

# MSc. Thesis Title

## Subtitle

Master's Thesis

Amalie Kjaer

Department of Computer Science D-INFK

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

August 22, 2024

# Abstract

The abstract gives a concise overview of the work you have done. The reader should be able to decide whether the work that has been done is interesting for him by reading the abstract. Provide a brief account of the following questions:

- What is the problem you worked on? (Introduction)
- What are the shortcomings of existing solutions? (Technical Gap)
- How did you tackle the problem? (Method)
- What were your results and findings? (Experiments)
- Why are your findings significant? (Conclusion)

The abstract should approximately cover half of a page, and does generally not contain citations.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Related Work</b>	<b>3</b>
2.1	Visual Question Answering Task (indoor + other)	3
2.2	Scene Graphs	3
2.3	Image-text pretrained models	4
2.4	Transformer encoder / decoders	4
2.5	Datasets	4
2.5.1	ScanNet	4
2.5.2	ScanQA	5
<b>3</b>	<b>Method</b>	<b>8</b>
3.1	Scene Graph Generation	9
3.1.1	Node features	9
3.1.2	Adjacency	10
3.1.3	Edge features	11
3.2	VQA Models	11
3.2.1	Baseline	11
3.2.2	Baseline + GNN	12
3.2.3	ScanSG-GNN	12
3.2.4	ScanSG-Transformer	12
3.2.5	Batching and Masking	12
3.2.6	Loss function	12
<b>4</b>	<b>Experiments</b>	<b>13</b>
4.1	Scene Graph Generation	13
4.1.1	Node Construction	13
4.1.2	Edge Construction	15
4.1.3	Edge Label Classification	17
4.1.4	Dataset Statistics	17
4.2	Visual Question Answering Model	17
4.2.1	GraphVQA	17
4.2.2	Experimental Setup	17
4.2.3	Model Performance on ScanSG + ScanQA	18
4.2.4	Addressing Overfitting	19
4.2.5	Model Performance on Ideal Dataset	20
<b>5</b>	<b>Discussion</b>	<b>22</b>

<b>6 Conclusion</b>	<b>23</b>
<b>A The First Appendix</b>	<b>24</b>

# Chapter 1

## Introduction

Give an introduction to the topic you have worked on:

- *What is the rationale for your work?* Motivate the problem, *e.g.* with a general description of the problem setting, narrowing down to the particular problem you have been working on in your thesis. Allow the reader to understand the problem setting.
- *What is the technical gap in existing work?* Briefly outline how this problem has been tackled before, and what the shortcomings of the existing solutions are.
- *What is your work doing to fix it?* Given the above background, state briefly the focus of the work.

### OBJECT IDENTIFICATION

Motivation The problem Challenges Objectives / Goals Contributions Methods Existing work Structure of the thesis Main results  
include venn diagram to explain the dataset + the task

## Chapter 2

# Related Work

The related work section has the following purposes:

- *Is the overview concise?* Give an overview of the most relevant work to the needed extent. Make sure the reader can understand your work without referring to other literature.
- *Does the compilation of work help to define the “niche” you are working in?* Another purpose of this section is to lay the groundwork for showing that you did significant work. The selection and presentation of the related work should enable you to name the implications, differences, and similarities sufficiently in the discussion section.

description of task

### 2.1 Visual Question Answering Task (indoor + other)

VQA vs 3D-QA OBJECT IDENTIFICATION!!

indoor is a subtask but it is interesting because we can include some 3D element to it - also this lays the foundation for answering harder questions later on, which cannot be answered simply with 2D methods: eg. about distances and paths, which would be interesting for navigation tasks.

- Task description
- Expected inputs / outputs
- Question ontology
- Classical methods (non-SG), strengths and limitations
- Scene Graph methods (Graph Question Answering), strengths and limitations (most of these for indoor scenes say they use graphs but they never utilise edge info)

description of data

### 2.2 Scene Graphs

- Scene graph formal definition + advantages
- Scene graph generation methods

- Graph neural networks
- Applications and limitations

description of the ”building blocks” used to create the model

## 2.3 Image-text pretrained models

## 2.4 Transformer encoder / decoders

GNN can be formulated as a

## 2.5 Datasets

ScanQA is the only question dataset exclusively about indoor scenes. Wrap text around figures? Can add this section to 'Related work' instead of an entire independent section.

### 2.5.1 ScanNet

add an example of a 3D segmented scene + segmented RGB-D to illustrate what the dataset looks like? can also add to appendix

The ScanNet dataset 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.

In this section, we report some useful dataset statistics for the scenes for which ScanQA question-answer pairs exist.

Figure XX show the type of distribution of scene types. The three most common scene type are 'Bedroom', 'Living Room' and 'Bathroom'.

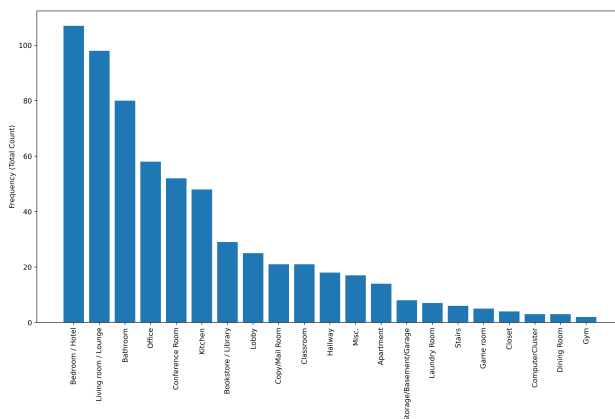


Figure 2.1: Scene type distribution

Figure XX shows the distribution of object labels. We note that the most common label is 'wall' (as all scenes contain multiple walls), followed by 'chair'.



Figure 2.2: Object label distribution **remove "other" and add to text instead how many "other" were excluded from the top-30**

**+ Avg number of object instances per scene)**

Figure XX 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: **80% of scenes have between ... and ... scenes per  $m^2$**

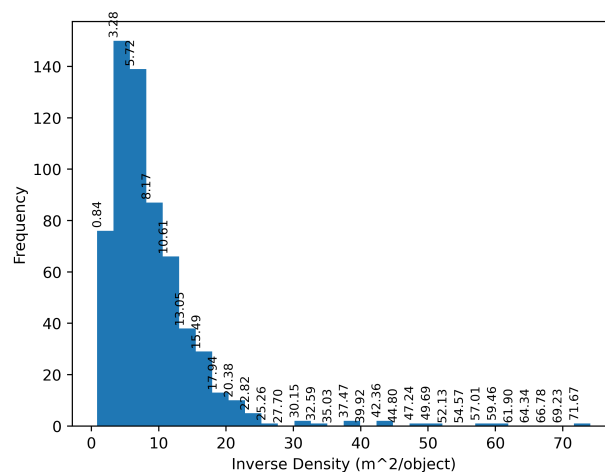


Figure 2.3: scene density distribution

## 2.5.2 ScanQA

**[Add that each question-answer pair has corresponds to one unique scene, which is specified]**



The ScanQA dataset [XX] consists of 30,238 question-answer pairs about 633 ScanNet scenes. 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 XX 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.

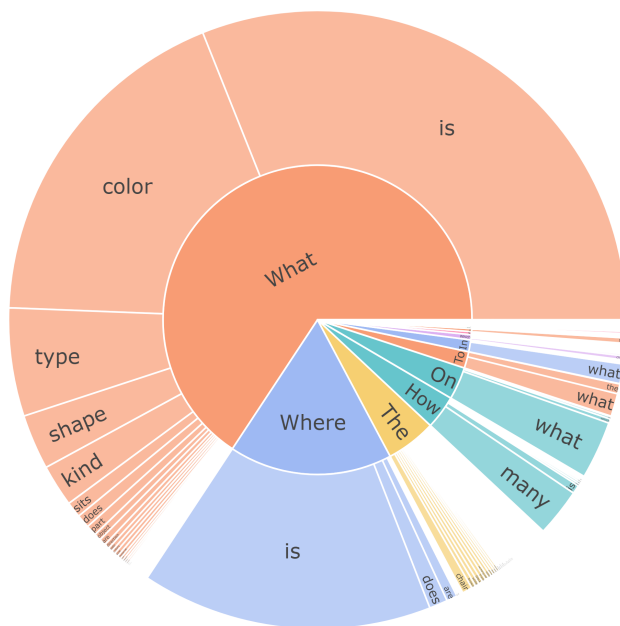


Figure 2.4: Most common question types by wording

**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 XX 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 XX, 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 2.1: 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 XX 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".

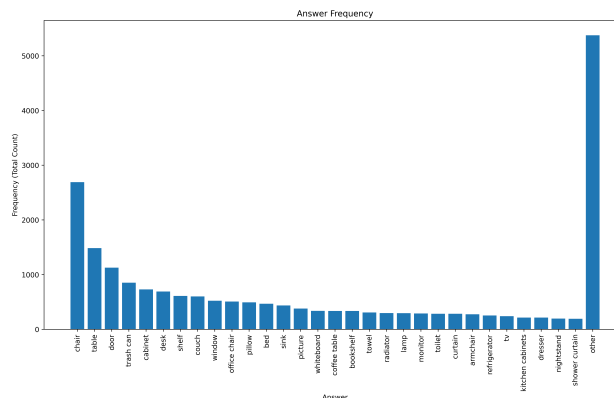


Figure 2.5: Answer distribution of 30 most common answers. The "other" label groups all other less frequent labels. **add other(x) with x number of labels contained here, remove title and x-axis title. also make this figure smaller and cleaner and match color scheme to plot above. IMPORTANT: make it vertical instead of horizontal, more pleasant to read in this document**

# Chapter 3

## Method

This project addresses the task of Visual Question Answering within 3D indoor environments (3D-VQA). 3D-VQA consists in answering open-vocabulary, free-form questions about an environment. For example, given segmented RGB-D images of a hallway and the question "Where did I put my keys?", the 3D-VQA model should be able to locate the most relevant object in the scene and output an accurate response to the question, such as "On top of the drawer at position  $(x, y, z)$ ".

The two main challenges of the 3D-VQA task are 1) to create a rich scene representation, often starting from RGB-D frames and 2) to combine scene representations and text queries in a 3D-VQA model producing an accurate output answer. The 3D-VQA task hence requires generating and merging different data modalities in a multimodal model, which conditions output answers on the given question-scene input pairs.

While current research often claims to utilize scene graphs as scene representations, it does not utilize edge information in the 3D-VQA model, and solely relies on node information. This means that current methods do not leverage the full potential of scene graphs, and do not bake any relational or structural information of the scene into its representation. Furthermore, existing 3D-VQA models often rely solely on pre-trained language-vision models, such as CLIP, to bring the question and scene representation into the same embedding space, lacking a learning mechanism.

To mitigate these limitations, we propose to develop:

1. A dataset of scene graphs for the ScanNet dataset (ScanSG), paired with questions from the ScanQA dataset, regarding the graphs. This is, to our knowledge, the first dataset of scene graph-question pairs for 3D indoor environments.
2. An end-to-end 3D-VQA classification model (ScanSG-GNN), which combines graph and text modalities to answer questions about the scene.

It should be noted that the proposed VQA model is a classification model, rather than a generation model. Hence, the model does not aim to output free-form answers, but rather solves the task of locating the most relevant object to the question posed. For example, the question "What color is the chair by the desk" should return the relevant instance of "chair", rather than the answer "the chair is black". In other words, the proposed model returns the object which answers the question "Where should one look to answer the question?", rather than a free-form answer to the question.

This classification task is easier to evaluate and interpret than the generation task, and hence a good starting point for evaluating the model's ability to understand question-scene pairs. Furthermore, successfully solving the classification task ensures that the proposed VQA model relies on structural scene understanding rather than textual priors, since a classification model must focus on instance accuracy rather than semantic accuracy. In other words, solving the classification task rather than the generation task forces the model to

learn to understand the scene and the question, rather than to learn common sense. Our model can be easily extended to a generation model by adding a decoder component to the model output.

The following sections detail the scene graph generation method used to obtain the ScanSG dataset, as well as the proposed 3D-VQA classification models.

+ in ScanQA paper, it is shown that text generation works MUCH better when it is trained alongside classification. This shows that the classification task is actually super important and the better we can perfect it/understand it, the more likely we are to develop a good generation model using that.

+ what makes this task hard is that its basically zero-shot classification!

### 3.1 Scene Graph Generation

[Add figure here w each step summarized, + examples of output scene graph (can also add to appendix)?]

We aim produce semantically meaningful graph representations of the ScanNet indoor scenes. The scene representation should simplify the point-cloud representation into a scene graph, while preserving 3D structural information. The 3D structural information differentiates scenes with the same objects placed differently, and hence enables reasoning over the space.

To achieve this, we let graph nodes represent segmented objects in the scene, and graph edges represent proximity, such that nearby nodes are connected by an edge. We note that there is no such "ground truth" representation, since scene graph generation requires some design choices. 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.
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".

The following sections detail the heuristics-based method used in this paper to generate ScanSG. We first describe the node feature generation, then the adjacency generation and finally the edge feature generation. Together, these components form the ScanSG dataset.

#### 3.1.1 Node features

While existing SGG methods commonly segment the scene as a first step (for an end-to-end method), we opt to use oracle segmentation provided by ScanNet to get a more accurate SG dataset.

We initialize the node embeddings with CLIP-embedded images of the object of interest. However, as each object may appear in many RGB-D frames under different views, we must select which views to embed (**View Selection**), and how to combine these into a single node embedding (**Cropping + Embedding**).

+ CLIP embeds the appearance and semantic meaning of the object (color, shape, function, ...). This is great because if we provide a textual description of the object, then its CLIP embedding should be quite close to the image. (CLIP was trained on image-description pairs).

**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. 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.

[ multiplying vs. adding? if adding, We note that weights could be added to both the "size" and "visibility" components of the score to weigh their relative importance.]

**Cropping** For each object, we then crop the top- $k$  views around the object of interest. In the ScanSG dataset, a square crop centred around the object, and resized to  $224 \times 224$  is used. This is because CLIP was trained on this input size. Other cropping methods are explored and compared in section XX.

**Embedding** Pre-trained CLIP was chosen to embed images as it brings images and text into the same embedding space, allowing for comparison between the two. For each object, the top- $k$  crops are then CLIP-embedded (using the Hugging Face implementation [XX]) 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.

### 3.1.2 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.

+ The threshold method is justified because from scene XX, the object density is pretty consistent across scenes - so a single threshold value should work for all scenes.

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 estimate the shortest distance between two objects, 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. This can be calculated as:

$$d = \dots$$

A threshold of 0.2 was selected heuristically. The adjacency matrix was symmetrized to ensure bidirectional relationships between nodes. Additionally, self-loops were added.

### 3.1.3 Edge features

We compare two learning-based approaches to annotate/predict the edge label of the edges determined above.

We hand-label 275 edges from the following prepositions list: on/above, under inside, contains next to attached to, supporting

However this dataset is strongly imbalanced (mostly next to).

**Random Forest**

**MLP**

## 3.2 VQA Models

[for models, add the number of total parameters] [also add details about the GNN you use (GCN with 512 in, 512 out)].

Figure XX illustrates the general 3D-VQA classification method. VQA models take as inputs a question and a scene graph (generated as described above), and outputs the most relevant object in the scene to the question. Regardless of architecture, any VQA model must have the following components: 1) a scene encoder, 2) a question encoder, 3) object selection head, which combine scene and question encodings to select the output object.

We explore solutions for each of these components, and compare the performance of three distinct families of model architectures: a baseline model (no learning), a GNN-only (learning only for scene encoding) model, and a transformer-based model (learning for scene encoding, question encoding and selection head). Some of the components are shared across models, this helps us identify which components of the final proposed model are actually helpful.

The following sections detail the architecture and design choices of each model.

We incrementally build up the proposed model the baseline, adding complexity/degrees of freedom to the model at each step.

### 3.2.1 Baseline

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

**Scene Encoder:** The scenes are encoded as the set of its ScanNet-SG scene graph nodes (without edges, obtained as described in section XXX).

**Question Encoder:** The question encoding is the sentence-level CLIP embedding of the question.

**Object Selection Head:** The model computes the cosine similarity between the question embedding and the node embeddings of the scene graph. The node with the highest similarity is selected as the output object.

While this baseline model is a simple and common approach to the VQA task [ref], it has some important limitations. Firstly, the scene encoder incurs a large information loss: by discarding edges, it does not encode any 3D or structural information about the scene. It therefore does not capture the overall room layout, and lacks contextual awareness. This implies that objects with similar appearance may have indistinguishable encodings, regardless of their context and neighbouring objects, and that the overall scene encoding does not change if even if objects are moved around. For example, the baseline model may not be able to differentiate

between a chair in the middle of the room and a chair in the corner of the room, as the scene representation would be the same. This means that while semantic accuracy might be high, instance accuracy is expected to be low. Furthermore, the model has no learning mechanism, meaning that the scene representation cannot adapt to new questions.

### 3.2.2 Baseline + GNN

To address these limitations, we explore the Baseline + GNN model, which uses a graph neural network to learn better node embeddings. The model is illustrated in Figure XXX. The model consists of the following components:

**Scene Encoder:** The ScanNet-SG scene graphs are passed through a dimension-preserving GNN. The GNN updates node embeddings iteratively by aggregating information from neighbouring nodes. This ensures each node becomes aware of its context.

- What type of GNN architecture? How many layers? How many hidden units? What activation function? What aggregation function? What loss function?

**Question Encoder:** Same as the baseline model.

**Object Selection Head:** Same as the baseline model.

The GNN-only model is expected to perform better than the baseline model, as structural information is added to the scene encoding. This model should therefore better differentiate between objects with similar appearance but different context. Furthermore, the scene encoder is learned, meaning that the model can learn to adapt scene encodings to better match the question encoding. In other words, the model can learn to align the question and scene embeddings in a learned way, rather than a fixed way as in the baseline model. However, the GNN-only model still has some limitations. Firstly, the model is still limited by the question encoding and object selection, which are not learned. This means that the model may struggle with questions that are not well-aligned with the learned scene graph.

### 3.2.3 ScanSG-GNN

The transformer-based model incorporates learning for the question encoding and object selection head. It is illustrated in Figure XXX. The idea was to create a model with more degrees of freedom than the GNN-only model: instead of trying to align the fixed question embeddings and learned scene embeddings using cosine similarity, the model learns to predict a score for each object instance in the scene based on the learned scene and question embeddings. High score indicates high relevance to the question.

The model is composed of four main components: 1) a question encoder, 2) a scene graph encoder, 3) a cross-attention mechanism, and 4) a classifier.

The question encoder is a transformer-based model which encodes the question into a fixed-size vector. The scene graph encoder is a graph neural network which encodes the scene graph into a fixed-size vector. The classifier is a multi-layer perceptron which takes the question and scene graph embeddings as input and outputs the most relevant object in the scene to the question.

How many layers? How many hidden units? What activation function? What aggregation function? What edge update function? What loss function?

### 3.2.4 ScanSG-Transformer

### 3.2.5 Batching and Masking

### 3.2.6 Loss function

Cross-entropy loss Multi-object Cross-entropy loss Focal loss

## Chapter 4

# Experiments

[We prepare x different question datasets to experiment on: 1) easy 2) hard 3) complete/mixed 4) constant scenes]

In this section, we detail experimental methods and results for the generation of the ScanSG dataset and the proposed ScanSG-Transformer model.

### 4.1 Scene Graph Generation

[Note: we remove scenes with segmentation errors (some segmented objects do not appear in any of the images – specifically scenes 180, 279, 305, 530, 597)]

In order to train a successful 3D-VQA model, we must ensure that its input data, namely generated 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 testing in section XXX.

#### 4.1.1 Node Construction

We evaluate the impact of different cropping methods, k-values and visibility scores on the quality of the generated node CLIP-embeddings. The following paragraphs detail the methods and parameters considered for each of these design choices.

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

1. 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 [XXX]: the crop is resized to 224 in its minimum dimension, and then randomly cropped to a  $224 \times 224$  square.
2. 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 [XXX].

3. A square crop centred around the object, resized to  $224 \times 224$ . 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.
4. A tight, rectangular crop, resized to  $224 \times 224$ . 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.

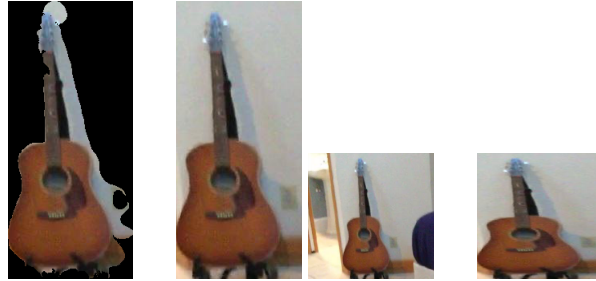


Figure 4.1: Caption 1      Figure 4.2: Caption 2      Figure 4.3: Caption 3      Figure 4.4: Caption 4

Figure 4.5: Overall figure caption

**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 different embeddings. We note that increasing the k-value highly increases the computational cost. It is therefore important to find the optimal k-value that balances these trade-offs. In this analysis we consider the following k-values:  $k = 1$ ,  $k = 3$ , and  $k = 10$ .

**Visibility score:**

1.  $s_1 = \frac{n}{w \times h} \times \frac{b}{8}$
2.  $s_2 = w_p \frac{n}{w \times h} + w_c \frac{b}{8}$  with  $w_p = 1$ ,  $w_c = 1$
3.  $s_2$  with  $w_p = 100$ ,  $w_c = 1$
4.  $s_2$  with  $w_p = 1$ ,  $w_c = 100$

To evaluate the comparative quality of these cropping methods and k-values, we perform the following experiment: first, we generate scene graphs using each of the three cropping methods, with  $k = 1$ ,  $k = 3$  and  $k = 10$  respectively. We therefore have a total of 9 different scene graph node generation methods. Then, we CLIP-embed all the object labels in the scene, and calculate the similarity between the embeddings of the object crops and the embeddings of the object labels. For each object, we select the label with the highest similarity, and compare it to the ground truth label. We calculate the accuracy of the scene graph nodes for each of the 9 methods, and choose the best performing method for the final model. Note that we only use semantic accuracy (since there is no context-awareness mechanism or learning in this experiment) to

evaluate the quality of the scene graph nodes. The semantic accuracy is the percentage of nodes that are embedded closest to their label in the CLIP embedding space. We repeat this experiment for each visibility score listed above, keeping the cropping method and k-value constant. Tables XX shows the results of this experiment.

Table 4.1:

Cropping method	k-value		
	1	3	10
Tight masked crop	49.5	50.1	52.1
Tight crop	57.6	58.9	60.3
Square crop	57.1	58.3	<b>62.2</b>
Tight resized crop	56.2	57.4	60.2

Table 4.2:

Visibility Score	Accuracy
s1	62.2
s2	78.2
s3	
s4	

## Results and Interpretation

Table XX shows some examples of inaccurate node embeddings. The table shows the crop of the object, the ground truth label, and the label with the highest similarity to the object crop. These inaccuracies are most often caused by other objects appearing in the object of interest’s crop for the following reasons:

- they occlude the object of interest (see floor example)
- they are contained within the object of interest (see doorframe example)
- they are similar to the object of interest (see shelf example)
- the object of interest is rectangular so a square crop includes other objects (see guitar case example)

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.

showcases some of the important clip limitations. we further discuss these limitations in Section XX (Discussion).

The experiments also highlight the necessity of including some learning into the model, because the CLIP-embedding of the object is not very close to the CLIP-embedding of the object label. And this label task is easier than the question task, as shown in Figure XXX.

### 4.1.2 Edge Construction

Same model, same node embeddings (only change is the edges).

Dense graphs perform better than sparse graphs. Cannot use no edges because then the scene structure is lost. So these edges help the model learn.

Table 4.3: add the GT and pred label under each image, show them all side by side.

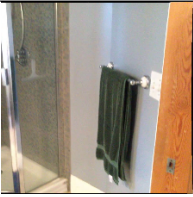
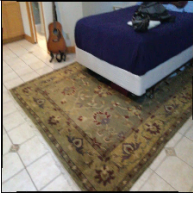
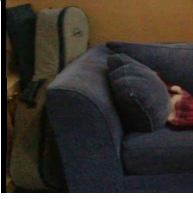
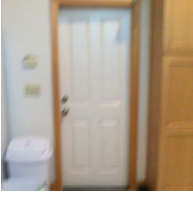
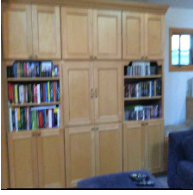
Crop	Ground Truth Label	Most Similar Label
	Wall	Shower
	Floor	Bed
	Guitar Case	Couch
	Doorframe	Door
	Shelf	Cabinet

Table 4.4: accuracy (percent)

Threshold value (m)	EM@1 (%)
0.2	<b>20.1</b>
0.5	14.6
0.8	9.8
Random edges	4.0

We train GNN model on the scene graphs with different threshold values for the edges. We evaluate the performance of the model on the test set, and choose the best threshold value for the final sg dataset. We also compare the performance of the model with the best threshold value to the performance of the model with random edges. Table XXX shows the results of this experiment.

Simple GNN model - $\zeta$  show that chosen threshold is better than random edges. Also show that edges can have a huge impact on the final answer - $\zeta$  must be chosen carefully.

### 4.1.3 Edge Label Classification

### 4.1.4 Dataset Statistics

(Add to appendix instead) Average number of edges per graph, "impact window"? ...

## 4.2 Visual Question Answering Model

In this section, we assess the performance of each model described in section XXX on the generated ScanSG dataset. The evaluation metrics and hyperparameters common to all experiments are described below. The details specific to each experiment are described in the relevant sections, such that the results can be reproduced.

Our model introduces 2 new components compared to the traditional baseline: edge information + cross-attention. This section investigates the added benefit of each component in the context of the ScanSG + ScanQA dataset.

Turns out cross attention definitely contributes to higher performance (the ScanSG-noGNN always outperforms the baseline). The GNN does not actually contribute, but this is most likely because it is not suited to this dataset. this dataset contains very little data which can actually take advantage of the gnn structure. In conclusion, this is the best we can do with the data we have. Need better Q dataset (maybe one that lists all possible answers!!), and maybe need to think of other ways to embed the nodes (such as owl?).

### 4.2.1 GraphVQA

what we tried, the results were random on the scenes tested. the main issue was that this method relies on the "programs" but we had none for this method. so this is also a sign that the model was overfit to their specific dataset.

### 4.2.2 Experimental Setup

**Dataset** The test/train split from the original ScanQA paper [XXX] is used. Multi-answer questions are removed such that each question has exactly one answer. Unless otherwise stated, the training set consists 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 unseen 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.

**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), EM@5 and EM@10 achieved. EM@K is the percentage of questions for which the correct answer is in the top-K predicted answers. We report three 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@10 score indicates that the model is learning to predict the correct answer with higher confidence, while an increasing EM@1 score indicates that the 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 XXX summarizes the results achieved by each model.

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

### 4.2.3 Model Performance on ScanSG + ScanQA

Table XX shows the performance of each model on the generated ScanSG+ScanQA dataset.

Table 4.5: Performance on the full ScanSG+ScanQA dataset, with node embeddings computed from the top-k image crops, as described in XX.

Model	EM@1	EM@5	EM@10	Sem-EM@1
Baseline	28.1	59.7	73.8	36.4
Baseline + GNN	12.3	38.8	54.3	18.9
ScanSG-GNN	25.3	64.2	81.2	33.5
ScanSG-noGNN	<b>43.6</b>	<b>73.9</b>	<b>86.6</b>	<b>55.7</b>

**Baseline** Since the baseline model does not require any training, it is simply evaluated on the test set of the ScanSG dataset. The hyperparameters and loss function are irrelevant.

The baseline model does not utilize edge information (which encodes scene structure and 3D information), and thus does not have the ability to differentiate between instances of the same object class. This results in a higher semantic accuracy (Sem-EM@1) than instance accuracy (EM@1), as seen in Table XX.

Furthermore, since the baseline relies on static, pre-computed text and image embeddings, it 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 the image, it is expected that question embeddings may not align with their associated answer.

#### **Baseline + GNN** Learning rate for Baseline + GNN:

The Baseline+GNN model adds a learning component to the Baseline model, in the form of a GCN which propagates node information to its neighbours, such that each node gains an awareness of its surroundings. This makes for a context-aware scene representation.

However, Table XX shows that this method performs worse than the simple baseline. 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 embeddings such that they better align with the questions they answer. However, as some questions have the same answer but very different embeddings, this creates conflicting training examples. Noisy data impedes model learning.

+ maybe add a figure to illustrate this

+ also maybe not enough data that actually needs a gnn. pull these two conclusions together in the final conclusion

**ScanSG-GNN** The ScanSG-GNN model adds learnable, word-level text embeddings, and a cross-attention mechanism. The performance of this model is similar to that of the baseline model, but with higher confidence (higher EM@5 and EM@10 scores than the baseline).

**ScanSG-noGNN** This model is the ScanSG-GNN model stripped of the GNN. The difference between ScanSG-noGNN and the baseline is that there is word-level embedding and a cross-attention mechanism.

This model outperforms the ScanSG-GNN model. However, since it has been stripped of the GNN, it has no context awareness (two scenes with the same objects in different positions would have the same encoding), and should not be able to answer relational questions. This means that the model is not learning the scene structure but rather language priors.

This model performs the best, indicating that the cross-attention mechanism with the word-level embeddings work better than the cosine similarity of embeddings. This also indicates that the GNN does not benefit model performance.

We note that all models display strong overfitting. [add proof in appendix](#). In the following section, we address the following questions, which arise from the experiment: 1) Can we regularize the ScanSG model to prevent overfitting? 2) Can the overfitting be explained by poor data quality? and 3) Why does the GNN component decrease accuracy?

[likely because the gnn is data hungry and there is not that much data that requires information propagation. the hard questions do but also they are likely too hard \(even for human, they have many possible answers\).](#)

#### 4.2.4 Addressing Overfitting

The discrepancy between the train and test accuracy (and loss) suggests that the models suffer from overfitting. This is further illustrated in Figures XXX. While the train accuracy increases with increasing number of training epochs, the test accuracy initially increases, and starts to decrease after around 6 epochs. In Figure XXX, the train loss decreases with increasing number of epochs, whereas the test loss initially decreases, and then starts to increase around 6 epochs. This indicates that the model is overfitting to the training data, limiting its generalization to the test data.

#### Model Regularisation

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):
- Dropout:
- Increased batch size:
- Decreased LR:
- Normalization:
- Focal loss:
- Decreased model size: Number of parameters: Describe regular model (ScanSG-GNN) vs XS model (ScanSG-GNN-XS) vs. downsized model (ScanSG-GNN-32)
- Architectural changes: Removing some of the building blocks: Without GNN, without textencoder, with scene transformer encoder)

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 XXX.

Table 4.6: redo these experiments if possible

	Test Accuracy
No changes	20.1
Batch size = 512	17.9
Weight decay = 1e-3	20.6
LR = 1e-5	17.5
Normalization	12.01
Focal Loss	19.0
Dropout	
XS model	19.2

We note that none of the regularization methods described above improved generalization. We therefore investigate the impact of data quality on the model’s ability to generalize.

#### 4.2.5 Model Performance on Ideal Dataset

We investigate the impact of the question dataset on model performance. For these experiments, we split the ScanQA dataset into the following smaller datasets:

- Easy dataset: contains only questions which contain the answer. Train set: 11806 questions about 555 scenes. Test set: 1083 questions about 71 scenes.
- Hard 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 dataset: contains all questions, same dataset as used in previous experiments.

#### Ideal Graph Nodes, Full Question Dataset

To understand the impact of the node embeddings on the model performance, we embed the object labels into graph nodes, rather than object images. The only difference from the previous experiment are the node embeddings. Table XX shows the results of this experiment.

Table 4.7: Performance on the full ScanSG+ScanQA dataset, with node embeddings computed from the object labels

Model	EM@1	EM@5	EM@10	Sem-EM@1
Baseline	50.7	74.4	82.8	62.7
Baseline + GNN	16.2	45.3	62.1	24.7
ScanSG-GNN	33.1	73.1	88.0	43.6
ScanSG-noGNN	<b>59.0</b>	<b>84.0</b>	<b>92.2</b>	<b>72.4</b>

We note that results are significantly higher across all models, and the overfitting is slightly mitigated **add proof**. This shows that poor node embeddings contribute to overfitting, and low performance on the test set. The limitations of CLIP causing this were illustrated in Figure XX, and are further discussed in Section XX. These results highlight the importance of high quality node embeddings.

The baseline outperforms the ScanSG-GNN model, and the ScanSG-noGNN is once more the best performing model. To understand why this may be the case, we conduct the following experiments.

### Ideal Graph Nodes, Easy Question Dataset

Table 4.8: **Easy Q dataset Add semantic accuracy**

Model	EM@1	EM@5	EM@10	Sem-EM@1
Baseline	67.3	91.7	96.6	83.3
Baseline + GNN	19.8	47.8	63.3	31.5
ScanSG-GNN	40.9	82.5	91.7	56.6
ScanSG-noGNN	<b>76.0</b>	<b>93.6</b>	<b>97.1</b>	<b>95.0</b>

results significantly higher and overfitting mitigated (show this somehow). this method is likely better because we embed the exact label that appear in the question, so if the cross-attention mechanism works well, it just needs to select the word in the question which is also a node label.

add attention pattern for not the best model here!! and show that the model just focuses on the matching word.

+ and conclude that this model is good! (76) for easy input data!

### Ideal Graph Nodes, Hard Question Dataset

Table 4.9: **Easy Q dataset Add semantic accuracy**

Model	EM@1	EM@5	EM@10	Sem-EM@1
ScanSG-GNN	24.7	57.6	72.3	29.7
ScanSG-noGNN	34.9	62.4	77.9	43.0

If easy dataset is better with no-GNN because it doesnt need context awareness, maybe hard dataset can actually benefit from gnn module. so we test on hard data only and see if this is still true.

still true - 2 possible explanations for this: - either the questions are just too hard and cant be answered (even i cant answer them tbh, there are too many possibilities) - gnns are quite data hungry [reference] and there is not much "hard" data in this dataset. to prove this, gradually increase datasize and show that adding more data slightly improves the proplem? maybe even do a projection of how many datapoints you would need?

Also note that the gap between gnn and no-gnn is much smaller than in previous experiments.

todo: would be useful to take a look at which questions the gnn vs the no-gnn gets wrong.



## Chapter 5

# Discussion

The discussion section gives an interpretation of what you have done [1]:

- *What do your results mean?* Here you discuss, but you do not recapitulate results. Describe principles, relationships and generalizations shown. Also, mention inconsistencies or exceptions you found.
- *How do your results relate to other's work?* Show how your work agrees or disagrees with other's work. You can rely on the information you presented in the “related work” section.
- *What are the implications and applications of your work?* State how your methods may be applied and what implications might be.

Make sure that the introduction/related work and the discussion section act as a pair, i.e. “be sure the discussion section answers what the introduction section asked” [1].

Table 5.1: **compare with original scanqa paper**

Model	EM@1	BLEU-1
ScanQA [...]	23.45	31.56
ScanQA2 [...]	23.92	32.72

**Are other methods doing this actually generalizing? Or are they just memorizing the dataset?**

I don't really think our model is worse off than the others, i think all models suffer from the overfitting problem.

**Dataset size:** The dataset is relatively small, which might have an impact on the performance of the model - might not be able to generalize well to unseen data, overfitting to the training. For reference/comparison, the VQAv2 dataset has 1.1M questions, and the GQA dataset has 22M questions (2D VQA datasets). Yet we only have 20K

**Dataset quality:** Dataset contents: most quesitons can be answered with 2d information only...

**Limitations of CLIP** CLIP limitations (eg. high dimensional data, etc.)

**Requirements for a better dataset**

A lot of the other papers dont prove that the method is generalisable – they do not report scores on other datasets and do not report the difference in train/test accuracy. the only papers that do this are simple clip method.

**Future work** could consider another way to encode the scene – maybe add some kind of context nodes? difficult to actually assess how integrating edge/context information is helpful for the VQA problem because we dont have a high quality question dataset. this should be a priori

## **Chapter 6**

### **Conclusion**

List the conclusions of your work and give evidence for these. Often, the discussion and the conclusion sections are fused.

## Appendix A

# The First Appendix

In the appendix, list the following material:

- Data (evaluation tables, graphs etc.)
- Program code
- Further material

**add scansg dataset statistics?**

# Bibliography

- [1] R.A. Day and B. Gastel. *How to Write and Publish a Scientific Paper*. Cambridge University Press, 2006.