Referring Expression Comprehension for CLEVR-Ref+ Dataset

by

Kuldeep Singh Rathor

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2020 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Yezhou Yang
Michael Simeone

ARIZONA STATE UNIVERSITY

August 2020

ABSTRACT

Referring Expression Comprehension (REC) is an important area of research in Natural Language Processing (NLP) and vision domain. It involves locating an object in an image described by a natural language referring expression. This task requires information from both Natural Language and Vision aspect. The task is compositional in nature as it requires visual reasoning as underlying process along with relationships among the objects in the image. Recent works based on modular networks have displayed to be an effective framework for performing visual reasoning task.

Although this approach is effective, it has been established that the current benchmark datasets for referring expression comprehension suffer from bias. Recent work on CLEVR-Ref+ dataset deals with bias issues by constructing a synthetic dataset and provides an approach for the aforementioned task which performed better than the previous state-of-the-art models as well as showing the reasoning process. This work aims to improve the performance on CLEVR-Ref+ dataset and achieve comparable interpretability. In this work, the neural module network approach with the attention map technique is employed. The neural module network is composed of the primitive operation modules which are specific to their functions and the output is generated using a separate segmentation module. From empirical results, it is clear that this approach is performing significantly better than the current State-of-the-art in one aspect (Predicted programs) and achieving comparable results for another aspect (Ground truth programs).

DEDICATION

First and foremost, I dedicate this work to my mummy and papa without whose support and love I won't be writing this. I also dedicate this to my younger brother without whom I'd have worried about my parents a lot. And last but not the least, all of my close friends, peers, mentors and all the teachers without whose contribution to my growth, it'd not have been fruitified.

ACKNOWLEDGMENTS

Throughout my work and my Masters, I received a lot of support both personal and professional and I'd really like to thank all of those involved. First and foremost, I'd like to thank my advisor and mentor, Dr. Chitta Baral for his immense support and guidance throughout my Master's degree here at Arizona State University. This work has only been possible because of knowledge which I have gained undertaking his course and mentorship. His constant encouragement and comments have been invaluable in the continuous improvement of my work.

I would also like to thank my committee members Dr. Yezhou Yang and Dr. Michael Simeone, who offered guidance and support whenever I needed. I'd also like to thank Dr. Heni Ben Amor and Simon Stepputtis for their support and for providing resources required for this work. I'd also like to acknowledge my lab peers for being there for me. To be specific, I'd thank to Tejas Gokhale and Pratyay Banerjee for their crucial inputs at the time when I really needed for this work. I can't move forward without thanking Sanjay, Ashok, Samarth, Shailaja, Ishan, Aurgho, Vaishnavi, Kuntal, Arindam, Arpit for their friendship, discussions and support.

I'd also like to thank my friends/roommates at ASU with whom this time was really a fun and learning experience.

Last but not the least, I would like to thank my mummy, papa and younger brother Bhupendra for encouraging me and standing with me through my struggling times. Thank you for everything.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

In an environment, referring expressions(Winograd, 1972)are the natural language entities used to unambiguously identify or locate the particular objects in a given scene or image. In recent research, the objects are identified in the form of either a segmentation mask or a bounding box. Normally, when humans refer to a particular object in a natural environment, they provide information about that object in the statement they say. This statement can be considered as Referring Expression.

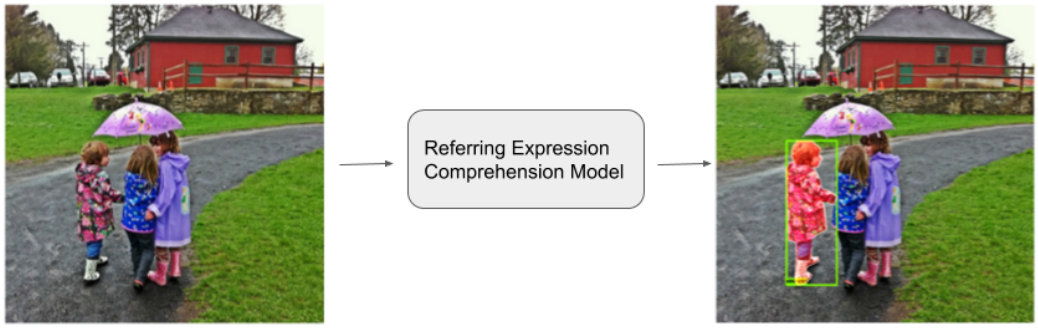

Figure 1.1: An Example of Referring Expression Comprehension

In Figure 1.1, There is an image and a referring expression stating "*Girl in flowery raincoat on left*". Feeding these two inputs into Referring Expression Comprehension model, it gives the image with a specified object in a bounding box. In this case, it's the girl on left. Here it is evident that the task of referring expression comprehension is a problem featuring both Natural Language and Visual aspects.

In this work, the experiments are performed on CLEVR-REF+(Liu *et al.*, 2019) dataset which is a synthetic dataset specially for dealing with dataset biases. Below is an Example of that. In Figure 1.2, there is an image and a referring expression

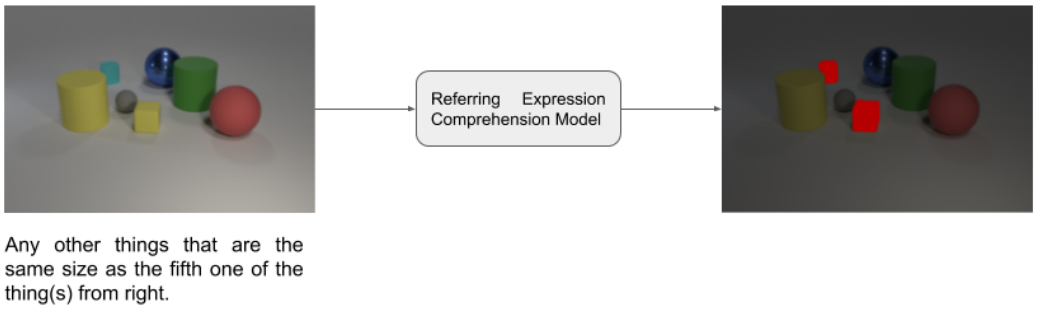

Figure 1.2: An Example of Referring Expression Comprehension for CLEVR-REF+ Dataset

stating "Any other things that are the same size as the fifth one of the thing(s) from right." When these inputs are fed into the model, it produces an segmented image as an output. In this output, it is clearly visible that the objects have segmented mask as identified by the referring expression.

1.1 Motivation

Referring expression comprehension task has ramifications which have relevance throughout the field of Vision-Language as well as Human-Robot Interaction(Goodrich and Schultz, 2007). Identifying an object with the help of a referring expression is a critical application in robotics and vision field.

Recent developments (Yu *et al.*, 2016; Mao *et al.*, 2016; Kazemzadeh *et al.*, 2014) for this task have taken RefCOCO, RefCOCO+ and RefCOCOg datasets into consideration and performed well. The advantage of using these datasets is that they reflect the complexity of real-world to an extent. However, they also have limitations. For example, these datasets demonstrate strong biases that were evidently exploited by the models(Cirik *et al.*, 2018). To deal with these kinds of issues, a synthetic diagnostic dataset CLEVR-Ref+ was constructed for referring expression and a neural module network based model(Liu *et al.*, 2019) was proposed. This approach dealt

with the problem of biases and performed better than the existing models. However, this model suffered at the performance front as it fails to identify for some specific type of referring expressions. This work is intended to address these problems with specific types of referring expressions as well as improve on the evaluation metric for this task.

1.2 Contributions

- Redesigned the neural module network's primitives modules with the addition of a segmentation module for the referring expression comprehension task.
- Experiments on CLEVR-Ref+ dataset with some modifications in the model
- Improve on the evaluation metric so that the overall system can perform better at recognizing the object as directed by referring expression for a given image
- Ablation study for the model to verify the effectiveness of the model.
- Analysis of model performance based on various features of the referring expression.

Chapter 2

BACKGROUND

Language has played a crucial and vital role in the evolution of the civilization as humans see it right now. Not only question answering but also the usage of reference in human communication helps people to exchange information about an object or any other specific entity. Thus referring expressions have facilitated the linguistic evolution. In the words of philosopher John Searle (Searle, 1969):

"*Any expression which serves to identify any thing, process, event, action, or any other kind of individual or particular I shall call a referring expression. Referring expressions point to particular things; they answer the questions Who?, What?, Which?*"

The work in this thesis is built upon the problem associated with a subset of Referring Expression.

For quite some time, there has been tremendous progress in the computation approach for Natural Language processing and understanding. Some of them have been proven to be groundbreaking which has changed the course of research in NLP. Apart from just the language aspect, the research community has shown immense interest in the joint understanding of vision and natural language. As a result, multiple tasks have been formulated based on the combination of Images and Natural Language entities such as questions or statements. Some of the prominent tasks are as follows:

- **Image Captioning**: This task focuses on generating sentence(s) based on an image as an input.

- **Visual Question Answering:** As the name suggests, this task comprises of an image and a natural language question. Given an Image and the relevant questions, the system finds the correct answer.

- **Referring Expression Generation and Comprehension:**

In this work, the main focus is on the Referring Expression based on a joint understanding of Natural Language and Vision. As mentioned in the previous chapter, Referring Expressions are either simple phrases or statements having a larger context for a respective image. Considering this scenario, There are two possibilities with respect to the referring expressions which are the following:

- **Referring Expression Generation:** This task involves the generation of the natural language entities (i.e. Referring Expression) that identify specific target object present in an image.

- **Referring Expression Comprehension:** As mentioned in the previous chapter, Referring Expression Comprehension task is about localizing or identifying the object in an image described by a given referring expression.

The main focus of this work is on the Referring Expression comprehension task where the goal is to identify the object referred by the expression. Recent developments(Hu *et al.*, 2016b; Mao *et al.*, 2016) regarding Referring Expression tasks have seen the Referring Expression Generation and Comprehension in a mix altogether. Since these tasks are a joint venture between Language and Vision, this work lays out the important basic information which would be the base for this work.

2.1 Deep Learning

Deep learning which is also known as Deep Structured Learning, is the subfield of a broader family of Machine Learning methods based on Artificial Neural Networks(Hassoun *et al.*, 1995) with representation learning(Bengio *et al.*, 2013). These learnings can be supervised, semi-supervised or unsupervised in nature. In supervised learning, the problem has training data and corresponding labels that serve as a teaching signal. Contrarily, in unsupervised learning, there is only training data and the model needs to learn the patterns or regularities from data without any label. Semi-supervised learning falls between supervised and unsupervised learning. In this approach, a small amount of labeled data is combined with a large amount of unlabeled data during training.

Deep learning architectures such as deep neural networks(Bengio, 2009; Schmidhuber, 2015), deep belief networks(Hinton, 2009), recurrent neural networks(Mikolov *et al.*, 2010) and convolutional neural networks(LeCun *et al.*, 1998) have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection, and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

In this thesis, the task lies in the category of supervised learning problem which falls under the joint venture of Natural Language processing and computer vision. First of, the most basic building block of the Deep learning i.e. Neural Network (McCulloch and Pitts, 1943) would be discussed. A neural network is a network composed of many basic elements called neuron or perceptron. A simple look at neuron architecture would be like this in figure 2.1.

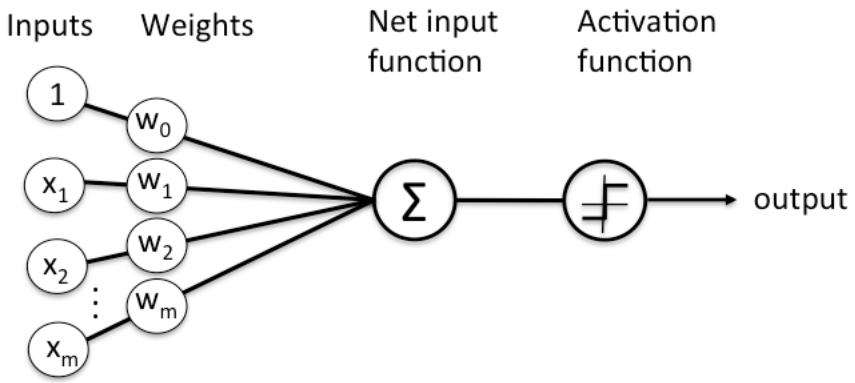

Figure 2.1: A Simple Neuron/Perceptron

This neuron receives an input vector $x = (x_1, x_2, x_3, ...x_m)$, and then multiplies it with weights $w = (w_1, w_2, w_3, ..w_n)$. The result is summed with a variable called bias, which can be represented as b. This summation can be defined as below :

$$z = \sum_{i=1}^{m} x_i * w_i + b$$

Then z is provided as input to a nonlinearity function (also known as activation function), which generates the output as following :

$$a = g(z) = g(\sum_{i=1}^{m} x_i * w_i + b)$$

Most commonly used activation functions are sigmoid function $f(x) = 1/(1+exp(-x)))$ tanh function $f(x) = 1 - exp(-2*x)/(1 + exp(2*x))$ and rectified linear unit(ReLU) $f(x) = max(0, x)$. A neural network (shown in Figure 2.2) is composed of many neurons stacked horizontally(next to each other) and vertically(on top of each other) and is followed by a final output layer. If there are several layers in a network, for a specific layer i, it receives input from previous layer $x_i = a_{i-1}$ then compute the activation a_i and pass it to the next layer. For a binary classification problem, the

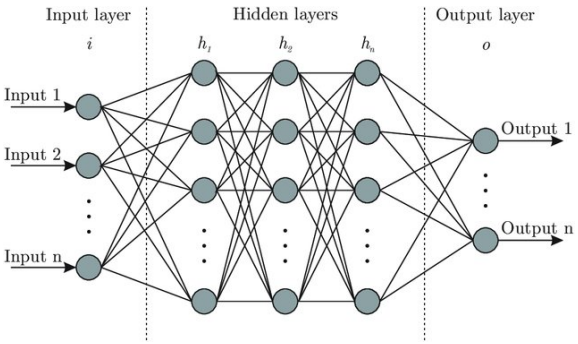

Figure 2.2: A Neural Network

final neuron output a real-valued scalar p, taking value from 0 to 1, which can be interpreted as probability to a specific category, say *class 1*. If p is larger than 0.5, then model output prefers to *class 1* with respect to input, or if p is smaller than 0.5, the model chooses *class 2*.

Now that, the computation flow of a network is already defined, the supervised signal needs to be taught to the neural network so that it can know how to adjust the parameter to give better predictions for training data. Generally, cross entropy loss is used as the loss function for classification problem.

$$L(D;\theta) = \frac{1}{N}\sum_{i=1}^{N} -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$$

where $\hat{y}_i$ is the output of neural network, $\hat{y}_i \in (0,1)$ and y_i is the label for sample x_i, which is either 0 or 1. Since the likelihood of the data D given parameter is

$$P(y_i = 1|x_i;\theta) = \hat{y}_i^{y_i} * (1-\hat{y}_i)^{1-y_i}$$

$$\text{Likelihood}(D;\theta) = \prod_{i=1}^{N} P(y_i = 1|x_i;\theta)$$

The negative log likelihood Likelihood$(D;\theta)$ can be written in $L(D;\theta)$. In order to minimize $L(D;\theta)$, the gradient needs to be calculated first, then parameters should be adjusted along the gradient direction. this process is iterated until the function converge to a locally optimum. This is called the gradient descent algorithm.

$$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \frac{\partial L(D;\theta)}{\partial \theta} | \theta = \theta_{\text{old}}$$

where α is a small positive scalar called the learning rate.

2.1.1 Recurrent Neural Network

The Recurrent Neural Network(RNN) is a variant of the artificial neural network. It has the capability to process the sequential data by taking the output h_{t-1} of itself at previous time step t_1 as input at time step t. In a formal way, It can be represented with the formula $h_t = f_\theta(h_{t-1}, x_t)$, where f is the function depending on the network architecture which is being used and h_t is the current internal state of model. Since its value is computed based on the current input and the previous timestep hidden state output, it encodes the information about the history of data. Each time the recurrent neural network receives a new input x_t from the sequence of data, it updates the internal state h_t using the formula above. θ is represented as the parameters for this model which is used at every time step. For vanilla RNN architecture, the representation would be given by

$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1})$$

and here tanh is a nonlinearity function. Another nonlinearity function such as ReLU can also be used here. RNNs have been pretty useful for sequential data and one of the applications of RNN is the recurrent neural network language model as illustrated in Figure 2.3. A language model is a probabilistic model of a sequence $x_1, x_2, x_3, ..., x_n$.

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{j<i}) \tag{1}$$

The probability $p\left(x_i|x_{j<i}\right))$ is calculated by linearly transforming the current internal state into a probability distribution over all words in the vocabulary.

$$p\left(x_i|x_{j<i}\right) = \text{softmax}\left(W_{ho}h_t\right)$$

The parameters W_{ho}, W_{xh}, W_{hh}h for the recurrent neural network are learned by maximizing the equation 1 using cross entropy loss.

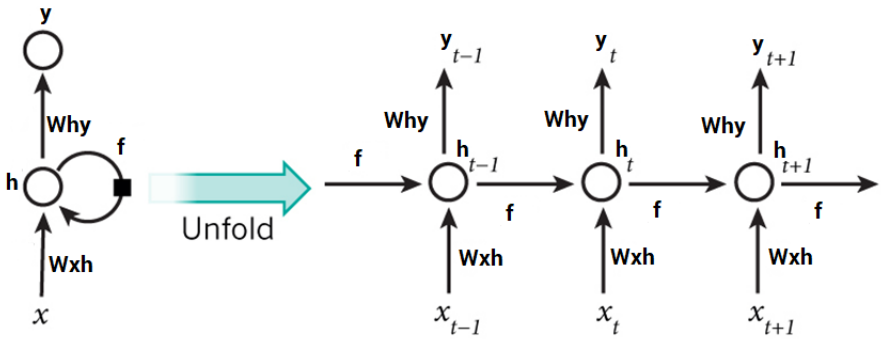

Figure 2.3: A Recurrent Neural Network

However, the vanilla RNN suffers from the problem of exploding gradients and vanishing gradients. It causes them not to retain previous information. To alleviate this shortcoming, Long Short Term Memory(LSTM) were proposed. In LSTM, the input h_{t-1} and x_t are computed in a more complex manner and besides the internal state h_t, LSTM also introduce memory cell c_t to store the long-period information. LSTMs are explicitly designed to avoid the long-term dependency problem which is faced by vanilla RNN. Retaining information for long periods of time is practically their default behavior. The LSTM cell is basically responsible for keeping track of the dependencies between the elements in the input sequence. Each time the LSTM can

decide to overwrite a memory cell, retrieve the content from it, or keep its contents for the next time step by using explicit gating mechanisms (viz. forget gate, input gate, and output gate) . The activation function of the LSTM gates is often the logistic sigmoid function. The values of h_t and c_t are updated as follows:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

In Recent developments, the RNN is used heavily for sequence to sequence problems such as machine translation, caption generation, language modeling, and so on. Since RNN works in a way that it keep previous information, when it is used as a sentence or phrase encoder, it actually encompasses the semantic meaning into real-valued vectors.

2.1.2 Convolutional Neural Network

In fully connected neural network, the node in current layer i is connected to each node in the previous layer $i-1$. However, in convolutional neural network, the node in layer i is only connected to a part of nodes in layer $i-1$. This structure is different from the common linear layer. This kind of neural architecture make the handling of data with spatial topology.(e.g, images, videos, character sequence in text) easier.This

neural architecture is known as convolution layer. It is obviously know that the pixels in image data are only correlated with nearby pixels. This way, a node in the upper layer only need to connect a part of the node in the lower layer and the connection weight can also be shared with all other nodes in the same upper layer.

In Figure 2.4, It can be seen that a node in upper layer(activation map in the figure) is computed from a 5×5-sized subregion in original image. In the convolution process, a fixed-size filter (also known as kernel) is slid along the horizontal and vertical direction. This process generates as many nodes in the upper layer as each node is corresponding to a different position for the filter. Weights used to computed the node in activation map is the same when filter slides in different position. The main function of the filters here is to look for certain local features such as edges, curves and other complex features in the input layer. Because of the local connectivity and parameter sharing property of Convolutional Neural Network, the number of parameters can be reduced in each layer. This allows to stack several layers to build a complex architecture. One of the examples of such kind of complex architecture is ResNet model(He *et al.*, 2015).

Apart from the convolution layer, use of pooling layer is also prevalent in the convolutional network architecture. The pooling layer works as a down-sampling component that can reduce the dimensionality of the input data by a fixed factor. A commonly used pooling layer is 2×2 max pooling layer with stride 2. It means that there would be a 2×2sized filter sliding in the input feature map horizontally and vertically with a step size of 2. For each position of filter, it receives 4 inputs and generateds the maximum out of them and returns that as output. Then the filter slides to the next position with step size of 2. Because of this, the neighbor positions not overlap with each other. So the size of input feature map is reduced by a factor of 4, which however comes at the cost of losing some local spatial information.

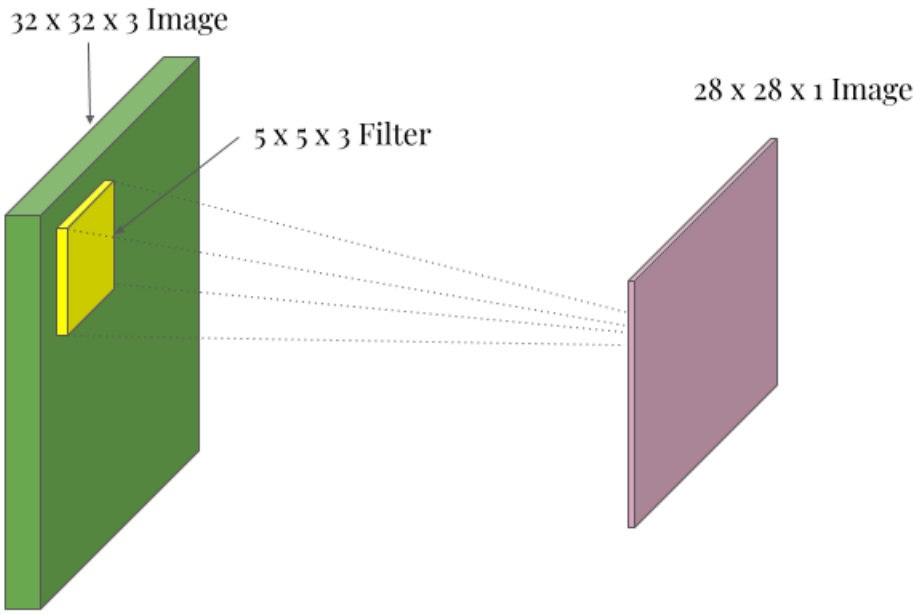

Figure 2.4: Convolution in A Convolutional Neural Network

A convolutional network is generally composed of convolution layer, fully connected layer(the common linear layer), pooling layer and RELU layer. The most common and prevalent use of a ConvNet has been the task of image classification. An image is provided as input to the network and the parameter of convolution layer and fully connected layer are learned in such a way that the network can output the right category of the input.

When the convolution networks trained on large scale image dataset are analyzed, it is observed that the lower layer in convnet captures the local information of input image such as texture, lines and shapes, while the high layer captures more abstract information such as the content of input. Therefore, once trained on large-scale image dataset such as ImageNet(Russakovsky *et al.*, 2015) for image classification task , it

can be said that the network along with the learned parameters are able to extract useful information from a given image. Extracted information from images are in the form of vectors and are called the representation of images. More specifically, the formula would be $v = CNN_\theta(I)$, where

- CNN is a convolutional neural network with pre-trained parameters θ
- I is the input image pixels
- v can be the vector of the layer right before the softmax layer

In the vision community, there have been tremendous developments regarding such architectures. Some of those models' parameters have been made public to use such as AlexNet(Krizhevsky *et al.*, 2012), VGGNet(Simonyan and Zisserman, 2014) and ResNet(He *et al.*, 2015).

In this thesis work, a ResNet model is utilized for feature extraction. This model is trained over Imagenet dataset.

Chapter 3

RELATED AND EXISTING WORK

Referring Expression Comprehension has been an important and crucial problem in Natural Language Processing(NLP). Because of its important application in Human-robot, human-computer interaction, and Language-Vision domain, a considerable amount of research work have been done in recent years. In this chapter, the recent related work and developments for the referring expression comprehension task will be discussed along with related datasets and techniques.

3.1 Referring Expression

Previous sections covered the idea about Referring Expression significantly and in this section, the related work and progress will be explained. Before deep learning models became prevalent, most studies focused on traditional approaches to understand the referring expression which included enumerating attributes (e.g. color, size, material, shape, etc) and predefined relationships explicitly. However, because of this approach, it covered a very small portion of real-world referring expressions and a large number of them went remained undealt. There are two ways using which referring expressions can identify an object in an image.

- **Object Detection:** Lots of work has been done in the object detection direction using a referring expression. In these works, the output is generally one bounding box containing the object which is referred by the referring expression. Work by (Mao *et al.*, 2016) and (Hu *et al.*, 2016b) adapted Image Captioning task by scoring each candidate bounding box proposal for the referring expression using a generative captioning model. (Mao et al., 2016) also

released a large dataset "Google Refexp dataset" for this purpose. The issue with these approaches is that they do not consider the target object's relationship with other objects in the image. However, It becomes difficult to localize an object if there are multiple objects with the same attributes. These models didn't take relations between the object into account which are necessary in case of objects with the same attributes. Work by (Rohrbach *et al.*, 2016) proposed an approach in which the referring expression is encoded using recurrent neural network and then attention to the relevant image region is learned to reconstruct the input phrase. While testing, the correct attention (the grounding) is evaluated. On another line, to deal with the visual context for referring expressions, work by (Nagaraja *et al.*, 2016) and (Yu *et al.*, 2016) focuses on the integration of the context between objects to understand better the referring expression groundings (i.e. localizing of the referred object) and show that modeling context between objects provides better performance than modeling only object properties. In the work of (Nagaraja *et al.*, 2016), input CNN features are obtained from a (region, context-region) pair in which a complete image together with other objects in the image is considered a context region. There is high very probability that this feature vector may implicitly encode relational information between target objects and the other object. However, (Yu *et al.*, 2016), in their approach to encode relation information, also consider object comparison features a part of visual representation apart from the CNN features to give the way for implicit encoding. For encoding the relation information explicitly, (Hu *et al.*, 2017b) proposed a modular deep architecture where they learn to parse the language expression into textual components instead of treating it as a single entity and align these components with image regions end-to-end. (Luo and Shakhnarovich, 2017) used a discriminative

comprehension model to improve the referring expression generation. The work of (Yu *et al.*, 2017) is also an interesting one where both referring expression generation and comprehension tasks are handled jointly and their framework lets interact these task modules with each other resulting in improved performance in both generation and comprehension tasks. (Yu *et al.*, 2018) used a learned parser and proposed a Modular Attention Network where the referring expression is decomposed into three modular components related to subject appearance, location, and relationship to other objects. Their architecture utilized both language-based attention as well as visual attention. It has displayed to focus well on the subjects and their relationships.

- **Object Segmentation:** In this technique of Referring expression comprehension, a segmentation mask is returned by the system once the image and a referring expression are supplied. (Hu *et al.*, 2016a) used an LSTM network to encode the referring expression into a vector representation, and a fully convolutional network to extract a spatial feature map from the image, thus to produce pixel-wise binary segmentation for the target object. The work of (Liu *et al.*, 2017) features a convolutional multimodal LSTM network that allows multimodal interaction between language, image, and the corresponding spatial information at each word of the referring expression. ((Li *et al.*, 2018) build on top of the work of (Liu et al., 2017) and propose Recurrent Refinement Network which is an architectural improvement and provides better quality segmentation. Similarly, (Margffoy-Tuay *et al.*, 2018) present a Dynamic Multimodal Network which consists of various modules (viz. Visual Module, Language Module, Synthesis Module, and Upsample Module). They show that they are able to generate detailed segmentation using this approach.

3.2 Dataset Bias and Diagnostic Datasets

One of the most prevalent and ubiquitous tasks of Artificial Intelligence is to develop systems that can reason and perform relevant actions(such as visual question answering and referring expression comprehension) about the visual information. For this purpose plethora of datasets(such as FM-IQA(Gao *et al.*, 2015), Visual Genome(Krishna *et al.*, 2017), DAQUAR(Malinowski and Fritz, 2014), COCO-QA(Ren *et al.*, 2015a), Visual Madlibs(Yu *et al.*, 2015), Visual7W(Zhu *et al.*, 2016)) have been introduced and multiple models have been presented to tackle the above-mentioned problems. These kinds of tasks require various kinds of abilities for providing the solution. Some of them are perpetual abilities, ability to perform logical inference, or utilizing commonsense world knowledge(Ray *et al.*, 2016). For some time, researchers believed that those learning systems were performing as expected but at a later stage, it was evidently shown (Cirik *et al.*, 2018) that these systems might not actually perform the reasoning, instead, they were leveraging the inherent biases problems in the datasets. For example, a system may correctly answer the question "What covers the ground?" but it may not be because it understands the scene but because there is affinity in the dataset about the questions when it is snow-covered. Also, there are chances that the model may be heavily exploiting the imbalanced distribution in the dataset. It was also shown evidently that dataset biases exist in referring expression datasets. In an ideal situation, the dataset should be unbiased to reflect the performance of the model's true level of understanding. To deal with this issue, a synthetic dataset CLEVR was introduced by (Johnson *et al.*, 2017a) for visual question answering. On top of this, (Liu *et al.*, 2019) constructed

CLEVR-Ref+ dataset by re-purposing CLEVR for the task of referring expression comprehension task to address concerns of biases. While they dealt with dataset biases issues, their work also performed significantly better on referring expression task.

3.3 Foundation Work

3.3.1 Neural Module Network

The dominant paradigm in deep learning is a "one size fits all" approach. To solve any kind of problem, a fixed model architecture is prepared and trained with the hope that it can capture everything about the relationship between the input and output, and learn parameters for that fixed model from labeled training data.

But real-world reasoning doesn't work this way always. It involves a variety of different capabilities, combined and synthesized in new ways for every new challenge one encounters in the real world. It would be great if a model could be built that can dynamically determine how to reason about the problem in front of it depending on its input. Such a network would be able to choose its own structure on the fly. Neural module networks (NMNs)(Andreas *et al.*, 2016b) are such kind of which incorporate this flexible approach to problem-solving while preserving the expressive power that makes deep learning so effective. A neural module network is an ensemble of artificial neural network structure characterized by a series of independent neural networks moderated by some intermediary. Each independent neural network serves as a module and operates on separate inputs to accomplish some subtask of the task the network hopes to perform.

when this approach is used to train the model, a monolithic single deep network is not the end product, but rather a collection of neural "modules", each of which implements a single step of reasoning. When this trained model is used in conjunction with a new problem instance, these modules can be assembled dynamically to produce a new network structure tailored specifically to that problem. This thesis work builds upon the interesting idea of Neural Module Network only.

The concept of Neural Module Network was introduced by *(Andreas et al., 2016)* for Visual Question Answering task. The idea of this compositional model stems from some previous works. The process of selecting a different network graph for each input data is fundamental to both recurrent neural network (where the network grows in the proportion of the length of the input) and recursive neural network (where the network is constructed according to the syntactic structure of the input)(Socher *et al.*, 2013). In both cases, the computation unit (e.g. LSTM(Hochreiter and Schmidhuber, 1997) or GRU(Cho *et al.*, 2014) cell) is applied repeatedly. Another work, Memory Networks(Weston *et al.*, 2014) also inspires the chaining of different modules in sequence. Work by *(Neelakantan et al., 2015)* proposes a procedure for learning to compose programs from a set of function primitives whose behavior is fully specified.

Subsequently, *(Andreas et al., 2016)* propose the method of both assembling the network computation graph on the fly and thus simultaneously allowing the modules to perform heterogeneous computations in their paper "**Deep Compositional Question Answering with Neural Module Networks**". This approach makes use of the compositionality of language and importantly, all modules in an NMN are independent and composable, which allows the computation to be different for

each problem instance. This approach generates a dynamic network computation graph each time for different inputs instead of having a fixed network computation graph. Now the whole process of leveraging Neural Module Network for VQA task as proposed by *(Andreas et al., 2016)* is explained in brief below.

- Decompose questions into their linguistic substructures using Stanford Parser (Klein and Manning, 2003) to answer each question individually.
- Instantiate these structures by constructing an appropriately shaped neural network from an inventory of reusable neural modules (such as classifying colors etc.).
- Execute this constructed network along with Image features for computation.
- these modules are learned jointly, rather than trained in isolation, and specialization to individual tasks (identifying properties, spatial relations, etc.) arises naturally from the training objective.

Now an example for better understanding of the approach is considered below.

In figure 3.1, there is an image and a corresponding question " *What color is his tie?*". The answers to the above question will be computed in the following steps.

- The ***find[tie]*** module first predicts a heatmap corresponding to the location of the tie.
- Next, the ***describe[color]*** module uses this heatmap to produce a weighted average of image features
- These image features are finally used to predict an output label.

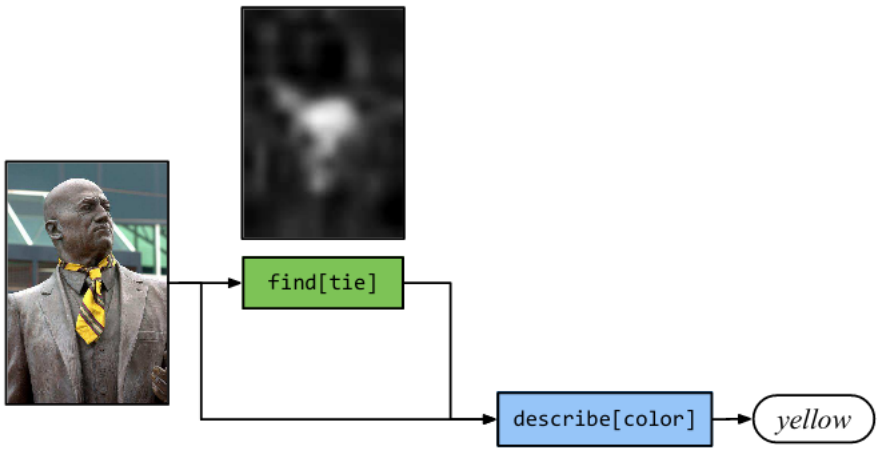

Figure 3.1: NMN for Answering The Question *What color is his tie?* From Work of (Andreas *et al.*, 2016b)

As it can be seen that the question is decomposed into smaller units to deal with specific reasoning step. Based on this work, further improvements were proposed by various works.

The modules in the above work are assembled according to hand-written rules (dependency parser), and thus do not allow generalization over novel inputs. In order to improve upon this relatively rigid NMN approach (Andreas *et al.*, 2016a) extend on their work and proposed "Dynamic Neural Module Networks". In this work, they incorporate a limited form of network layout prediction by learning to rerank a list of three to ten candidates. This set of candidates is generated by rearranging modules predicted by a dependency parser. It's clear that both the above-mentioned models rely on semantic parsing of the questions, which means that the neural modules and their composition must be hand-tuned to correspond with the sentence.

Extending on the work by (Andreas *et al.*, 2016a), an End-to-End module network for visual question answering is proposed by (Hu *et al.*, 2017a). Instead of using Dependency Parser (like in previous one) or solving it as a ranking problem , An optimal layout policy predictor is learned to predict a layout for NMN. This approach learns to optimize over the full space of network layouts rather than acting as a reranker and requires no parser at evaluation time.

Also, in one different approach by (Johnson *et al.*, 2017b), authors built on the NMN approach and modify the natural language component of their network by developing an LSTM based Sequence-to-Sequence model (Program Generator) to predict the Neural Module Network structure. The input of the Program Generator is a question and output is a functional program (i.e. a sequence of functions), thus allowing for more flexibility. They use a generic architecture for all the neural modules. Thus, in an NMN all neural modules share the same generic architecture with different instantiations (or say different parameters).

Further, (Mascharka *et al.*, 2018) in their work reuse the Natural Language component (Program generator) from the work of (Johnson *et al.*, 2017b). However, they change the way the neural modules are used. They apply the use of operation-specific neural modules instead of having a generic architecture for neural modules. These modules are designed for specific tasks keeping the reasoning process in the mind. For example, Relate module works for spatial relations.

3.4 Existing Work For Comparison

The CLEVR-Ref+ dataset built by (Liu et al., 2019) is used in this work and the performance of this work is further compared with the previous work done by them and other relevant work with this approach.

3.4.1 Speaker-Listener-Reinforcer Model (SLR)

SLR(Yu *et al.*, 2017) is a detection model which means that this model gives output in the form of a boundary box that localizes the intended object. In this model, Authors tackle the problem of Referring expression generation and comprehension simultaneously. They present a unified framework which is composed of three modules: speaker, listener, and reinforcer.

- **Speaker**: The speaker generates referring expressions and is based on a CNN-LSTM framework. Here, a pre-trained CNN model is used to define a visual representation for the target object and other visual contexts. Then, a Long-short term memory (LSTM) is used to generate the most likely expression for the given visual representation. Here, the visual representation includes the target object, context, location/size features, and two visual comparison features.

- **Listener**: The listener performs the task of referring expression comprehension. To implement this behavior, they use a joint-embedding model which encodes the visual information from the target object and semantic information from the referring expression into a joint embedding space that embeds vectors that are visually or semantically related closer together in the space. Here for referring expression comprehension task, given a referring expression representation, the listener embeds it into the joint space, then selects the closest object in the embedding space for the predicted target object. They use an LSTM to encode the input referring expression and the same visual representation as the speaker to encode the target object (thus connecting the speaker to the listener).

- **Reinforcer**: Besides using the ground-truth pairs of the target object and referring expression for training the speaker, they also use reinforcement learning(Sutton *et al.*, 1998) to guide the speaker toward generating less ambiguous expressions (expressions that apply to the target object but not to other objects). This reinforcer module is composed of a discriminative reward function and performs a non-differentiable policy gradient update to the speaker.

Since there is no focus on referring expression generation in this work, only the listener part of the model should be considered for comparison. This work's approach is different in the sense that there is no use of joint embedding of visual and natural aspects altogether in a single entity. Both visual and natural language information are processed separately. Also, they use VGGnet for extracting visual features. In this work, the experiments are done with ResNet-101 and ResNet-152 model which are trained on the ImageNet dataset.

3.4.2 MAttNet

MAttNet(Yu *et al.*, 2018) or short for Modular Attention Network model focuses entirely on referring expression comprehension. In this work, they move towards a modular approach from monolithic approaches where expressions are treated as a single unit. They propose to decompose the referring expression into three modular components related to subject appearance, location, and relationship to other objects.

In this model, two types of attention are utilized: language-based attention that learns the module weights as well as the word/phrase attention that each module should focus on; and visual attention that allows the subject and relationship modules to focus on relevant image components. Module weights combine scores from all three

modules dynamically to output an overall score. For language attention, they use a bi-directional LSTM(Dyer *et al.*, 2015) to encode the context for each work. For the visual aspect, they use Faster R-CNN(Ren *et al.*, 2015b) and ResNet-101 for feature extraction.

This work is a little similar to this work in terms of modularity. Both of the work leverage the decomposability of the referring expression and move away from the monolithic approach. However, this work does not use language-based attention here as compared to MAttnet. Also, this work also makes use of ResNet-101 and ResNet-152 for the purpose of image features. This work uses visual attention maps in this work which kind of help in understanding the visual reasoning process as well. Program generator which handles language aspect, is a Seq2Seq model based on single-directional LSTM.

3.4.3 Recurrent Multimodal Interaction (RMI)

RMI(Liu *et al.*, 2017) is a segmentation model which means that this model gives output in the form of a segmentation mask that localizes the intended object. They propose a two-layered convolutional multimodal LSTM network that explicitly models word-to-image interaction.

In this work, the convolutional multimodal LSTM takes both visual feature and language representation as input to generate the hidden state that retains both the spatial and semantic information in memory. Therefore its hidden state models how the multimodal feature progresses over time. After seeing the last word, they use a convolution layer to generate the image segmentation result. A multimodal LSTM (mLSTM) uses the concatenation of the language representation and the visual feature at a specific spatial location as its input vector.

This work is different from the above work in the sense that they make use of joint embedding to process the language and visual features altogether. However, in this work, the language part is processed using a program generator which generates an intermediate form of the functional program. Based on this functional program, a neural module network is constructed instead of using a multimodal LSTM. The visual features are then processed through this modular network and a segmented image is generated at the end. Also, one similarity is that they also use ResNet-101 pretrained on ImageNet like this work. This work also shows that this approach performs significantly better than RMI model approach.

3.4.4 IEP-REF

IEP-REF(Liu *et al.*, 2019) is also a segmentation model. They proposed a diagnostic dataset CLEVR-Ref+ for referring expression comprehension task and also a segmentation model.

They present a modular network approach which is adapted from (Johnson *et al.*, 2017b) work. They basically use an LSTM based program generator to translate the referring expression into a functional program built using the function catalogue. Based on this functional program, a neural module network is composed by using modules which have generic architectures for all functions. They use generic binary and unary modules to compose the network. By executing this dynamically constructed neural module network along with image features, a segmented image is produced. This work is similar in the sense that both of these approaches use neural module network approach for problem solving. However, the main difference between

these two works is that primitive operation modules are used for constructing neural module network. These modules are specific for their respective operation. For example, *Relate* module encodes the spatial information and *Attention* module focuses on visual information with some attributes like size, color etc.

Another similarity is that the program generator component is reused here for translating referring expression into the functional program. It is retrained for this task. Also, for input visual features, the last layer of the conv4 stage of ResNet101 is used which is pretrained on ImageNet. The output is of size $1024 \times 20 \times 20$. These image features are translated into lower dimensional image features of size $128 \times 20 \times 20$. However, here experiment is done with ResNet-152 image features as well.

Chapter 4

PROPOSED METHOD

4.1 An Overview On System Architecture

The previous chapters have covered the background which is required to understand the work in this thesis. Also, the related and existing methodologies for Referring Expression related tasks were also covered. In this chapter, the methodology used in this approach is explained. Given a referring expression and an image that contains multiple objects, the system should return a segmented image in which the referred object has been identified using the segmentation mask.

Before this approach is discussed, some design decisions should be explained which is the next part.

1. As mentioned previously, this work also builds upon the idea of Neural Module Network. However, the dataset which is required for this task is CLEVR-Ref+ which is a challenging dataset. This dataset differs from the original CLEVR dataset in some aspects of properties.

From table 4.1, it's clear that CLEVR-Ref+ dataset has above additional attributes which the work by (Mascharka *et al.*, 2018) didn't take into account. Hence the need to design modules for objects' ordinality and visibility attributes arises here.

	CLEVR-Ref+	CLEVR
Ordinality	+	−
Visibility	+	−

Table 4.1: Dataset Difference

Intuition for Visibility Attribute: Visibility of a particular object is a very simple and intuitive attribute. It's as comprehensive as the color or shape of the object. It basically shows whether an object is partially visible or fully visible. This gives the rise to the thought that the module which is used for focusing on attributes like color, shape should be able to model the visibility attribute. Hence for visibility attribute, Attention Module should be used here.

Intuition for Ordinality Attribute: Ordinality is a little more complex attribute than the previous one. For example, "*third of the cubes from the left*" phrase makes it clear how it is different than the Visibility attribute. First, all the cubes need to be found in the image, then the third cube from the left should be located. To localize these kinds of objects, the global context of the image is also required. In this case, there are two assumptions here.

- Treat ordinality attribute similar to color, shape attributes
- Keep the global context of the image into the consideration

Considering these two assumptions, there are two experiments with two plausible solutions. First is the use of the *Attention Module* similar to color and shape attributes similar to the work of (Mascharka *et al.*, 2018). The second solution seems to design a new module *FilterAttention Module* which encodes the global context in the image and makes it easy to follow the ordinality for objects.

2. This work experiments with the work of (Mascharka *et al.*, 2018) for Relate Module which basically encodes spatial information in TbD model. However, it does not improve on state of the art for this task. One possible reason could be the use of a larger feature map which is used in this work.

Intuition for Relate module modification: It is posited that since Large feature maps are being used here, Relate module does not seems to have better encoding for global context due to smaller receptive area. The assumption is that making use of an extra layer of dilated convolution should provide better encoding for a global context. Hence the proposal is that there should be made appropriate modifications in the Relate module in the form of adding dilated convolution.

3. This work also experiments with Same Module from (Mascharka *et al.*, 2018)'s work. Same module basically looks for the objects which have the same property as of the object mentioned in referring expression. In original work, simple concatenation of cross-related feature with previous attention map allows the convolutional filter to know which object was original and thus ignore it. However, in this work, an additional strategy is also applied and experimented with where instead of concatenation, subtraction of attention map from cross-related feature is performed.

Intuition for Same module modification: To explain the intuition, an example is given. There is an image of single-channel with dimension 4×4 and in that image, there are three classes of object and their labels are 1, 2, and 3. Here 0 denotes as background.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

If the object of interest in the referring expression where we want to apply Same operation is in the bottom right corner then previous module's attention map will focus on that corner object. In this case, input attention map would be:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After applying cross-relation on input image with attention map, the intermediate output within same module would be as below:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Here all the objects with same attribute are appearing. Now since Same module does not need to focus on the original object, it can be achieved by subtracting the input attention map from cross-related feature map and the output attention map would be as follow:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Here it can be seen that in the output attention map, the rest of the objects are appearing which have same property as the object in the bottom right corner.

4. Image feature extraction is an important component in this task. Previous works used ResNet-101 model for image feature extraction. ResNets are the model which are pre-trained on a very huge image dataset ImageNet. There is another deeper variant of ResNet which is ResNet-152. This work proposes to experiments with two sets of Image features.

- Extracted from ResNet-101
- Extracted from ResNet-152

The assumption here is that ResNet-152 model provides better image features and it is proved to perform better than ResNet-101 in Image classification task so it's assumed here that it should help in the task.

5. The NMN architecture in the work of (Mascharka *et al.*, 2018) by default produces an output of 128 x 20 x 20 feature map. This approach instead produces an output of 1 x 20 x 20 attention map because of the way the modules and Neural Module Network is setup in this work.

To generate a feature map from attention map, this work proposes to use Feature Regeneration Module which transforms attention map into feature map.

In the above points, the proposed additions and intuitions behind them are explained in the existing approach. In the following text, the architecture would be discussed in detail.

As this work shows in figure 4.1 below, first of all, features are extracted from the given image, and those features are transformed into lower-dimensional features using stem component. The referring expression is passed to a program generator component (Johnson *et al.*, 2017b) which transforms the referring expression into a functional program, and a neural module network is composed based on this func-

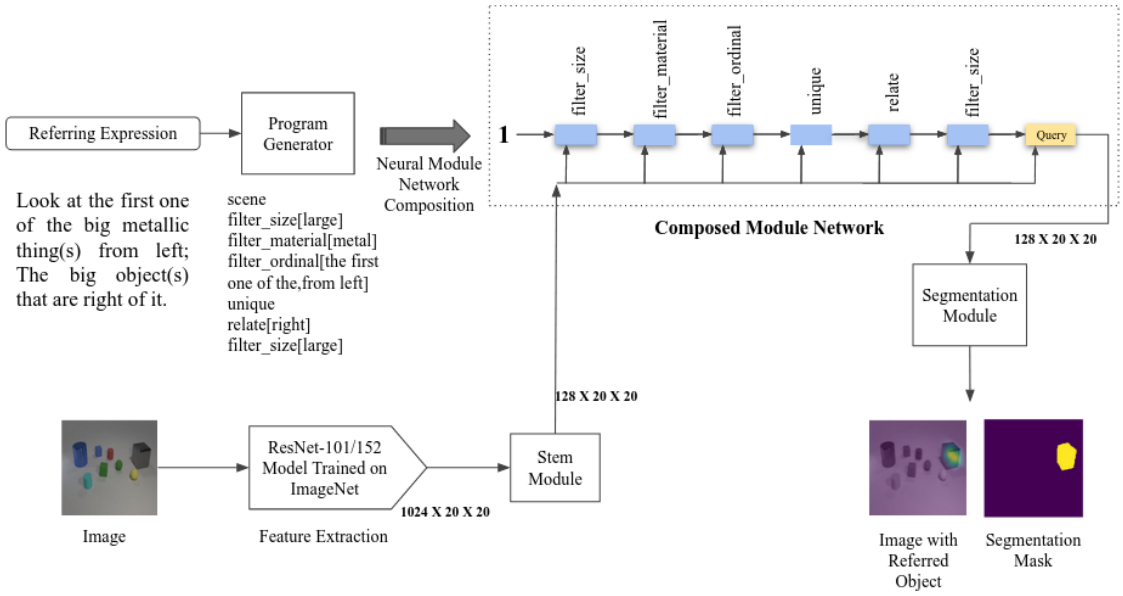

Figure 4.1: Architecture of The Proposed Model

tional program. For example, *Attention* modules are used for *filter* function, and *Relate* module is used for *relate* function. This neural module network is shown below in a much detailed view in figure 4.2. Here inputs are image features and attention map. The output of each module is an attention map.

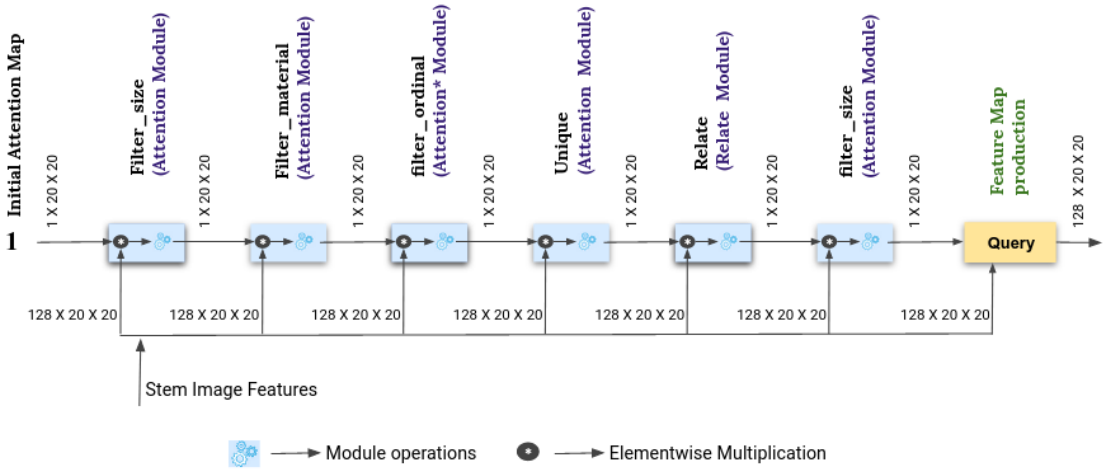

Figure 4.2: Detailed Overview of Neural Module Network

The features extracted from image, then, are passed through this neural module network and this network is executed along with them. The output of this neural module network is a one-dimensional attention map that is passed into a feature map generation module "Query" module. This module is considered as the end part of the neural module network. This "Query" module is further linked with the segmentation module at the end which generates segmentation mask for the given image and the referring expression. Here * represents that there are two variations of architecture for *filter_ordinal* function for which the intuition was explained previously.

This whole process involves the following variables:

1. w : an environment representation (image features)
2. x : a referring expression
3. y : a functional program
4. z : a ground truth image
5. M : a neural module network

These variables will be explored in further sections separately.

4.2 Dataset

It has been explained in the previous chapter that existing datasets have statistical biases associated with them. Ideally, the dataset should be free of any bias to make sure that the performance reflects the model's true level of understanding and reasoning capability. Reflecting on this kind of situation, (Johnson *et al.*, 2017a) proposed to use synthetic dataset CLEVR for the diagnostic purpose of these models. They constructed this dataset for Visual Question Answering task. (Liu *et al.*, 2019) created a new diagnostic dataset CLEVR-Ref+ for referring expression tasks by re-purposing CLEVR dataset.

4.2.1 The CLEVR-Ref+ Dataset

As it has been mentioned that CLEVR-Ref+ is built on top of CLEVR dataset, let's dive into the details of the same. This dataset requires complex reasoning to solve the task of referring expression comprehension and it can be used to conduct rich diagnostics to better understand the visual reasoning capabilities of the referring expression comprehension models.

Objects and Attributes:

The CLEVR-Ref+ contains objects which have following attributes with given set of values.

- Object Shape - Cube, Sphere, and Cylinder
- Object Size - Small, Large
- Object Material - Shiny "metal", Matte "rubber"
- Object's spatial relationships - left, right, behind, in front
- Object's color - blue, brown, cyan, gray, green, purple, red, yellow
- Object's Ordinality - second from left, fifth from front
- Object's Visibility - Partially Visible, Fully Visible

CLEVR-Ref+ dataset also contains one non-spatial relationship type which is the *same-attribute relation*. If two objects have equal attribute values for a specified attribute then the objects are said to be in that relationship.

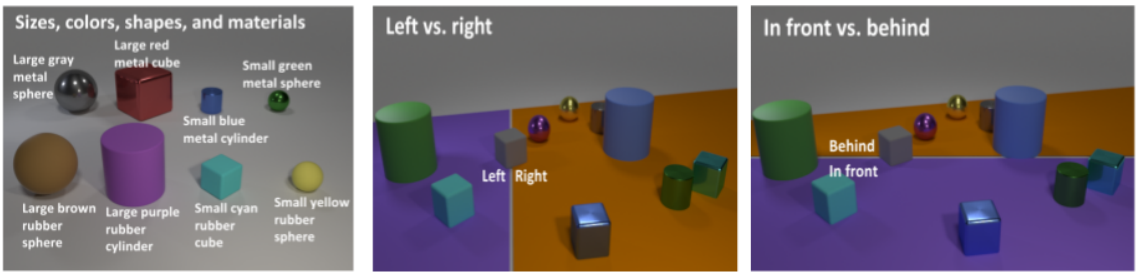

Figure 4.3: Objects, Their Attributes and Spatial Relationship

Image Representation:

The number of objects in each image varies between three to ten. These objects have random shapes, sizes, materials, colors, and positions. It has been made sure that while placing objects in images, no objects intersect or completely overlap each other. They are at least partially visible and that there are small horizontal and vertical margins between the image-plane centers of each pair of objects.

Referring Expression Representation:

Referring expressions are associated with their respective functional programs which are used to compose the neural module network, executing that along with image features gives the segmentation mask. A functional program is basically a structured collection of basic functions that correspond to primitive operations for visual reasoning such as querying object attributes (e.g. color). Thus, Referring Expressions can be represented by the composition of these functions or collectively as a functional program.

Referred Objects:

It's obvious that for referring expression comprehension task, the output won't be a a textual answer, but a bounding box or a segmentation mask.

Since CLEVR-Ref+ is synthetically developed, the exact 3D locations and properties of objects in the image are known. Ground truth programs which are associated with the referring expression can be used to identify the intended objects. Once the referred objects are obtained, those can be projected back to the image plane to get the ground truth segmentation mask. For this purpose, rendering was performed using Blender Software(Blender, 2018).

This dataset contains the exact same scenes as CLEVR dataset which is 70K images in the training set and 15K images in the test set. Each image is associated with 10 referring expressions.

4.3 Program Generator

It has been mentioned previously that the referring expression (x) is translated into a functional program (y) using the help of a program generator (π). When a referring expression is provided as input into this component, it produces a program that is composed of various functions from function catalog.

$$y = \pi(x)$$

Program Generator is an LSTM based sequence-to-sequence (Sutskever *et al.*, 2014) model. This program generator uses a multilayered Long Short-Term Memory (LSTM) to map the natural language referring expression as a sequence of words to a vector of fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector which is basically the predicted program as a sequence of functions.

Given a sequence of inputs $(x_1, \ldots, x_T)$, a standard Sequence-to-Sequence model produces sequence of outputs $(y_1, \ldots, y_{T'})$. The goal of the LSTM is to estimate the conditional probability $p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$ where $(x_1, \ldots, x_T)$ is an input sequence and $y_1, \ldots, y_{T'}$ is its corresponding output sequence whose length T' may

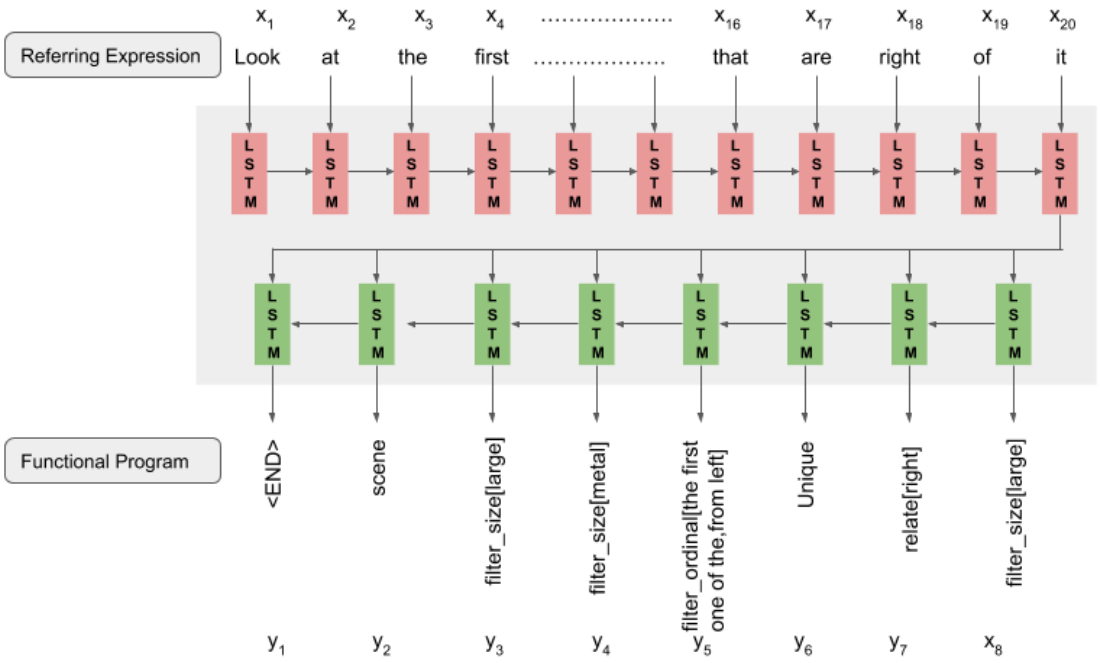

Figure 4.4: Seq2Seq Translation from Referring Expression to Functional Program Using Program Generator

differ from T. The LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence $(x_1, \ldots, x_T)$ given by the last hidden state of the LSTM, and then computing the probability of $y_1, \ldots, y_{T'}$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $x_1, \ldots, x_T$:

$$p\left(y_1, \ldots, y_{T'} | x_1, \ldots, x_T\right) = \prod_{t=1}^{T'} p\left(y_t | v, y_1, \ldots, y_{t-1}\right)$$

In this equation, each $p\left(y_t | v, y_1, \ldots, y_{t-1}\right)$ distribution is represented with a softmax over all the words in the vocabulary.

Figure 4.4 shows the high-level architecture of the program generator which takes a referring expression as input and generates a functional program as output. In this work, the encoder converts the discrete words of the input referring expression to vectors of dimension 300 using a learned word embedding layer; the resulting sequence

of vectors is then processed with a two-layer LSTM using 256 hidden units per layer. The hidden state of the second LSTM layer at the final timestep is used as the input to the decoder network. Below is the low-level architecture of the encoder component of the program generator.

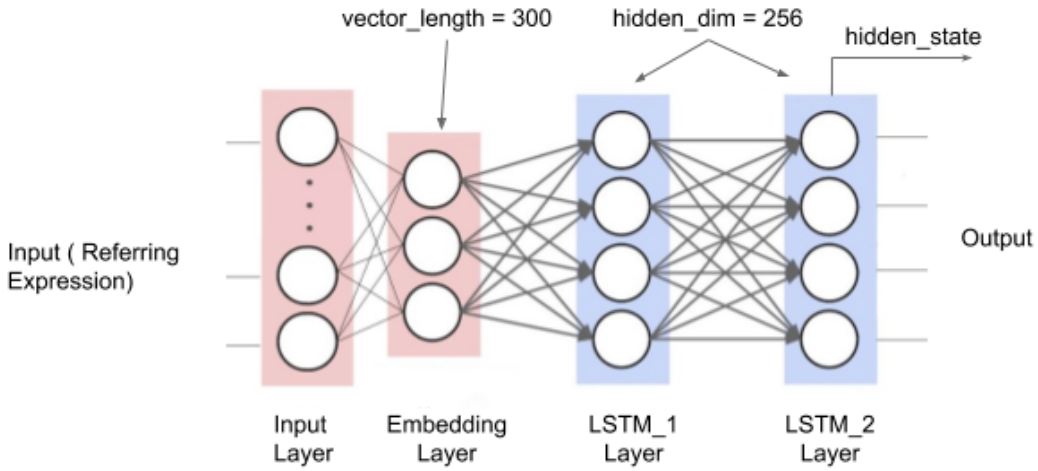

Figure 4.5: Architecture of Program Generator's Encoder, Adapted from (Fei-Fei Li, 2020)

The hidden state is calculated using below equation:

$$h_t = f\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$$

where x_t is the input vector at timestep t,

h_{t-1} is the hidden state at previous timestep,

$W^{(hx)}$ integrates input vector information,

$W^{(hh)}$ integrates information from the previous timestep

f is an activation function

At each timestep the decoder network receives both the function from the previous timestep (or a special <S T A R T> token at the first timestep) and the output from the encoder network. The function is converted to a 300-dimensional vector with a learned embedding layer and concatenated with the decoder output; the resulting

sequence of vectors is processed by a two-layer LSTM with 256 hidden units per layer. At each timestep the hidden state of the second LSTM layer is used to compute a distribution over all possible functions using a linear projection. Below is the low-level architecture of the decoder component of the program generator.

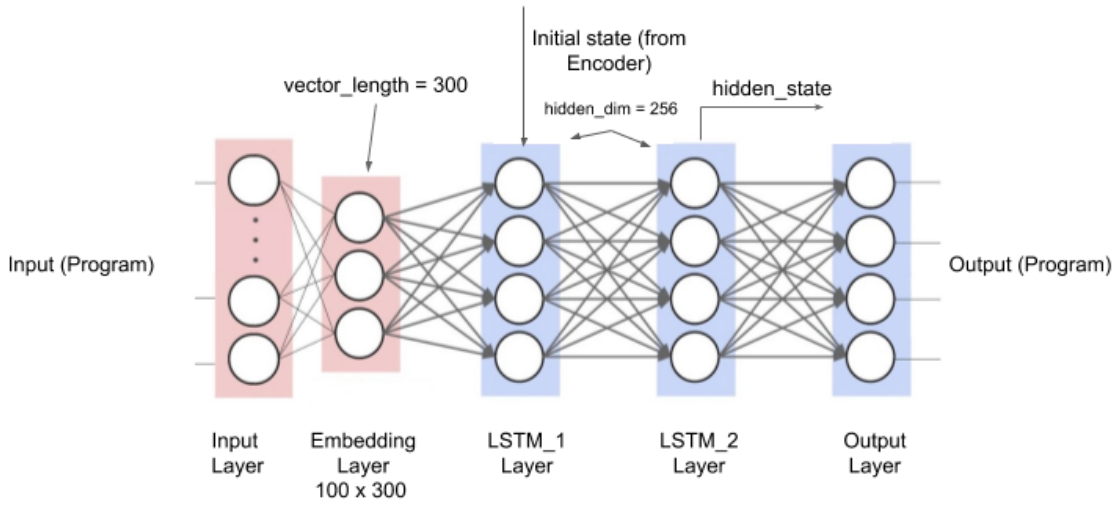

Figure 4.6: Architecture of Program Generator's Decoder, Adapted from (Fei-Fei Li, 2020)

The hidden state for the decoder is calculated using the below equation:

$$h_t = f\left(W^{(hh)} h_{t-1}\right)$$

The output y_t at timestep t is computed using below equation:

$$y_t = \text{softmax}\left(W^S h_t\right)$$

The outputs are calculated using the hidden state at the current time step together with the respective weight W^S. Softmax is used to create a probability vector that helps in determining the final output i.e. function.

In Figure 4.4 example, the referring expression is : "Look at the first one of the big metallic thing(s) from left; The big object(s) that are right of it." which is translated into following functional program (i.e sequence of programs).

- <END>
- scene
- filter_size[large]
- filter_material[metal]
- filter_ordinal[the first one of the,from left]
- unique
- relate[right]
- filter_size[large]

Once a functional program (y) is generated, a neural module network (M) is constructed using primitive operation modules ($n_1, n_2, n_3, \ldots$) which is covered in next section.

4.4 Primitive Operation Modules

As it has been mentioned in the previous section that a referring expression (x) is first translated into a functional program (y) using seq2seq based program generator (π) and based on that functional program, a neural module network (M) is constructed using modules from a set S of six types of primitive operation modules ($n_1, n_2, n_3, n_4, n_5, n_6$) corresponding to the specific functions in the functional program.

$$S := \{n_1, n_2, n_3, n_4, n_5, n_6\}$$

(Mascharka *et al.*, 2018) in their work Transparency by design, made use of primitive operation modules with attention map mechanism for Visual Question Answering task. This work adapts their approach and re-purpose the primitive operation modules for the referring expression comprehension task. This work also makes necessary changes in the architecture of those modules and add some modules for this task. These modules are designed to perform spatial transformations on visual attention to suit its specific task. The module architectures are the following:

- Attention Module
- Logical And Module
- Logical Or Module
- Relate Module
- Same Module
- Filter Attention Module

Now these modules will be discussed one by one. Before that, let's establish some notations for image features and module outputs.

- w : It represents the image features
- O_{i-1} : It represents the previous module output
- $\mathbf{1}$: It represents an all-ones tensor if there is no previous module output
- $\psi_{(k)}^{(Conv)} : \mathbb{R}^{\mathcal{C}_{k-1} \times \mathcal{H}_{k-1} \times \mathcal{W}_{k-1}} \rightarrow \mathbb{R}^{\mathcal{C}_k \times \mathcal{H}_k \times \mathcal{W}_k}$ represents a convolutional layer $\psi_{(k)}^{(Conv)}$ which produces an output o_k of dimension $\mathcal{C}_k \times \mathcal{H}_k \times \mathcal{W}_k$ from an input of dimension $\mathcal{C}_{k-1} \times \mathcal{H}_{k-1} \times \mathcal{W}_{k-1}$ where $\mathcal{W}_k = \left\lfloor \frac{\mathcal{W}_{k-1}}{s_k} \right\rfloor, \mathcal{H}_k = \left\lfloor \frac{\mathcal{H}_{k-1}}{s_k} \right\rfloor$ .. Here (s_k) is strides.

- ρ : It represents a $ReLU$ nonlinearity function
- σ : It represents a sigmoid nonlinearity function
- m_i : It represents i^{th} function in the functional program for which a module is chosen from the set of six predefined modules.

4.4.1 Attention Module:

This module focuses its attention to the regions of the image which contain an object with a specified property. For example, this module can be used to locate the green objects in the image. *Attention* module takes as input the image features along with a previous attention (or an all-one tensor in case of being the first *Attention* in the network.) It multiplicatively combines its input feature map and attention element-wise to focus the attention to the relevant region of the feature map. These attended features are processed by a series of convolutions and a heatmap of dimension $1 \times 20 \times 20$ is produced highlighting the objects that possess the attribute the module is looking for. This heatmap can be thought of as an *attention mask*. Here is the high-level diagram of Attention Module.

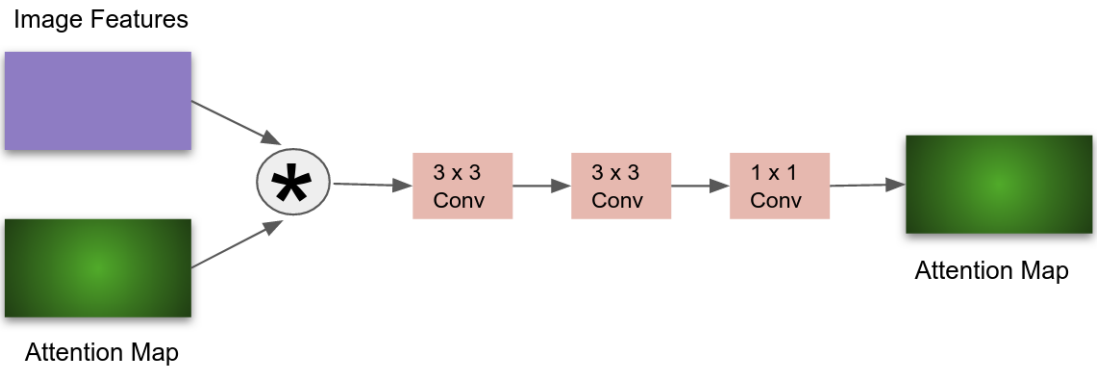

Figure 4.7: Architecture of Attention Module

If the image feature is x and previous module output is O_{i-1} then the intermediate processing would be as follows:

$$t_{m_i}^{128\times20\times20} = x^{128\times20\times20} \odot O_{i-1}^{1\times20\times20} \tag{4.1}$$

where t_{m_i} is the intermediate output for module corresponding to functionm_i after elementwise multiplication $\odot$ between above matrices and the superscript $(128 \times 20 \times 20)$ depicts the dimension of output. This intermediate output is passed through a series of Convolution layers and $ReLU$ as below:

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.2}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.3}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.4}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.5}$$

$$t_{m_i}^{1\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.6}$$

$$O_i^{1\times20\times20} = \sigma(t_{m_i}^{1\times20\times20}) \tag{4.7}$$

In the end, the intermediate output is passed through *sigmoid* function to produce a one-channel attention mask. Here kernel size of 3, zero-padding of 1 and stride of 1 is used for convolutional layers in equations 4.2 and 4.3. In equation 4.6, kernel size of 1, no padding and stride of 1 is used in the convolutional layer.

Let's consider an example where Attention Module is tasked with finding cubes. Given initial input attention of all ones, it will highlight all the cubes in the provided input features. Given an attention mask highlighting all the red objects, it will produce an attention mask highlighting all the red cubes.

Table 4.2: Attention Module

Index	Layer	Output Size
(1)	Features	$128 \times 20 \times 20$
(2)	Previous module output	$1 \times 20 \times 20$
(3)	Elementwise multiply (1) and (2)	$128 \times 20 \times 20$
(4)	Conv $(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(5)	ReLU	$128 \times 20 \times 20$
(6)	Conv $(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(7)	ReLU	$128 \times 20 \times 20$
(6)	σ(Conv $(1 \times 1, 128 \rightarrow 1)$)	$1 \times 20 \times 20$

*4.4.2 Logical **And** Modules*

It's a neural module that takes two attention masks as input and combines them using a set intersection. This is basically the same as taking the element-wise minimum of two attention masks.

If the two previous module outputs are O_j and O_k respectively for module m_j and m_k and these modules are acting as input for an AND module then the output of the module would be as follows:

$$O_i^{1\times20\times20} = O_j^{1\times20\times20} \owedge O_k^{1\times20\times20}$$

where $\owedge$ denotes the element-wise minimum.

Table 4.3: And Module

Index	Layer	Output Size
(1)	Previous module-1 output	$1 \times 20 \times 20$
(2)	Previous module-2 output	$1 \times 20 \times 20$
(3)	Elementwise minimum (1) and (2)	$1 \times 20 \times 20$

Let's take an example where this module could be used. Consider the referring expression "spheres that are left of the small metal cylinder **and** right of the green cube." Once regions which are in left of the small metal cylinder and right of the green cube have been localized, an *And* module would be used to find the intersection of the two. Here the output can be passed to an *Attention* module which will find spheres.

4.4.3 Logical **Or** *Modules*

It's a neural module that takes two attention masks as input and combines them using a set union. This is basically the same as taking the element-wise maximum of two attention masks.

If the two previous module outputs are O_j and O_k respectively for module m_j and m_k and these modules are acting as input for an OR module then the output of the module would be as follows:

$$O_i^{1\times20\times20} = O_j^{1\times20\times20} \Vee O_k^{1\times20\times20}$$

where $\Vee$ denotes the element-wise maximum.

Table 4.4: Or Module

Index	Layer	Output Size
(1)	Previous module-1 output	$1 \times 20 \times 20$
(2)	Previous module-2 output	$1 \times 20 \times 20$
(3)	Elementwise maximum (1) and (2)	$1 \times 20 \times 20$

Let's take an example where this module could be used. Consider the referring expression "cubes that are left of the grey cylinder **or** right of the large sphere." Once regions left of the grey cylinder and right of the large sphere have been localized, an *Or* module would be used to find the union of the two. Here the output can be passed to an *Attention* module which will find cubes.

4.4.4 Relate Module

This module focuses its attention to the region that has some spatial relation to the other region. For example, this module can be used for the spatial relationship between different regions. *Relate* module takes as input the image features along with attention mask from the previous module and produces an attention mask. Here is the high-level diagram of the Relate Module.

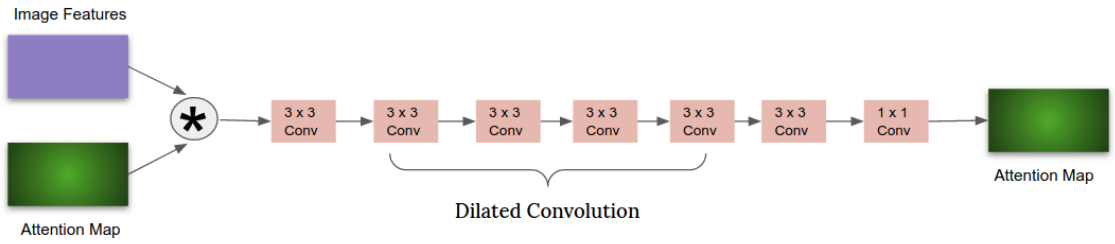

Figure 4.8: Architecture of Relate Module

As per the intuition mentioned previously, an extra layer of dilated convolution is added in the Related Module. If the image feature is x and the previous module output is O_{i-1} then the intermediate processing would be as follows:

$$t_{m_i}^{128\times20\times20} = x^{128\times20\times20} \odot O_{i-1}^{1\times20\times20} \tag{4.8}$$

where t_{m_i} is the intermediate output for module m_i after elementwise multiplication $\odot$ between above matrices and the superscript $(128 \times 20 \times 20)$ depicts the dimension of output. This intermediate output is passed through a series of Convolution layers and $ReLU$ as below:

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.9}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.10}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.11}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.12}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.13}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.14}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.15}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.16}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.17}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.18}$$

$$t_{m_i}^{128\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.19}$$

$$t_{m_i}^{128\times20\times20} = \rho(t_{m_i}^{128\times20\times20}) \tag{4.20}$$

$$t_{m_i}^{1\times20\times20} = \psi^{(Conv)}(t_{m_i}^{128\times20\times20}) \tag{4.21}$$

$$O_i^{1\times20\times20} = \sigma(t_{m_i}^{1\times20\times20}) \tag{4.22}$$

Table 4.5: Relate Module

Index	Layer	Output Size
(1)	Features	$128 \times 20 \times 20$
(2)	Previous module output	$1 \times 20 \times 20$
(3)	Elementwise multiply (1) and (2)	$128 \times 20 \times 20$
(4)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 1)$	$128 \times 20 \times 20$
(5)	ReLU	$128 \times 20 \times 20$
(6)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 2)$	$128 \times 20 \times 20$
(7)	ReLU	$128 \times 20 \times 20$
(8)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 4)$	$128 \times 20 \times 20$
(9)	ReLU	$128 \times 20 \times 20$
(11)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 8)$	$128 \times 20 \times 20$
(12)	ReLU	$128 \times 20 \times 20$
(13)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 16)$	$128 \times 20 \times 20$
(14)	ReLU	$128 \times 20 \times 20$
(15)	$\text{Conv}(3 \times 3, 128 \rightarrow 128,\ \text{dilate } 1)$	$128 \times 20 \times 20$
(16)	ReLU	$128 \times 20 \times 20$
(17)	$\sigma(\text{Conv}(1 \times 1, 128 \rightarrow 1))$	$1 \times 20 \times 20$

In the end, the intermediate output is passed through *sigmoid* function to produce a one-channel attention mask. Here kernel size of 3 is used for all convolution layers except the one in equation 4.21 where kernel size of 1 is used. In equation 4.9, 4.11, 4.13, 4.15, 4.17 and 4.19, padding and dilation of 1, 2, 4, 8, 16 and 1 are used respectively. Convolution layer in Equation 4.22 uses no padding. All the convolution layers use a stride of 1.

let's consider an example, "Cylinder to the left of the red cube." First of all, *Attention* modules would be used in the neural module network to determine the position of red cube, after that, a *Relate* would be used to attend the region which is spatially to the left. This module needs the global context for its operation so that regions to the far left can be influenced by an object on the far right. For this purpose, a series of dilated convolutions (Yu *et al.*, 2016) is used which expands the receptive field to the entire image and thus providing the global context required by *Relate* module. Here an extra layer of Dilated Convolution is used as mentioned in the intuition.

4.4.5 Same Module

This module focuses its attention to a region and extracts a property from that region, and then attends to every other region in the image that has the same property. As similar to *Relate* module, a *Same* module also takes into account context from distant spatial regions but its mechanism is different from that. *Same* module takes input features and an attention mask and produces an attention mask. It basically determines the index of the maximally-attended object, extracts the feature vector at that spatial location, then performs a cross-correlation between the object of interest and every other object in the scene to determine which objects share the same property as the object of interest. This correlated feature map then goes through a convolution block which produces an attention mask.

Based on the intuition, there are two experiments which take place in terms of Same module. In the first experiment, default concatenation is performed between cross-related image features and input attention map. In the second experiment, the subtraction of input attention map from cross-related feature map is performed to focus on objects with same attributes.

In Figure 4.9, it is represented in a single diagram as *Concatenation/Subtraction* denoting both architectural choices. However, in rest of the section, only concatenation part is expanded here with details.

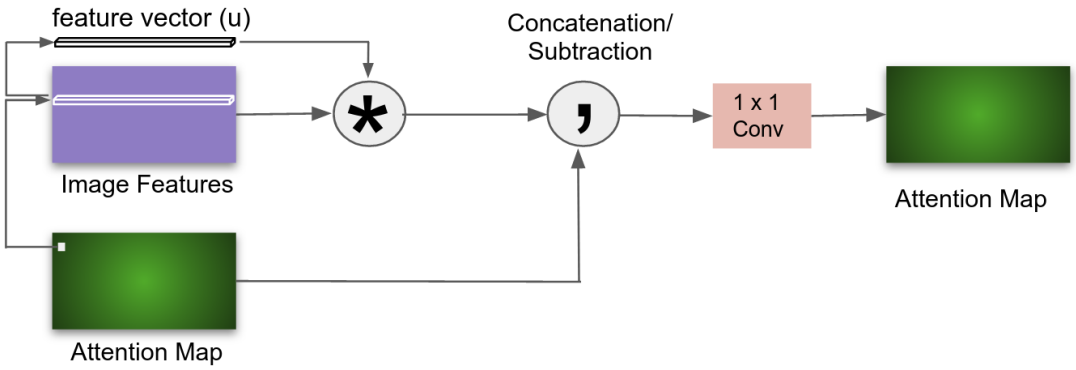

Figure 4.9: Architecture of Same Module

If the image feature is x and previous module output is O_{i-1} then, first of all, the arg max of O_{i-1} is taken spatially at each dimension. Doing so provides the coordinates (p^*, q^*) of the object for which *Same* operation needs to be performed.

$$(p^*, q^*)^{1\times1\times1} = \arg\max_{p,q} O_{i-1}^{1\times20\times20} \tag{4.23}$$

Now that, the coordinates of the object of the interest are obtained, the feature vector (u) at spatial location identified by the coordinate in the input feature map is extracted. This process gives the feature vector (u) which encodes the attribute of interest of that object.

$$u^{128\times1\times1} = x_{(p^*,q^*)}^{128\times20\times20} \tag{4.24}$$

Now elementwise multiplication is performed between feature vector (u) and input image feature (x) which basically is cross-correlation of feature vector of interest (u) with the feature vector of every other position in the image feature. This gives the cross-correlated feature map (v).

$$v^{128\times20\times20} = u^{128\times1\times1} \odot x^{128\times20\times20} \tag{4.25}$$

Here it needs to be taken care of the fact that *Same* operation must not attend to the original object because its task is to identify other objects with same property as of the object of interest. Hence, now original attention mask O_{i-1} is concatenated with cross-correlated feature map (v). This allows the convolutional filter to know which object was original and thus it can be ignored.

$$t_{m_i} = (v^{128\times20\times20}, O_{i-1}^{1\times20\times20}) \tag{4.26}$$

where t_{m_i} is the intermediate output for module m_i after elementwise concatenation of (v) and O_{i-1}. This intermediate output is passed through a Convolution layer and nonlinearity function as below:

$$t_{m_i}^{1\times20\times20} = \psi^{(Conv)}(t_{m_i}^{129\times20\times20}) \tag{4.27}$$

$$O_i^{1\times20\times20} = \sigma(t_{m_i}^{1\times20\times20}) \tag{4.28}$$

Let's consider an example, "Cylinder of the same color as the small cube.". First of all, the neural module network localizes the small cube using *Attention* module, then *Same* module is used to determine its color which produces an attention mask highlighting every region of an image that shares the same color as an object of interest (excluding the original object). Once all the objects with same color have been localized, *Attention* modules can be used to identify the cylinder of the same color.

Table 4.6: Same Module

Index	Layer	Output Size
(1)	Features	$128 \times 20 \times 20$
(2)	Previous module output	$1 \times 20 \times 20$
(3)	$\arg\max_{x,y}(2)$	$1 \times 1 \times 1$
(4)	$(1)_{(3)}$	$128 \times 1 \times 1$
(5)	Elementwise multiply (1) and (4)	$128 \times 20 \times 20$
(6)	Concatenate (5) and (2)	$129 \times 20 \times 20$
(7)	$\sigma(\text{Conv}(1 \times 1, 129 \rightarrow 1))$	$1 \times 20 \times 20$

4.4.6 Filter Attention Module

This module is designed to experiment with *filter_ordinal* function. As it has been mentioned previously that this functions might need global context in order to focus intended object, It's built on top of plain *Attention Module* by adding dilated convolutions. The architecture is as below in figure 4.10.

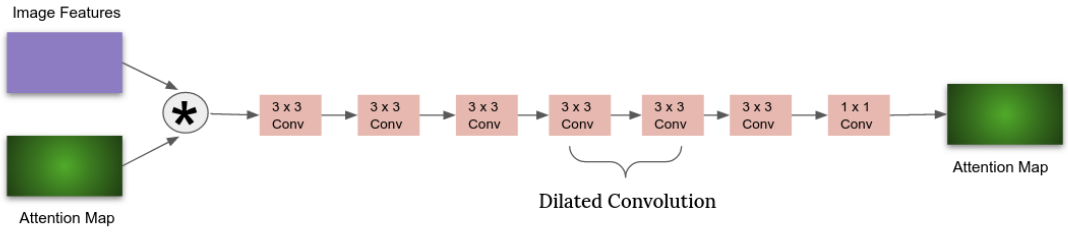

Figure 4.10: Architecture of Filter Attention Module

Further, the implementation detail for this module is in table 4.7.

Table 4.7: Filter Attention Module

Index	Layer	Output Size
(1)	Features	$128 \times 20 \times 20$
(2)	Previous module output	$1 \times 20 \times 20$
(3)	Elementwise multiply (1) and (2)	$128 \times 20 \times 20$
(4)	Conv$(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(4)	Conv$(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(4)	Conv$(3 \times 3, 128 \rightarrow 128,$ dilate 1)	$128 \times 20 \times 20$
(5)	ReLU	$128 \times 20 \times 20$
(6)	Conv$(3 \times 3, 128 \rightarrow 128,$ dilate 2)	$128 \times 20 \times 20$
(7)	ReLU	$128 \times 20 \times 20$
(8)	Conv$(3 \times 3, 128 \rightarrow 128,$ dilate 4)	$128 \times 20 \times 20$
(9)	ReLU	$128 \times 20 \times 20$
(15)	Conv$(3 \times 3, 128 \rightarrow 128,$ dilate 1)	$128 \times 20 \times 20$
(16)	ReLU	$128 \times 20 \times 20$
(17)	σ(Conv$(1 \times 1, 128 \rightarrow 1)$)	$1 \times 20 \times 20$

4.5 Other Components

4.5.1 Functional Program

As it has been mentioned previously, A referring expression is transformed into a functional program using *Program Generator*. It basically takes referring expression as input and produces a corresponding functional program. Based on this functional program a neural module network is constructed. This functional program is built using some basic functions. These functions are explained below:

- **Filtering functions:** These functions serve the purpose of filtering the input objects by some attribute and they return the subset of input objects which follow the input attribute. For example, *filter_color* with first input *red* will return the set of all red objects in the second input. (Johnson *et al.*, 2017b) gave following filtering functions

  - **filter_size** : filters based on size
  - **filter_color** : filters based on color
  - **filter_material** : filters based on material
  - **filter_shape** : filters based on shape

  Later, while working with CLEVR-Ref+, (Liu *et al.*, 2019) realized that for referring expression task, "ordinal" and "visible" category were not present in the function catalog. Those were added further hence now there are two more filtering functions.

  - **filter_ordinal** : filters based on ordinal for phrases such as "the third cube from right"
  - **filter_visible** : filters based on the visibility of the objects such as "barely visible green cylinder"

- **Relate Function:** This function specializes in the spatial relationship of the objects. When used, it returns all the objects in the image that have the specified spatial relationship with the input object. For example, if the input object is *Large* cylinder and the input spatial relation is *right*, then it would return the set of all objects in the image that are right of the large cylinder.

- **Logical Operators**
  - **AND** : This function refers to the *intersection*. As the name suggests, it returns the intersection of the two input sets
  - **OR** : This function refers to the *union*. As the name suggests, it returns the union of the two input sets
- **Same-attribute relation Function:** These functions return the set of objects which have the same attribute value as the input object (while not including the input object). For example, function *same_color* on a sphere would return a set of objects which have same as of the sphere in the image while excluding the input object. These are the following functions for this purpose.
  - **same_size**
  - **same_color**
  - **same_material**
  - **same_shape**
- **Scene:** This function returns the set of all objects in the image.
- **Unique Function:** If the input is a singleton set then this function returns it as a standalone object.

4.5.2 Neural Module Network Composition

The details of various primitive modules are already laid out in section 4.4. These modules are the building blocks of the neural module network which is built according to the functional program.

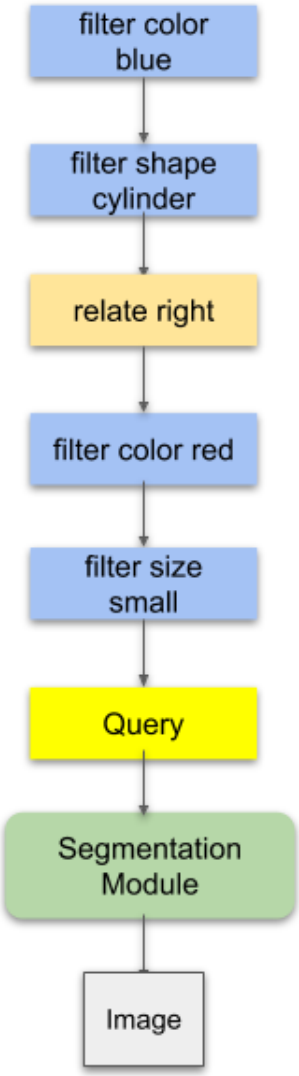

Figure 4.11: Example of Neural Module Network

As mentioned previously, A referring expression is transformed into a functional program using the program generator component based on the function catalog. Based on this functional program, a neural module network is constructed using the primitive operation modules. A module is a small neural network used to perform a given logical step. By composing appropriate neural modules together, referring expression comprehension task can be carried out and a referred object can be returned.

For example, Let's consider this referring expression : "**small red object which is right of the blue cylinder**". To refer that small red object, the output of the module which locates small objects can be composed with a module which finds objects which are blue in color, then with a module that finds cylinder shaped objects, then with a module which performs relate right operation which is followed by a module which finds red objects. This output finally goes into the module which finds small objects. The output from this last module goes to "Query" module which is explained further. The output of "Query" module is passed through the segmentation module which returns the image with a segmented mask as referred object.

4.5.3 Feature Map Production Using Query Module

The output of the composed neural module network is a one-dimensional attention map. Before sending to segmentation module, It needs to be made sure that it is converted to its original dimensional feature map $128 \times 20 \times 20$ from the size of $1 \times 20 \times 20$. This task is performed using "Query" module which takes attention map from the neural module network and the stem image features as input and by performing elementwise multiplication, it produces a feature map of $128 \times 20 \times 20$. This feature map is passed to the segmentation module further.

4.5.4 Image Feature Extraction Using Transfer Learning

In Deep learning, the models are trained with a large volume of data and they learn model weights and bias during training. These weights can be used by other network models to address other related problems. In computer vision, transfer learning(Pan and Yang, 2010) is usually expressed through the use of pre-trained models' weights from one or more layers. A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that needs to

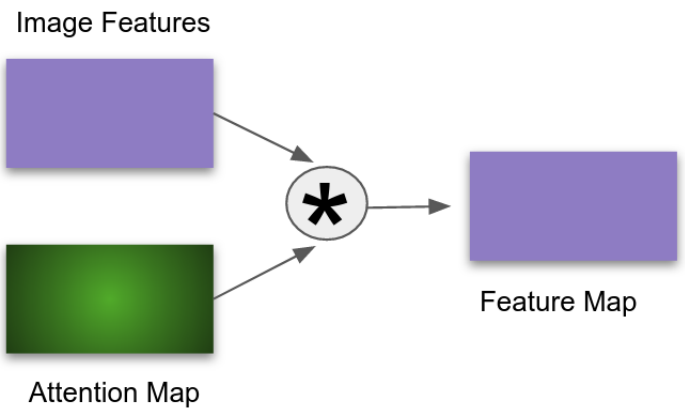

Figure 4.12: Query Module

be solved. Transfer learning can be implemented either by using layers' weights or by extracting features by directly using pre-trained models. Most commonly used pre-trained models in Computer vision is VGGNet(Simonyan and Zisserman, 2014), ResNet(He *et al.*, 2015) and Inception (Szegedy *et al.*, 2014).

In this work, Tthe feature extraction technique of transfer learning is applied. For the purpose of input visual features, ResNet-101 and ResNet-152 models are experimented with. These models are trained on ImageNet. The last layer output of the conv_4 stage of these models is used as image features which are of size $1024 \times 20 \times 20$. The input image is resized to 320×320 similar to work done by (Liu *et al.*, 2019).

These image features are high-dimensional in nature hence, in order to make them suitable for the model, these features are passed through a convolutional block called the "stem". This stem module transforms high-dimensional feature input from ResNet into lower-dimensional image features. It has convolutional and ReLU layers. The lower-dimensional features are further used by the neural module network which was explained in the previous section.

Table 4.8: Stem Block

Layer	Output Size
Input Image	$3 \times 320 \times 320$
ResNet101 conv4_6	$1024 \times 20 \times 20$
Conv$(3 \times 3, 1024 \rightarrow 128)$	$128 \times 20 \times 20$
ReLU	$128 \times 20 \times 20$
Conv$(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
ReLU	$128 \times 20 \times 20$

4.5.5 Segmentation Module

This module is the final part of the model which works to provide final segmented image prediction $z^{'}$ for the referring expression.

Once the neural module network along with "Query" module generates an output O_M (which is 128-channel feature tensor) according to the referring expression, this module takes that as input and transforms that into a 1-channel segmentation mask. This module has a submodule called "Residual Block" which performs convolution and $ReLU$ operations. It's denoted with Res.

First of all, O_M is passed through the residual block which generates an intermediate output t_M.

$$t_M^{128\times20\times20} = Res(O_M^{128\times20\times20}) \tag{4.29}$$

The output of this submodule is passed through an another series of convolution and $ReLU$ operation.

$$t_M^{128\times20\times20} = \psi^{(Conv)}(t_M^{128\times20\times20}) \tag{4.30}$$

Table 4.9: Residual Block

Index	Layer	Output Size
(1)	Previous module output	$128 \times 20 \times 20$
(2)	Conv$(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(3)	ReLU	$128 \times 20 \times 20$
(4)	Conv$(3 \times 3, 128 \rightarrow 128)$	$128 \times 20 \times 20$
(5)	Residual: Add (1) and (4)	$128 \times 20 \times 20$
(6)	ReLU	$128 \times 20 \times 20$

$$t_M^{128\times20\times20} = \rho(t_M^{128\times20\times20}) \tag{4.31}$$

It's noticeable from the output that it is of dimension $128 \times 20 \times 20$ but the output is expected to be the dimension of $128 \times 320 \times 320$ to get the output with respect to the output image. For this purpose, the upsampling of image features needs to be performed. Bilinear upsampling(Dong *et al.*, 2015) is used in this task and intermediate output of $128 \times 320 \times 320$ is generated. Bilinear upsampling is denoted with $BiUp$ here.

$$t_M^{128\times320\times320} = BiUp(t_M^{128\times20\times20}) \tag{4.32}$$

This output is further passed through a series of convolution layers and $ReLU$ operations as shown in table 4.8.

This process produces the final predicted output z' of dimension $1 \times 320 \times 320$.

$$t_M^{128\times20\times20} = \psi^{(Conv)}(t_M^{128\times20\times20}) \tag{4.33}$$

$$t_M^{128\times20\times20} = \rho(t_M^{128\times20\times20}) \tag{4.34}$$

$$t_M^{32\times20\times20} = \psi^{(Conv)}(t_M^{128\times20\times20}) \tag{4.35}$$

Table 4.10: Segmentation Module

Layer	Output Size
Neural module Network output	$128 \times 20 \times 20$
Residual Block	$128 \times 20 \times 20$
Conv$(1 \times 1, 128 \rightarrow 128)$	$128 \times 20 \times 20$
ReLU	$128 \times 20 \times 20$
Bilinear upsample	$128 \times 320 \times 320$
Conv$(1 \times 1, 128 \rightarrow 128)$	$128 \times 320 \times 320$
ReLU	$128 \times 320 \times 320$
Conv$(1 \times 1, 128 \rightarrow 32)$	$32 \times 320 \times 320$
ReLU	$32 \times 320 \times 320$
Conv$(1 \times 1, 128 \rightarrow 4)$	$4 \times 320 \times 320$
ReLU	$4 \times 320 \times 320$
Conv$(1 \times 1, 4 \rightarrow 1)$	$1 \times 320 \times 320$

$$t_M^{32\times20\times20} = \rho(t_M^{128\times20\times20}) \tag{4.36}$$

$$t_M^{4\times20\times20} = \psi^{(Conv)}(t_M^{32\times20\times20}) \tag{4.37}$$

$$t_M^{4\times20\times20} = \rho(t_M^{32\times20\times20}) \tag{4.38}$$

$$z'^{1\times20\times20} = \psi^{(Conv)}(t_M^{4\times20\times20}) \tag{4.39}$$

Here it's evident that after following this model's approach, a predicted image with a segmented object is produced as output.

To summarize, a referring expression x is transformed into a functional program y with the help of program generator π. A neural module network M is constructed based on this functional program and executed along with image features w which yields a neural module network output O_M. This output is passed through the segmentation module which produces the predicted image with the desired object in segmentation mask as output.

Chapter 5

EXPERIMENTS & RESULTS

5.1 Experiments

5.1.1 Dataset

The experiments are performed on the CLEVR-Ref+ dataset which is a synthetic dataset specifically designed for the referring expression comprehension task. This dataset was built solely for the purpose of dealing with biases issues considering the exploitation of biases in the real world datasets like RefCOCO and RefCOCO+. The goal is to make sure that the model reflects the true understanding of the comprehension task and it does not get the benefit of biases already present in the dataset.

The CLEVR-Ref+ dataset contains the exact same images from the CLEVR dataset. In CLEVR dataset, there are 70K images in the training set and 15k images in validation and test set. In CLEVR-Ref+, each image is associated with 10 referring expressions. However, in the work of (Liu *et al.*, 2019), they use only images from the test set and provide the final result on that set. Here, the same strategy is followed to be fair with the results, and the trained model is evaluated on the test dataset. There are various categories for referring expressions in table 5.1.

Another point to note here is that for this task, the output is not a textual answer, instead, it's a bounding box or a segmentation mask. Since the exact 3D locations and properties of objects in the image are already known, the ground truth program associated with the referring expression can be followed to identify which objects are being referred.

Table 5.1: Categories of Referring Expressions

Category	**Referring Expression (CLEVR-Ref+)**
Basic	The cyan cubes.
Spatial Relation	The green cylinders to the left of the brown sphere.
AND Logic	The green spheres that are both in front of the red cylinder and left to the yellow cube.
OR Logic	Cylinders that are either purple metal objects or small red matte things.
Same Relation	The things/objects that have the same size as the red sphere.

5.1.2 Evaluation Metrics

As mentioned previously, *Intersection over Union (IoU)* is used as the experiment's evaluation metric. IoU which is also known as Jaccard Index(Jaccard, 1901; Kosub, 2019), is the prominently used evaluation metric for comparing the similarity between two bounding boxes or segmentation masks. IoU encodes the shape properties (e.g. the widths, heights, and locations of two bounding boxes) of the objects under comparison) into the region property and then calculates a normalized measure that focuses on their areas. Thus, it is used to measure the accuracy of an object detection model on a particular dataset. IoU can be thought of as a metric to evaluate how close the prediction bounding box is to the ground truth.

To evaluate the IoU metric, the ground truth bounding box or segmentation mask and model predicted bounding box or segmentation mask are required. It is computed by using the following formulae.

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

For this experiment, Overall IoU is calculated as the total intersection area between prediction and ground-truth divided by total union area accumulative over the test samples.

5.1.3 Experimental Setup and Results

For the experiment, the images are resized to 320×320 so that the results of this work can be compared with the work done by (Liu *et al.*, 2019). The program generator component from previous work (Johnson *et al.*, 2017b) is reused in this work to generate the functional program from the referring expression.

In this approach, all the images have corresponding 10 referring expressions. All of the referring expressions are converted into functional programs using the program generator component. As another aspect of the model, lower-dimensional visual features are produced by passing the higher dimensional features (generated by ResNet model which is trained on ImageNet) into "stem" module which is a simple convolutional block. In this experiment, ground truth programs are used to train the neural module network. The training procedure thus takes triplets of image, program, and answer for training. This work uses Adam(Kingma and Ba, 2014) optimization method with the learning rate set to the value of 5×10^{-5} . Cross-Entropy is used as the loss function here. During training, IoU is used as the evaluation metric. Since ground truth programs are also provided for the test set, those can be used directly along with image features to generate predicted output and check against the ground truth answers. The batch size was set to 16. In the following subsection, the training process will be discussed and details of how it learns its module parameters will be presented.

Training Process :

As mentioned previously, the training procedure takes triplets of image, program, and answer for training. Due to the nature of architecture, This work has a dynamic modular structure instead of a monolithic one. It makes it easy for different types of Referring expressions. During the training process, parameters are not learned for a single neural network, instead, the parameters for multiple small neural networks (which are called neural modules here) are learned. These module parameters are jointly learned during training. Throughout this training process, these parameters are shared across neural module networks corresponding to different referring expressions. Below is the listing out of the training process in a simple manner for better understanding.

- Initially there is something similar to a bag of modules that contains neural modules for all the relevant functions in the vocabulary. For example, since there are 8 colors, there will be 8 different instantiations of *Attention Module*. *filter_color[red]* will have a different module than that of *filter_color[green]*. Same goes for other attributes as well.

- During training, for each referring expression, a functional program is generated. According to this functional program, modules which are needed, are pulled out and put together.

- For each referring expression, a learning signal is provided to some subset of the modules.

- Repeated iterations through referring expressions eventually hits all of the modules and update the module parameters.

- Thus, this training process allows those neural modules to specialize to their individual tasks.

An example is considered below to understand how these modules learn. There are two referring expressions as follows which refer to different objects.

$$x_1 = \text{The big green cubes}$$

$$x_2 = \text{small green spheres}$$

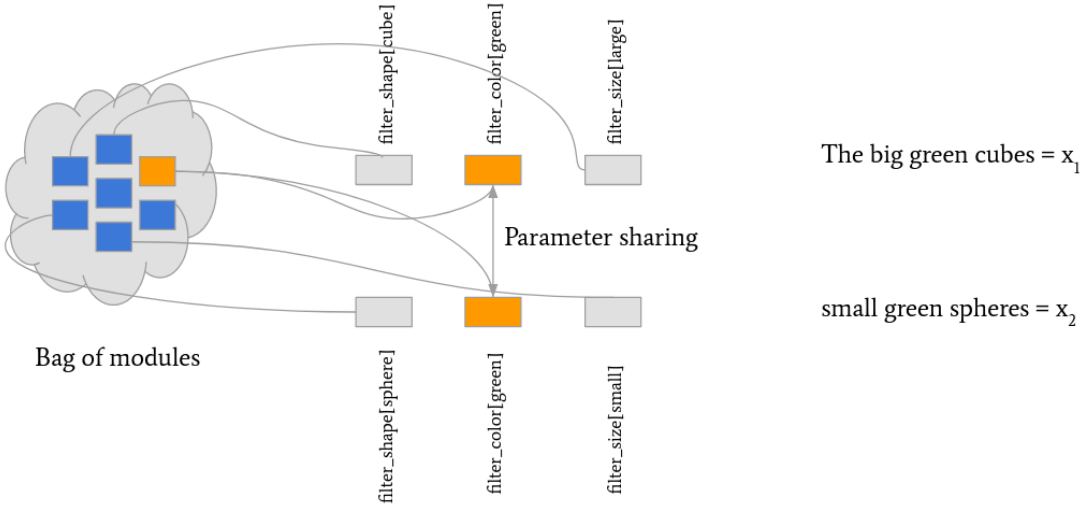

Figure 5.1: Training Process and Parameters Sharing

when these referring expressions are processed through Program Generator then they refer to following corresponding modules.

$$x_1 \rightarrow \text{filter\_shape[cube], filter\_color[green], filter\_size[large]}$$

$$x_2 \rightarrow \text{filter\_shape[sphere], filter\_color[green], filter\_size[small]}$$

Now there are these following Notations:

- Lower dimension encoded image features : $E_I = Stem(I)$ where I = Extracted Image Features, $Stem$ = Stem Module
- Initial All-ones Attention mask : $attn\_one$
- Generated attention mask from a module : $attn\_filt$

Now there are below shorthand notation for the modules

- Filter_shape[sphere] : filt-sphere
- Filter_shape[cube] : filt-cube
- Filter_color[green] : filt-green
- Filter_size[small] : filt-small
- Filter_size[large] : filt-large

The computation for these referring expressions would be as follows:

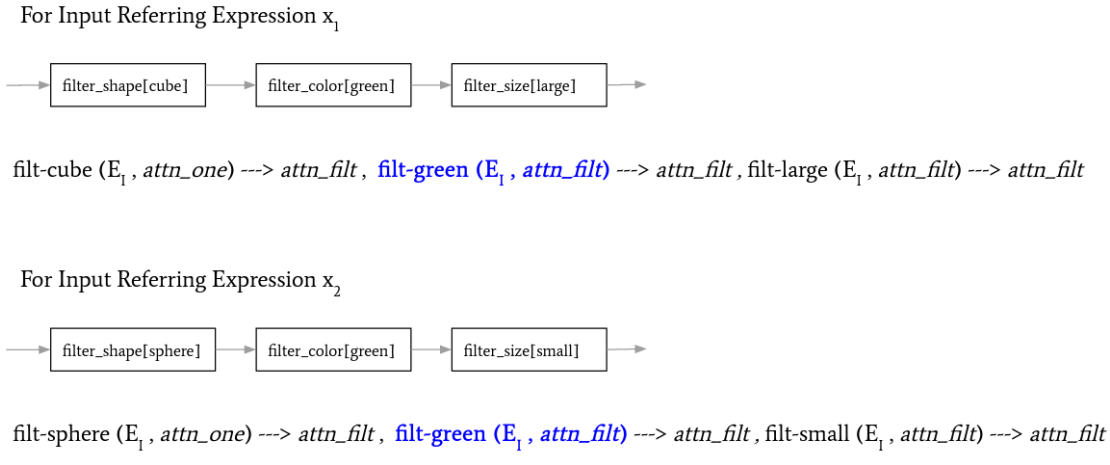

Figure 5.2: Example of Parameters Sharing

It is evident that here $filter\_color[green]$ module is common in both of the referring expressions. During training iteration, this module learns its parameters which are shared throughout the training process until training is completed. This strategy helps the neural module network to learn the module parameters for all of its modules. The output of the neural module network is followed by the Feature regeneration module. The output of Feature Regeneration Module (Query Module) is passed through Segmentation module which generates a predicted mask. The process is as shown below.

For Input Referring Expression x_i , $attn\_filt$ is generated.

- $Query(E_I, attn\_fit) \rightarrow E_I^{'}$ Where $E_I^{'}$ is regenerated feature map and Query is query module
- $Seg(E_I^{'}) \rightarrow y_p$ Where Seg is Segmentation Module and y_p is predicted segmented image tensor.
- Loss = CrossEntropy(y_p, y_{gt})

Cross-Entropy Loss is used as the loss function here which takes predicted segmented mask and ground truth answer as Input and produces loss.

5.1.4 Evaluation

The experimental results are summarized in Table 5.2. Detection models are evaluated by accuracy (i.e. whether the prediction selects the correct bounding box among given candidates), where MAttNet performs favorably against SLR. Segmentation models are evaluated by Intersection over Union (IoU), where IEP-Ref performs significantly better than RMI. This suggests the importance of model compositionality within the referring expression.

	Basic	Spatial Relation			Logic				
	0-Relate	1-Relate	2-Relate	3-Relate	AND	OR	Same	Accuracy	IoU
SLR	0.627	0.569	0.57	0.584	0.594	0.701	0.444	0.577	-
MAttNet	0.566	0.623	0.634	0.624	0.723	0.737	0.454	0.609	-
RMI	0.822	0.713	0.736	0.715	0.585	0.679	0.251	-	0.561
IEP-Ref ($GT_{program}$)	0.928	0.895	0.908	0.908	0.879	0.881	0.647	-	0.816
IEP-Ref (700K)	0.92	0.884	0.902	0.898	0.86	0.869	0.636	-	0.806
Our Model($GT_{program}$)	0.915	0.884	**0.921**	**0.925**	0.845	0.835	**0.685**	-	**0.814**
Our Model(700k)	0.915	0.883	**0.919**	**0.923**	0.843	0.833	**0.685**	-	**0.812**

Table 5.2: Current Results

The work here outperforms the RMI model significantly and with a bigger margin. When it is compared with IEP-REF model, there are two sets of experiments which needs to be considered here. When ground truth programs are used for constructing the neural module network, this work's model performs almost the same (**0.814**) in terms of overall IoU as of IEP-REF model (**0.816**). This work however outperforms the existing SOTA for 2-Relate, 3-Relate and Same type of referring expressions. On the other hand, when predicted programs are used for constructing the neural module network, this work's model outperforms (**0.812**) IEP-REF model (**0.806**) in terms of overall IoU as evident in the results table.

In table 5.3, It's empirically shown that image features extracted from ResNet-152 model help more in achieving better IoU compared to ResNet-101 model. It basically proves the assumption in the previous section that ResNet-152 features might be a good choice to go with.

In table 5.4 the results from various experiments are displayed which basically corresponds to architectural changes based on the intuitions mentioned in this work. In these experiments, different combinations are tried out and it is attempted to see how these modules affect the performance. In this table, for all the experiments

Table 5.3: Experiment with Different Image Features

Image Features	IoU
ResNet-101	0.797
ResNet-152	0.814

Table 5.4: Experiments with Modules

Module	Relate (#1)	Relate (#2)	Same (Concatenation)	Same (Subtraction)	Filter Attention Module	IoU
Exp. 1	**Yes**	No	**Yes**	No	No	0.800
Exp. 2	**Yes**	No	No	**Yes**	No	0.804
Exp. 3	**No**	**Yes**	**Yes**	**No**	**No**	**0.814**
Exp. 4	No	**Yes**	No	**Yes**	No	0.802
Exp. 5	No	**Yes**	**Yes**	No	**Yes**	0.809

hyperparameters such as learning rate, batch size, etc remain fixed. Only the modules with "Yes" are changed for that particular experiment. All these experiments were performed for ground truth programs and image features were extracted from ResNet-152 model.

Chapter 6

ANALYSIS & DISCUSSION

6.1 Step-by-step Process for Visual Reasoning

In the previous chapter, the primitive operation modules are explained which are used for constructing neural module network. The property of these modules which makes the reasoning of the model understandable is the use of Attention maps. The approach which is applied here is inspired directly by the work of (Mascharka *et al.*, 2018). From the understanding of the previous chapter, it is known that some modules focus only on the local features in the image. For example, *Attention* module pays attention on the distinct objects and their properties. Modules like *Relate* rely on the global context in order to perform relate operation.

For referring expression task, to identify the intended object, the visual reasoning process requires localizing objects based on their distinguishable visible property (such as color, shape, visibility, material, etc.) as well as given spatial information. These modules produce a one-dimensional attention mask which explicitly demarcates the relevant spatial regions in the image. These one-dimensional attention masks are passed throughout between the modules of the neural module network. Using this technique makes sure that understanding and visual reasoning process of the model is easy to understand. In this section, one end-to-end example is explained to understand the whole process of neural module network construction along with the visual reasoning process step-by-step.

Below is an example where a referring expression x and an image w are given:

x : 'The rubber object(s) that are in front of the first one of the cube from front and left of the second one of the tiny rubber cube from left'

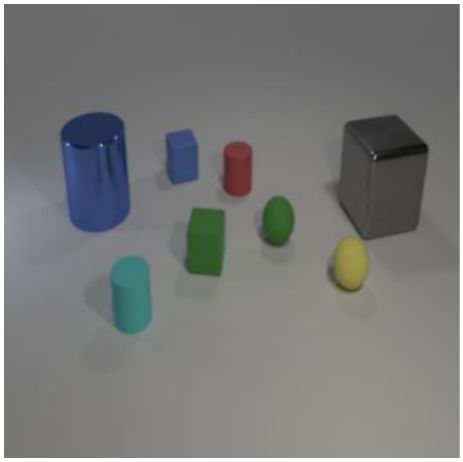

Figure 6.1: Referring Environment Image

In the beginning, the referring expression is passed to the program generator which based on a function catalog, produces a functional program. Based on this functional program, a neural module network is constructed using primitive operation modules as explained in the previous chapter. In this case, modules are used based on the following functions.

- scene
- filter_shape[cube]
- filter_ordinal[the first one of the,from front]
- relate[front]
- scene
- filter_size[small]

- filter_material[rubber]
- filter_shape[cube]
- filter_ordinal[the second one of the,from left]
- relate[left]
- intersect
- filter_material[rubber]

Based on these functions, a neural module network is constructed. Now on another front, image features are extracted from ResNet-101 model which is pre-trained on ImageNet. These features are fed through a simple convolutional block which is called 'stem' (similar to work of (Johnson *et al.*, 2017b)). The stem component transformes the high dimensional image features into lower-dimensional features which are further used by the model. These features are provided to each module in the neural module network and this is how it is ensured that image features are readily available to each module and no information is lost while performing visual reasoning.

This neural module network processes these image features through each module while generating attention masks. Once these modules are done processing, the one-dimensional feature map is passed through a module that converts it into 128 channel feature map. In the end, this feature map is passed to a segmentation module that predicts the image with segmentation mask. This segmentation mask indicates the referred object. In figure 6.2, the reasoning process is explained with attention images.

As explained before, the referring expression 'The rubber object(s) that are in front of the first one of the cube from front and left of the second one of the tiny rubber cube from left' is decomposed into various primitive operations.

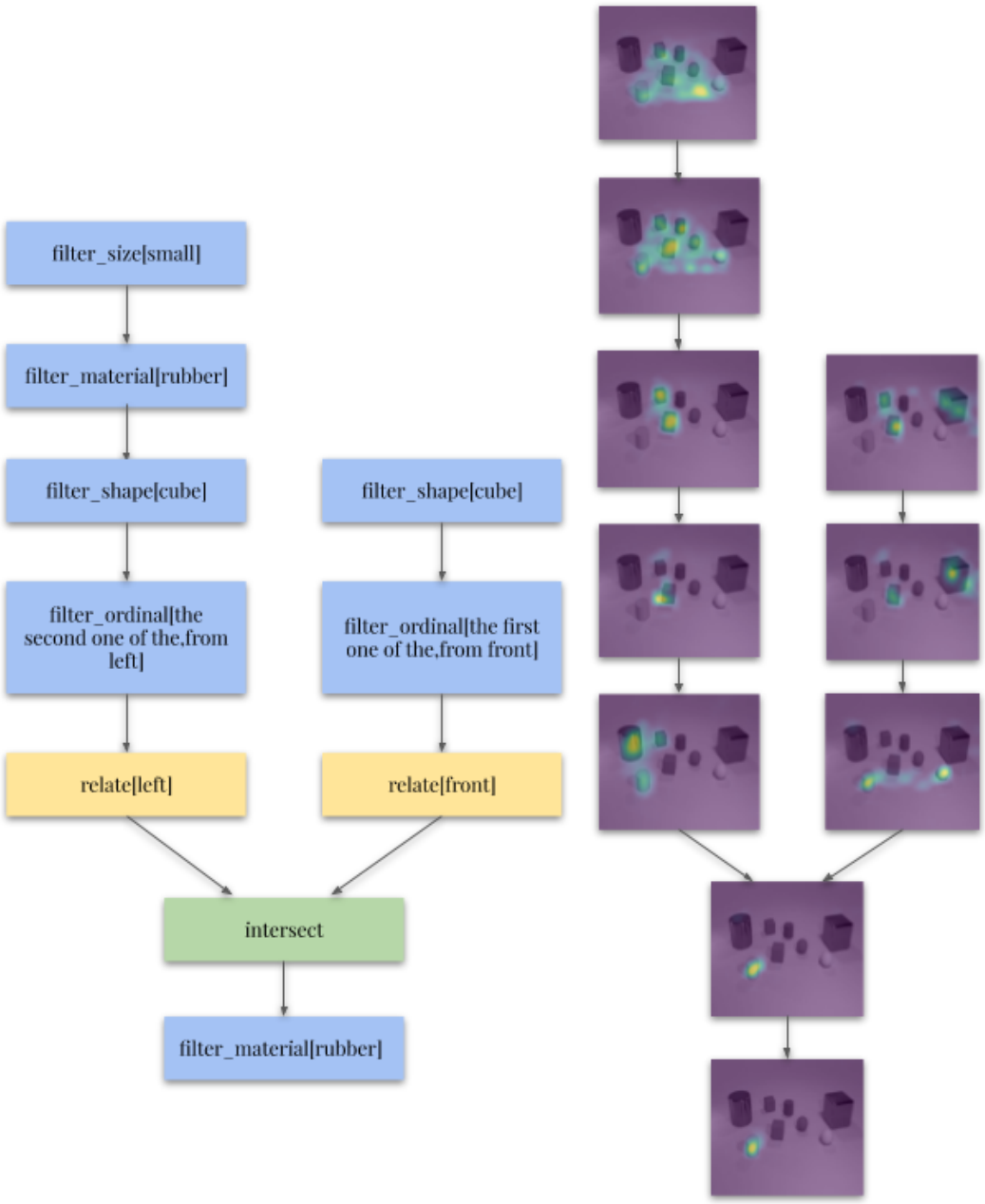

Figure 6.2: Visual Reasoning for Referring Expression Comprehension

There are two parts of this expression which are joined using the intersect module. In the first part of the reasoning tree, *filter_size[small]* based module finds all the small objects, then *filter_material[rubber]* based module finds small objects which

are of rubber material, then *filter_shape[cube]* based module filers out small rubber cubes, then *filter_ordinal[the second one of the,from left]* module finds the cube which is second one of the small rubber cubes from left. Now *relate[left]* based module encodes spatial information about the region which are left of that small rubber cube. In the second part of the reasoning tree, *filter_shape[cube]* based module finds all the cubes in the image, then *filter_ordinal[the first one of the,from front]* based module finds the cube which is the first one from the front. Now *relate[front]* based module encodes the spatial information about the region which is in front of the cube. Further, the *intersect* module takes these attention masks from these two chains and produces an attention mask which is the intersection of these two previous outputs. In this case, the focus can be seen on small cylinder. In the end, *filter_material[rubber]* based module takes that as input and focuses on the cylinder which is in the front only.

As it is clear throughout the reasoning process that all the relevant regions and objects are highlighted. This makes the reasoning intuitively understandable.

6.2 Ablation Studies

The basic idea behind ablation studies(Meyes *et al.*, 2019) is to check how a model performs if a particular component is removed from the overall model. As this approach is modular network based, it is imperative to perform ablation studies for this work. One would gain insights about the structure and organization of the represented knowledge within the network, providing transparency and understanding of the network's behavior. This section further experiments and investigates this work in terms of ablation studies.

6.2.1 *Removal of Unique Module*

Unique module tries to make sure that a single object is returned if it is passed with a set of objects and that set of object is singleton. If this module is removed, the model should have some problem in discerning the intended object because this module kind of helps in asserting that there is only one object which is being referred here (from the previous module).

While experimenting, Unique module is removed and the model is trained again, and it is observed that there is very some drop in the average IoU value. While keeping the whole model intact, this work achieves the IoU of 0.814. However, after the removal of this component, the value of IoU drops to 0.785. This is a decrease of 3.5%. Hence it can be concluded here that removal of unique module actually affects the performance hence unique module plays an important role in performing complex referring expression comprehension task where it needs to make sure that only one object is being referred from previous module.

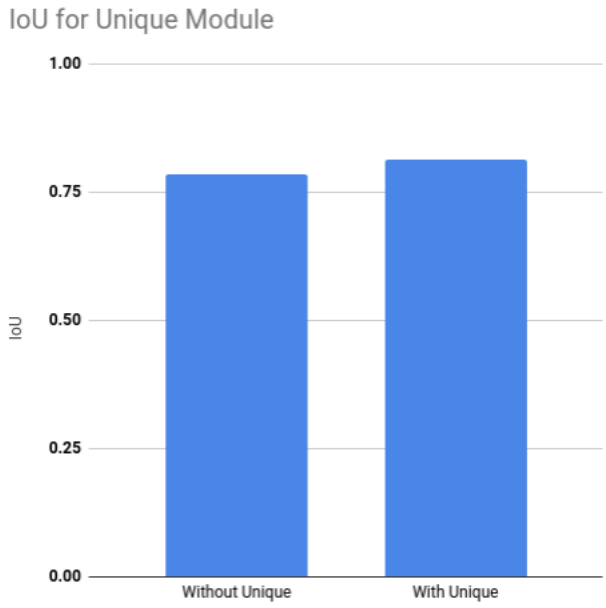

Figure 6.3: Comparison of IoUs for Inclusion and Exclusion of Unique module

6.2.2 *Removal of Dilated Convolution Layer*

The logic behind using dilated convolution is that it provides a larger receptive area for the kernel and hence encoding the global context more easily. If the dilated convolution layer which was added to improve on the global context, is removed then the model should have an overall performance issue.

While experimenting, the extra dilated convolution layer is removed and the model is trained again, and it is observed that there is very some drop in the average IoU value. While keeping the whole model intact, this work achieves the IoU of 0.814. However, after the removal of this component, the value of IoU drops to 0.800. This is a decrease of 1.7%. Hence it can be concluded here that removal of the layer affects the performance little bit however it is required to provide a better global context.

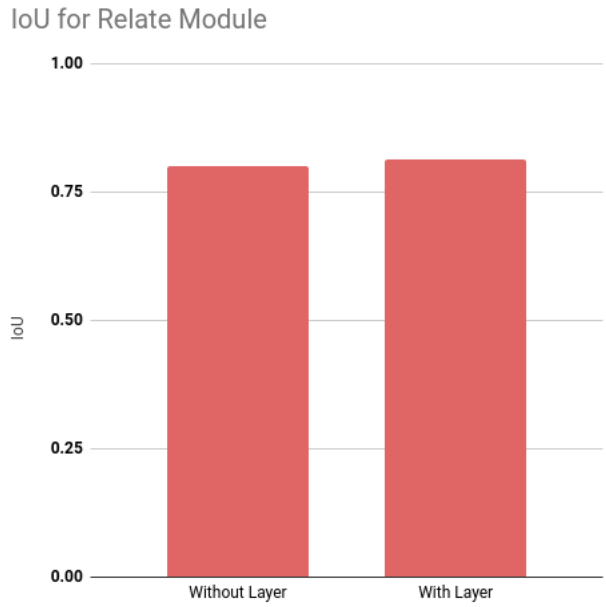

Figure 6.4: Comparison of IoUs for Inclusion and Exclusion of Layer in Relate Module

6.3 Error Analysis

From figure 6.5 shows an example for this work where a referring expression and an image produce an image with the intended object. This section looks into the examples which this system does not get right or fails to provide correct object.

From table 5.2, it's evident that "Same" module is the hardest module to learn. It contributes to the IoU of only 0.685 which is pretty less compared to other modules. This result suggests that same-attribute concept based module is hard to learn.

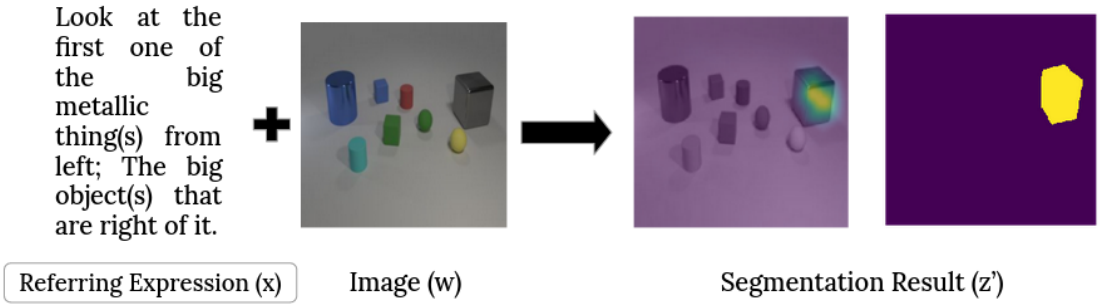

Figure 6.5: Example of This work

Below are given two examples to consider this theory.

Here referring expression r_1 ="Any other things that have the same material as the fifth one of the small thing(s) from front" and image are given in figure 6.6. The

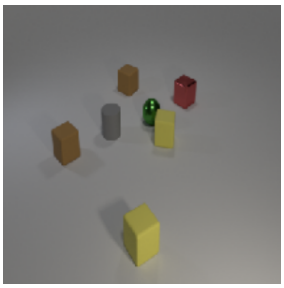

Figure 6.6: Image 1

reasoning process for this referring expression and image can be seen in figure 6.7.

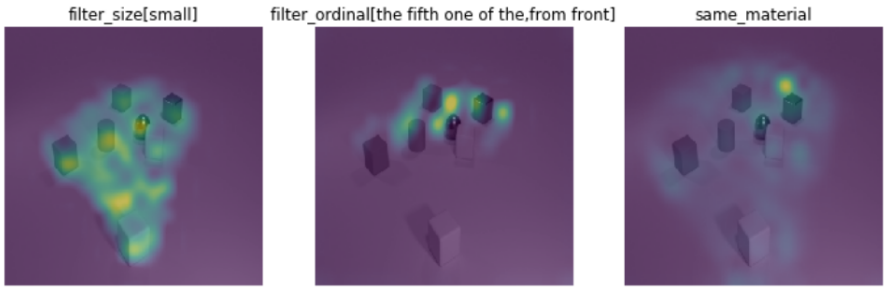

Figure 6.7: Reasoning Process for Image 1

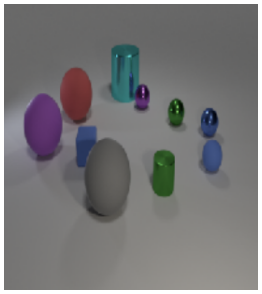

Figure 6.8: Image 2

Here one can see that *filer_ordinal* and *same* module does not perform their respective task that efficiently. As mentioned, these concepts are hard to grasp or learn. This might be the reason behind this performance.

Here referring expression r_2 = "Any other purple object(s) of the same size as the third one of the sphere(s) from right" and image are given in figure 6.8. The reasoning process for this referring expression and image can be seen in figure 6.9.

Both of these two examples clearly suggest that *Same* and *filter_ordinal* modules are actually difficult concepts to learn and are the main source of error in this work.

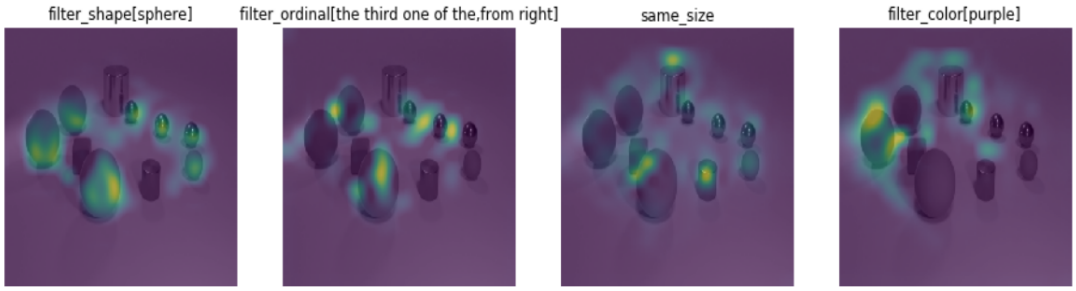

Figure 6.9: Reasoning Process for Image 2

6.4 Analysis For Referring Expressions With Variations And Non-Reasoning Phrases

While dealing with Referring Expressions, one can be sure that they are complex and compositional in nature. Natural Languages are generally diverse and rich in terms of synonyms and the way they are formed. For example, one can consider following example:

$x =$ Small green cubes which are on the left side of the metal spheres.

It's human habit to state the same sentence using different words and manners for simple reason of people are different. This referring expression can be paraphrased using synonyms and different words along with varying structure. Some of the paraphrased referring expressions are following:

$a =$ ***Tiny*** green cubes which are on the left side of the metal spheres.

$b =$ ***The*** small green cubes which are ***to the left*** of the metal spheres.

$c =$ small green cubes which are ***to the left*** of the ***shiny*** spheres.

Above mentioned referring expressions are some of the permutations of the same referring expression which one could compose using different words and synonyms. Here it can be seen that depending on the synonyms, determinants and sentence structure, different referring expressions can refer to the same objects. In this analysis, the robustness of the model against these variations would be tested.

For this experiment, adjectives such as ***small*** and ***large*** would be interchanged with ***tiny*** and ***big*** respectively. The use of determinants may be removed or retained. Going the same way, ***object*** and ***thing*** can be interchanged. A ***cube*** can be called a ***block***. Similarly, one can refer a ***sphere*** as a ***ball***. The material property of an

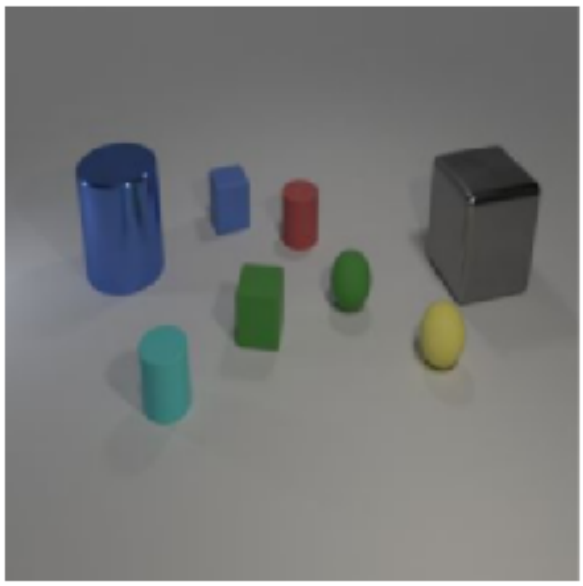

Figure 6.10: Image (w)

object can also be referred in multiple ways. The metal property can be referred by ***metal***, ***shiny***, and ***metallic*** in the referring expressions. Similarly, ***rubber*** objects can be referred as ***matte*** objects. Even the small non-reasoning phrases like ***Look at***, ***Find*** and words like ***it*** and ***that*** can be changed without altering the core meaning of the referring expression. The spatial relationship also can take different forms. For example, left spatial relationship can be denoted by following phrases,

- left of
- to the left of
- on the left side of

The image w is given in figure 6.10. The corresponding referring expression in this case is following:

x = *"Look at the first one of the large metal thing(s) from left; The large object(s) that are right of it."*

The functional program for this referring expression is generated through Program Generator. The functional program is following:

- filter_size[large]
- filter_material[metal]
- filter_ordinal[the first one of the,from left]
- unique
- relate[right]
- filter_size[large]

As explained previously, the above referring expression can be represented in multiple forms depending on synonyms and different words. Some of those are following:

x_1 = Look at the first one of the ***big shiny*** thing(s) from left; The ***big*** object(s) that are right of ***that***.

x_2 = Look at the first one of the large ***metallic*** thing(s) from left; The large object(s) that are ***on the right side of*** it.

x_3 = Look at the first one of the ***big shiny object***(s) from left; The large object(s) that are **to the right of** it.

x_4 = ***Find*** the first one of the large ***shiny*** thing(s) from left; The ***big*** object(s) that are right of ***that***.

x_5 = ***Find*** the first one of the ***big metallic*** thing(s) from left; The large object(s) that are ***on the right side of*** it.

x_6 = ***Find*** the first one of the ***big*** metal ***object***(s) from left; The ***big*** object(s) that are right of it.

Table 6.1: Functional Programs Generated From Program Generator

	x_1	x_3	x_5
1	filter_size[large]	filter_size[large]	filter_size[large]
2	filter_material[metal]	filter_material[metal]	filter_material[metal]
3	filter_ordinal[the first one of the,from left]	filter_ordinal[the first one of the,from left]	filter_ordinal[the first one of the,from left]
4	unique	unique	unique
5	relate[right]	relate[right]	relate[right]
6	filter_size[large]	filter_size[large]	filter_size[large]

x_7 = Look at the first one of the ***large metallic object***(s) from left; The ***big thing***(s) that are right of ***that***.

Here it can be seen that the referring expression x can be paraphrased in various ways using synonyms, different words and phrases. However all of these variations of referring expression refer to same object in this image (w). When these referring expressions are processed through Program Generator, they generate the same functional program for each of them. These functional programs are shown in table 6.1 for some of the above variations.

Here it can be seen that after changing the original referring expression in multiple ways in terms of adjectives, synonyms and non-reasoning phrases (such as *Look at* and *Find*), the model still constructs same neural module network to refer the intended objects. It suggests that this model can deal with small changes (in form of synonyms and adjectives) to the referring expressions and still produce the same result.

Chapter 7

CONCLUSION

7.1 Summary

Using a monolithic structure to solve the problems in deep learning has been prevalent for a long time. With the advent of a modular approach for deep learning,a there is a visible transformation in the way problems are approached now. This thesis works in the direction of the modular approach for deep learning.

This work leverages the composability of language with the help of neural module network approach to tackle a very challenging problem in Vision-Language domain and addresses the Referring Expression Comprehension task. This task basically involves identifying a particular object in a given image with the help of a natural language statement which is known as Referring Expression.

This work applies neural module network approach on CLEVR-Ref+ dataset(Liu *et al.*, 2019) which is a synthetic dataset. The natural language component is processed by an LSTM based Program Generator(Johnson *et al.*, 2017b) which helps in building the neural module network. On the other hand, ResNet(He *et al.*, 2015) models are used to extract image features. Executing image features along with this neural module network followed by a segmentation module provides the intended object in the image.

Chapter 1 introduces the problem task of Referring expression comprehension at hand and outlines the contribution of this work. Chapter 2 explains the background work which is required to understand the basic building blocks and techniques related to this task (such as Neural networks, RNN, CNN, etc.). Chapter 3 talks about the

foundation work of the Neural module network along with the dataset on which the experiments were performed. This chapter also discusses previous related work and successful approaches. Chapter 4 explains the design decisions taken in this work and outlines the overall architecture for referring expression comprehension task. It also explains all the operation-specific modules which are required to build the neural module network. Chapter 5 lays out the experimental setup along with dataset, evaluation metric. It also presents the results in this work and shows that it achieves state-of-the-art results for predicted programs and comparable results for ground truth programs. Chapter 6 explains the visual reasoning process for the task with heatmap and also provides information about the ablation studies and error analysis for this work. The visual reasoning process is quite intuitive for the end-user and easily understandable.

7.2 Limitations and Future Recommendations

Chapter 6 explains some of the limitations of this work. For any given referring expression and image, a neural module network is constructed from a set of pre-defined modules. If there are more attributes added to the dataset then it is required to design separate modules for them.

Also, some existing modules such as *same* and *filter_ordinal* are not efficient as other modules are. Same-attribute relationship and ordinality are the properties that are hard to learn here as evident from error analysis and quantitative analysis.

Because of the modular approach, in this work, weak supervision is performed. A potential approach for improving further can be thought in terms of intermediate supervision. Since CLEVR-Ref+ dataset provides coordinate information of objects, that can be used for intermediate supervision.

This work uses CrossEntropy as loss function and then evaluation is performed for IoU metric. (Rezatofighi *et al.*, 2019) introduced a generalized version of IoU as both a new loss and a new metric. One recommendation would be to use GIoU as metric and experiment with that.

REFERENCES

Andreas, J., M. Rohrbach, T. Darrell and D. Klein, "Learning to compose neural networks for question answering", arXiv preprint arXiv:1601.01705 (2016a).

Andreas, J., M. Rohrbach, T. Darrell and D. Klein, "Neural module networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 39–48 (2016b).

Bengio, Y., "Learning deep architectures for ai", Foundations and trends® in Machine Learning , 1, 1–127 (2009).

Bengio, Y., A. Courville and P. Vincent, "Representation learning: A review and new perspectives", IEEE transactions on pattern analysis and machine intelligence , 8, 1798–1828 (2013).

Blender, "Blender - a 3d modelling and rendering package", URL `http://www.blender.org` (2018).

Cho, K., B. Van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", arXiv preprint arXiv:1409.1259 (2014).

Cirik, V., L.-P. Morency and T. Berg-Kirkpatrick, "Visual referring expression recognition: What do systems actually learn?", arXiv preprint arXiv:1805.11818 (2018).

Dong, C., C. C. Loy, K. He and X. Tang, "Image super-resolution using deep convolutional networks", IEEE transactions on pattern analysis and machine intelligence , 2, 295–307 (2015).

Dyer, C., M. Ballesteros, W. Ling, A. Matthews and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory", arXiv preprint arXiv:1505.08075 (2015).

Fei-Fei Li, D. X., Ranjay Krishna, "Cs231n: Convolutional neural networks for visual recognition", URL `https://cs231n.github.io/neural-networks-1/` (2020).

Gao, H., J. Mao, J. Zhou, Z. Huang, L. Wang and W. Xu, "Are you talking to a machine? dataset and methods for multilingual image question", in "Advances in neural information processing systems", pp. 2296–2304 (2015).

Goodrich, M. A. and A. C. Schultz, "Human-robot interaction: a survey", Foundations and trends in human-computer interaction , 3, 203–275 (2007).

Hassoun, M. H. *et al.*, "Fundamentals of artificial neural networks", (1995).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", CoRR (2015).

Hinton, G. E., "Deep belief networks", Scholarpedia , 5, 5947 (2009).

Hochreiter, S. and J. Schmidhuber, "Long short-term memory", Neural Comput. , 8, 1735–1780, URL https://doi.org/10.1162/neco.1997.9.8.1735 (1997).

Hu, R., J. Andreas, M. Rohrbach, T. Darrell and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 804–813 (2017a).

Hu, R., M. Rohrbach, J. Andreas, T. Darrell and K. Saenko, "Modeling relationships in referential expressions with compositional modular networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1115–1124 (2017b).

Hu, R., M. Rohrbach and T. Darrell, "Segmentation from natural language expressions", in "European Conference on Computer Vision", pp. 108–124 (Springer, 2016a).

Hu, R., H. Xu, M. Rohrbach, J. Feng, K. Saenko and T. Darrell, "Natural language object retrieval", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4555–4564 (2016b).

Jaccard, P., "Etude comparative de la distribution florale dans une portion des alpes et des jura", (1901).

Johnson, J., B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2901–2910 (2017a).

Johnson, J., B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick and R. Girshick, "Inferring and executing programs for visual reasoning", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 2989–2998 (2017b).

Kazemzadeh, S., V. Ordonez, M. Matten and T. Berg, "Referitgame: Referring to objects in photographs of natural scenes", in "Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)", pp. 787–798 (2014).

Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980 (2014).

Klein, D. and C. D. Manning, "Accurate unlexicalized parsing", in "Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1", pp. 423–430 (Association for Computational Linguistics, 2003).

Kosub, S., "A note on the triangle inequality for the jaccard distance", Pattern Recognit. Lett. pp. 36–38 (2019).

Krishna, R., Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations", International Journal of Computer Vision , 1, 32–73 (2017).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in "Advances in neural information processing systems", pp. 1097–1105 (2012).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE , 11, 2278–2324 (1998).

Li, R., K. Li, Y.-C. Kuo, M. Shu, X. Qi, X. Shen and J. Jia, "Referring image segmentation via recurrent refinement networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 5745–5753 (2018).

Liu, C., Z. Lin, X. Shen, J. Yang, X. Lu and A. Yuille, "Recurrent multimodal interaction for referring image segmentation", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 1271–1280 (2017).

Liu, R., C. Liu, Y. Bai and A. L. Yuille, "Clevr-ref+: Diagnosing visual reasoning with referring expressions", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4185–4194 (2019).

Luo, R. and G. Shakhnarovich, "Comprehension-guided referring expressions", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 7102–7111 (2017).

Malinowski, M. and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input", in "Advances in neural information processing systems", pp. 1682–1690 (2014).

Mao, J., J. Huang, A. Toshev, O. Camburu, A. L. Yuille and K. Murphy, "Generation and comprehension of unambiguous object descriptions", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 11–20 (2016).

Margffoy-Tuay, E., J. C. Pérez, E. Botero and P. Arbeláez, "Dynamic multimodal instance segmentation guided by natural language queries", in "Proceedings of the European Conference on Computer Vision (ECCV)", pp. 630–645 (2018).

Mascharka, D., P. Tran, R. Soklaski and A. Majumdar, "Transparency by design: Closing the gap between performance and interpretability in visual reasoning", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 4942–4950 (2018).

McCulloch, W. S. and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", The bulletin of mathematical biophysics , 4, 115–133 (1943).

Meyes, R., M. Lu, C. W. de Puiseau and T. Meisen, "Ablation studies in artificial neural networks", arXiv preprint arXiv:1901.08644 (2019).

Mikolov, T., M. Karafiát, L. Burget, J. Černockỳ and S. Khudanpur, "Recurrent neural network based language model", in "Eleventh annual conference of the international speech communication association", (2010).

Nagaraja, V. K., V. I. Morariu and L. S. Davis, "Modeling context between objects for referring expression understanding", in "European Conference on Computer Vision", pp. 792–807 (Springer, 2016).

Pan, S. J. and Q. Yang, "A survey on transfer learning", IEEE Transactions on Knowledge and Data Engineering pp. 1345–1359 (2010).

Ray, A., G. Christie, M. Bansal, D. Batra and D. Parikh, "Question relevance in vqa: identifying non-visual and false-premise questions", arXiv preprint arXiv:1606.06622 (2016).

Ren, M., R. Kiros and R. Zemel, "Exploring models and data for image question answering", in "Advances in neural information processing systems", pp. 2953–2961 (2015a).

Ren, S., K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in "Advances in neural information processing systems", pp. 91–99 (2015b).

Rezatofighi, H., N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression", (2019).

Rohrbach, A., M. Rohrbach, R. Hu, T. Darrell and B. Schiele, "Grounding of textual phrases in images by reconstruction", in "European Conference on Computer Vision", pp. 817–834 (Springer, 2016).

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge", International journal of computer vision , 3, 211–252 (2015).

Schmidhuber, J., "Deep learning in neural networks: An overview", Neural networks pp. 85–117 (2015).

Searle, J. R., "Speech acts: An essay in the philosophy of language.", (1969).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556 (2014).

Socher, R., J. Bauer, C. D. Manning and A. Y. Ng, "Parsing with compositional vector grammars", in "Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)", pp. 455–465 (2013).

Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", in "Advances in neural information processing systems", pp. 3104–3112 (2014).

Sutton, R. S., A. G. Barto *et al.*, "Introduction to reinforcement learning", (1998).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions. arxiv 2014", arXiv preprint arXiv:1409.4842 (2014).

Weston, J., S. Chopra and A. Bordes, "Memory networks", arXiv preprint arXiv:1410.3916 (2014).

Winograd, T., "Understanding natural language", (1972).

Yu, L., Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal and T. L. Berg, "Mattnet: Modular attention network for referring expression comprehension", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1307–1315 (2018).

Yu, L., E. Park, A. C. Berg and T. L. Berg, "Visual madlibs: Fill in the blank image generation and question answering", arXiv preprint arXiv:1506.00278 (2015).

Yu, L., P. Poirson, S. Yang, A. C. Berg and T. L. Berg, "Modeling context in referring expressions", in "European Conference on Computer Vision", pp. 69–85 (Springer, 2016).

Yu, L., H. Tan, M. Bansal and T. L. Berg, "A joint speaker-listener-reinforcer model for referring expressions", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 7282–7290 (2017).

Zhu, Y., O. Groth, M. Bernstein and L. Fei-Fei, "Visual7w: Grounded question answering in images", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 4995–5004 (2016).