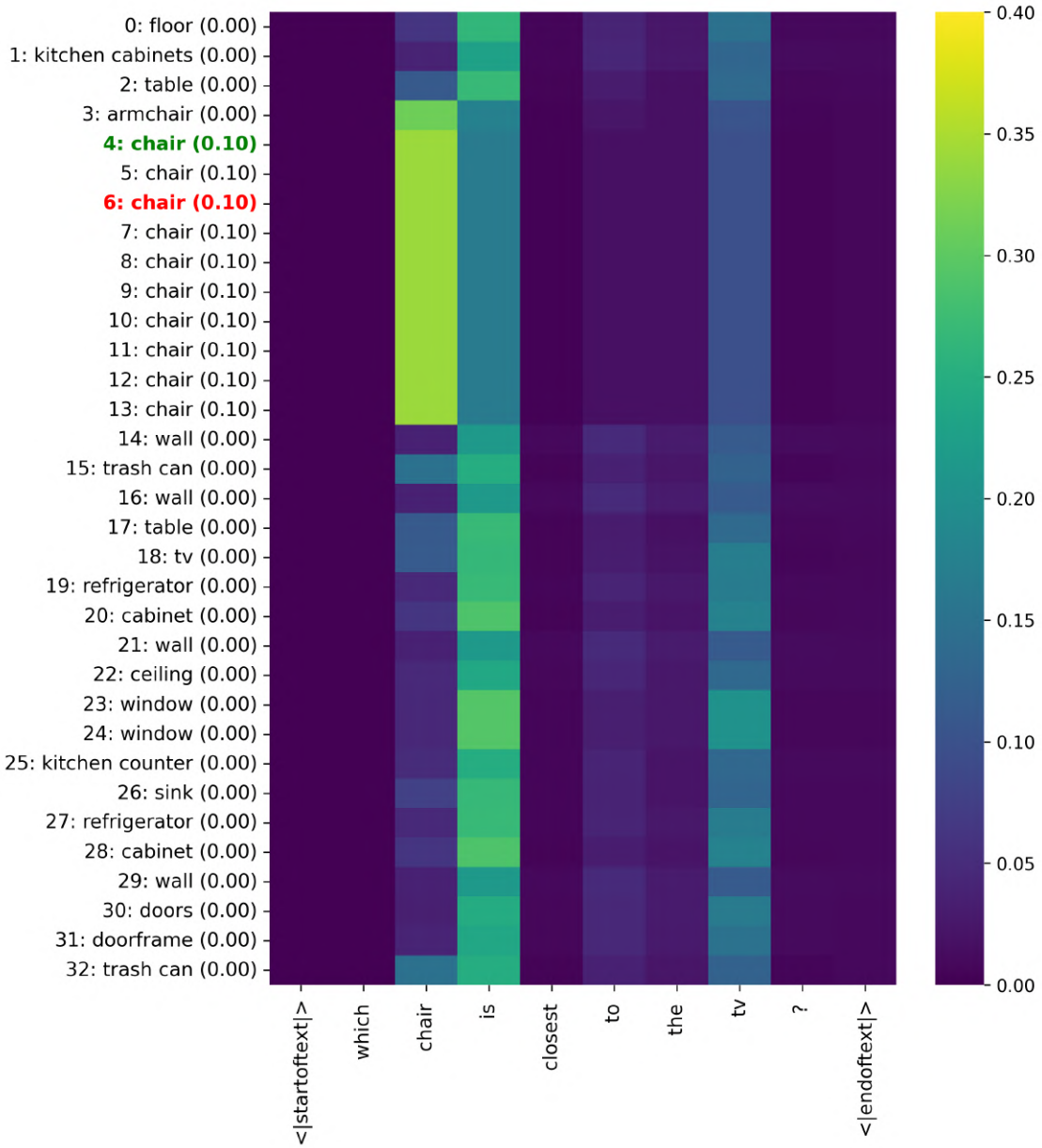
Appendix A

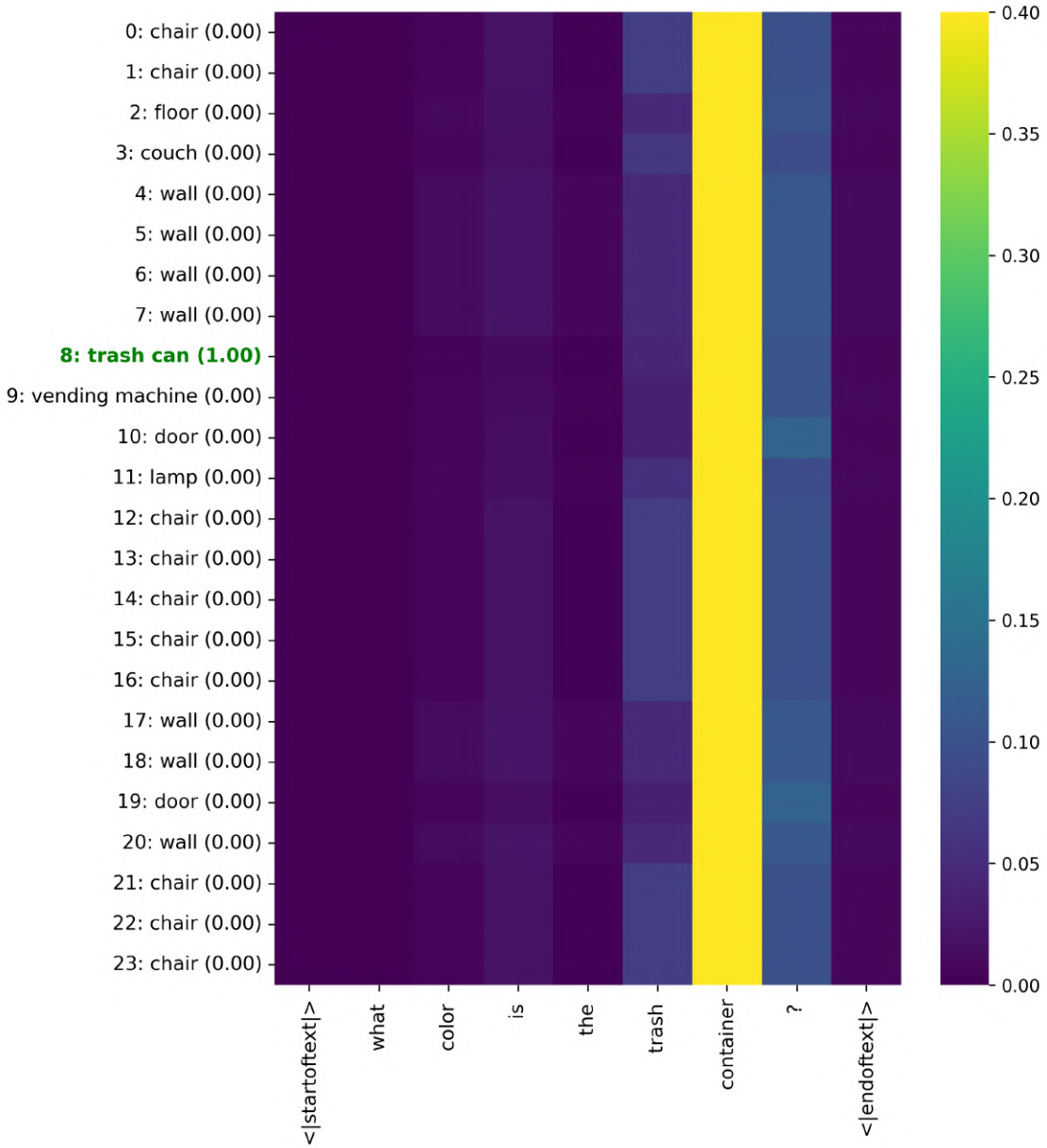
The following figures present more examples of the attention pattern of the AttSQ model on Easy ScanSG + Easy ScanQA test data. The x-axis shows word-level tokens, including the start of sentence and end of sentence tokens. The y-axis shows all object instances present in the scene, and the prediction probability in brackets. Green font indicates the GT label and red font indicated the predicted instance (if this was different from the GT instance). The attention pattern shows that the trained model successfully identifies the subject of the question.


0: chair (0.10)
1: chair (0.10)
2: floor (0.00)
3: couch (0.00)
4: wall (0.00)
5: wall (0.00)
6: wall (0.00)
7: wall (0.00)
8: trash can (0.00)
9: vending machine (0.00)
10: door (0.00)
11: lamp (0.00)
12: chair (0.10)
13: chair (0.10)
14: chair (0.10)
15: chair (0.10)
16: chair (0.10)
17: wall (0.00)
18: wall (0.00)
19: door (0.00)
20: wall (0.00)
21: chair (0.10)
22: chair (0.10)
23: chair (0.10)
<|startoftext|>
what
is
left
of
the
stack
of
red
chairs
?
<|endoftext|>
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


0: floor (0.00)
1: kitchen cabinets (0.99)
2: table (0.00)
3: armchair (0.00)
4: chair (0.00)
5: chair (0.00)
6: chair (0.00)
7: chair (0.00)
8: chair (0.00)
9: chair (0.00)
10: chair (0.00)
11: chair (0.00)
12: chair (0.00)
13: chair (0.00)
14: wall (0.00)
15: trash can (0.00)
16: wall (0.00)
17: table (0.00)
18: tv (0.00)
19: refrigerator (0.00)
20: cabinet (0.00)
21: wall (0.00)
22: ceiling (0.00)
23: window (0.00)
24: window (0.00)
25: kitchen counter (0.00)
26: sink (0.00)
27: refrigerator (0.00)
28: cabinet (0.00)
29: wall (0.00)
30: doors (0.00)
31: doorframe (0.00)
32: trash can (0.00)
<|startoftext|>
what
color
are
the
lower
kitchen
cabinets
?
<|endoftext|>
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


0: floor (0.00)
1: kitchen cabinets (0.00)
2: table (0.00)
3: armchair (0.00)
4: chair (0.00)
5: chair (0.00)
6: chair (0.00)
7: chair (0.00)
8: chair (0.00)
9: chair (0.00)
10: chair (0.00)
11: chair (0.00)
12: chair (0.00)
13: chair (0.00)
14: wall (0.00)
15: trash can (0.50)
16: wall (0.00)
17: table (0.00)
18: tv (0.00)
19: refrigerator (0.00)
20: cabinet (0.00)
21: wall (0.00)
22: ceiling (0.00)
23: window (0.00)
24: window (0.00)
25: kitchen counter (0.00)
26: sink (0.00)
27: refrigerator (0.00)
28: cabinet (0.00)
29: wall (0.00)
30: doors (0.00)
31: doorframe (0.00)
32: trash can (0.50)
<|startoftext|>
what
color
is
the
trash
can
?
<|endoftext|>
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


0: floor (0.00)
1: kitchen cabinets (0.00)
2: table (0.00)
3: armchair (0.00)
4: chair (0.10)
5: chair (0.10)
6: chair (0.10)
7: chair (0.10)
8: chair (0.10)
9: chair (0.10)
10: chair (0.10)
11: chair (0.10)
12: chair (0.10)
13: chair (0.10)
14: wall (0.00)
15: trash can (0.00)
16: wall (0.00)
17: table (0.00)
18: tv (0.00)
19: refrigerator (0.00)
20: cabinet (0.00)
21: wall (0.00)
22: ceiling (0.00)
23: window (0.00)
24: window (0.00)
25: kitchen counter (0.00)
26: sink (0.00)
27: refrigerator (0.00)
28: cabinet (0.00)
29: wall (0.00)
30: doors (0.00)
31: doorframe (0.00)
32: trash can (0.00)
<|startoftext|>
which
chair
is
closest
to
the
tv
?
<|endoftext|>
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


0: chair (0.00)
1: chair (0.00)
2: floor (0.00)
3: couch (0.00)
4: wall (0.00)
5: wall (0.00)
6: wall (0.00)
7: wall (0.00)
8: trash can (1.00)
9: vending machine (0.00)
10: door (0.00)
11: lamp (0.00)
12: chair (0.00)
13: chair (0.00)
14: chair (0.00)
15: chair (0.00)
16: chair (0.00)
17: wall (0.00)
18: wall (0.00)
19: door (0.00)
20: wall (0.00)
21: chair (0.00)
22: chair (0.00)
23: chair (0.00)
<|startoftext|>
what
color
is
the
trash
container
?
<|endoftext|>
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

Bibliography

[1] Panos Achlioptas, Lin Shao Fan, Jean-Emmanuel Deschaud, Leonidas Guibas, Oliver Kaick, Jonathan Lambourne, Guillem Martínez Rodriguez, John Owens, and Loic Landrieu. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–19, 2020.

[2] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering, 2016.

[3] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera, 2019.

[4] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding, 2022.

[5] Eslam Mohamed Bakr, Yasmeen Alsaedy, and Mohamed Elhoseiny. Look around and refer: 2d synthetic semantics knowledge distillation for 3d visual grounding, 2022.

[6] Daigang Cai, Lichen Zhao, Jing Zhang, Lu Sheng, and Dong Xu. 3djcg: A unified framework for joint dense captioning and visual grounding on 3d point clouds. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16443–16452, 2022.

[7] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, January 2023.

[8] Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner. Scanrefer: 3d object localization in RGB-D scans using natural language. *CoRR*, abs/1912.08830, 2019.

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017.

[10] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog, 2017.

[11] Mingtao Feng, Zhen Li, Qi Li, Liang Zhang, XiangDong Zhang, Guangming Zhu, Hui Zhang, Yaonan Wang, and Ajmal Mian. Free-form description guided 3d visual graph network for object grounding in point cloud, 2021.

[12] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023.

[13] Zoey Guo, Yiwen Tang, Ray Zhang, Dong Wang, Zhigang Wang, Bin Zhao, and Xuelong Li. Viewrefer: Grasp the multi-view knowledge for 3d visual grounding with gpt and prototype guidance, 2023.

[14] William L. Hamilton. *Graph Representation Learning*, volume 14 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2020.

[15] Dailan He, Yusheng Zhao, Junyu Luo, Tianrui Hui, Shaofei Huang, Aixi Zhang, and Si Liu. Transrefer3d: Entity-and-relation aware transformer for fine-grained 3d visual grounding. In *Proceedings of the 29th ACM International Conference on Multimedia*. ACM, October 2021.

[16] Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. Language-conditioned graph networks for relational reasoning, 2019.

[17] Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang. Multi-view transformer for 3d visual grounding, 2022.

[18] Wencan Huang, Daizong Liu, and Wei Hu. Dense object grounding in 3d scenes, 2023.

[19] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019.

[20] Md Farhan Ishmam, Md Sakib Hossain Shovon, M. F. Mridha, and Nilanjan Dey. From image to language: A critical analysis of visual question answering (vqa) approaches, challenges, and opportunities, 2023.

[21] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021.

[22] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering, 2019.

[23] Zeju Li, Chao Zhang, Xiaoyan Wang, Ruilong Ren, Yifan Xu, Ruifei Ma, and Xiangde Liu. 3dmit: 3d multi-modal instruction tuning for scene understanding, 2024.

[24] Weixin Liang, Yanhao Jiang, and Zixuan Liu. Graghvqa: Language-guided graph neural networks for graph-based visual question answering, 2021.

[25] Zhihong Lin, Donghao Zhang, Qingyi Tao, Danli Shi, Gholamreza Haffari, Qi Wu, Mingguang He, and Zongyuan Ge. Medical visual question answering: A survey. *Artificial Intelligence in Medicine*, 143:102611, September 2023.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[27] Julian Lorenz, Robin Schön, Katja Ludwig, and Rainer Lienhart. A review and efficient implementation of scene graph generation metrics, 2024.

[28] Junyu Luo, Jiahui Fu, Xianghao Kong, Chen Gao, Haibing Ren, Hao Shen, Huaxia Xia, and Si Liu. 3d-sps: Single-stage 3d visual grounding via referred point progressive selection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[29] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers, 2022.

[30] Taiki Miyanishi, Daichi Azuma, Shuhei Kurita, and Motoki Kawanabe. Cross3dvg: Cross-dataset 3d visual grounding on different rgb-d scans, 2024.

[31] Maria Parelli, Alexandros Delitzas, Nikolas Hars, Georgios Vlassis, Sotirios Anagnostidis, Gregor Bachmann, and Thomas Hofmann. Clip-guided vision-language pre-training for question answering in 3d scenes, 2023.

[32] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 01 2014.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[34] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. Languagerefer: Spatial-language model for 3d visual grounding. *CoRR*, abs/2107.03438, 2021.

[35] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation, 2023.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[37] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings, 2016.

[38] Sanghyun Woo, Dahun Kim, Donghyeon Cho, and In So Kweon. Linknet: Relational embedding for scene graph, 2018.

[39] Zhengyuan Yang, Songyang Zhang, Liwei Wang, and Jiebo Luo. Sat: 2d semantics assisted training for 3d visual grounding, 2021.

[40] Shuquan Ye, Dongdong Chen, Songfang Han, and Jing Liao. 3d question answering, 2022.

[41] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering, 2019.

[42] Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3dvg-transformer: Relation modeling for visual grounding on point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2908–2917, 10 2021.

[43] Yaoyao Zhong, Junbin Xiao, Wei Ji, Yicong Li, Weihong Deng, and Tat-Seng Chua. Video question answering: Datasets, algorithms and challenges, 2022.

[44] Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment, 2023.