Learning to quantify from images

Ionut-Teodor Sorodoc

Faculty of Information and Communication Technology

Department of Intelligent Computer Systems

UNIVERSITY OF MALTA

December, 2016

Submitted in partial fulfilment of the requirements for the degree of

Master of Science in Human Language Science and Technology (HLST)

FACULTY OF
INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITY OF MALTA

Declaration

Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta). The dissertation submitted is my own work, except where acknowledged and referenced.

I understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Student Name:	Ionut-Teodor Sorodoc
Course Code:	CSA5310 HLST Dissertation
Title of work:	Learning to quantify from images

Signature of Student:

Date:

Supervisor(s)

Raffaella Bernardi
Center for Mind/Brain Sciences
University of Trento
,
Aurelie Herbelot
Center for Mind/Brain Sciences
University of Trento
and
Michael Rosner
Department of Intelligent Computer Systems
University of Malta

Local Advisor

Michael Rosner
Department of Intelligent Computer Systems
University of Malta

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Raffaella Bernardi for her support. The research for this thesis could not have been carried out without her guidance and encouragement.

I would also like to thank my co-supervisors Aurelie Herbelot and Michael Rosner for their feedback and advice.

I am grateful for the opportunity that The European Masters Program in Language and Communication Technologies gave to me. Thanks to them, I had the chance to learn from the best professors and to have guidance from the best researchers in three of the most beautiful places in the world.

I want to thank all my colleagues and friends from University of Trento for their invaluable support and for creating some of the best moments in my life.

Last but not least, I would like to thank my parents for their constant love and support.

Abstract

Combining visual understanding and natural language is one of the most important reseach problems in artificial intelligence. Recent research involving image captioning has shown very good results in jointly learning from images and text using models such as convolutional neural networks for object recognition and word embeddings extracted from huge amounts of text. Visual Question Answering (VQA) adds an extra layer of complexity because it has to pay attention to details, not just to give a vague description and it is a challenging testbed for multimodal systems. On one hand, neural networks have been shown to obtain excellent results on the task, using relatively simple systems which learn correlations between linguistic and visual features. On the other hand, the same systems have struggled with questions that require deeper reasoning. In particular, 'number questions' have been associated with low performance, and have only been studied superficially.

In this thesis, we focus on questions which may be answered with a quantifier (e.g., *Which proportion of dogs are black? Some/most/all of them*, etc.) relative to an image. We show that in order to learn to quantify, a multimodal model has to obtain a genuine understanding of linguistic and visual inputs and of their interaction. We build a dataset which is suitable for this task and we propose a model that extracts a fuzzy representation of the set of the queried objects (e.g., *dogs*) and of the queried property in relation to that set (e.g., *black* with respect to *dogs*), outputting the appropriate quantifier for that relation. Our model outperforms a state-of-the-art VQA system on this challenging task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer Vision and Computational Linguistics are currently in a golden age. New multimodal models are being built for challenging tasks which were still out of reach a few years ago and can now be integrated in many end-user applications. Visual Question Answering (VQAs) and caption generation are examples of such tasks which have seen dramatic performance improvements. Beside having direct applications in the real world, where visual media is extremely prominent, their performance measures to what extent language and vision models capture meaning, giving us important theoretical insights.

The recent improvements observed in VQA applications are due to the new generation of 'deep learning' Neural Network (NN) models, combined with the availability of large image datasets. Such models have shown that even very simple NN structures can capture complex interactions between the features of a data source in a way that outperforms more complex models. For instance, [Zhou et al., 2015a] have recently demonstrated that a simple bag-of-word baseline can achieve state-of-the-art performance on a major VQA dataset [Antol et al., 2015a]. The authors argue that the performance of the model is however only due to the excellent ability of the network to encode correlations, concluding that there is a need to move from 'memorizing correlations' to 'actual reasoning and understanding'. Their work highlights how tricky it can be to evaluate whether a model genuinely grasps the meaning of images and words. In the present paper, we propose a task that we believe will help the community towards achieving this goal.

A Visual Question Answering (VQA) system receives as input an image and a free-form natural language question about the image and it outputs a natural language answer

which will reflect the information from the image relative to the question.

In the VQA task, current evaluations focus on questions about the properties of specific objects (location, color, etc.) while in the caption generation task, it is the entire image which is considered. In both cases, it is clearly possible to learn correlations between the words in the question, the words in the answer, and parts or all of the image. For instance, a picture of a person eating a cake, presented to the NN with the question *what is she eating?* will reliably activate the association between the verb *eat* and food-related word/visual features, producing the correct answer. In order to test a model beyond this simple mechanism, we need a task where a particular association between query and image does not always result in the same answer.

"Number questions" have only been marginally studied in the literature, and the performance of VQA systems on those questions has proved to be rather poor [Ren et al., 2015, Antol et al., 2015a]. Further, the literature has focused nearly exclusively on modelling exact cardinals, giving a biased account of the quantification phenomenon. In this paper, we investigate a new type of question, requiring some understanding of generalised quantifiers (*few, most, all*, etc). We are interested in such questions because they cannot be answered by simply detecting the area of an image that relates to the query. For instance, the question *which proportion of dogs are black?* requires more than the identification of dog and black features in the image: the network must be able to reason about the relationship between those features and to generate one of several potentially similar quantifier words. The ability to identify the members of the set and their shared properties demands some form of deeper reasoning which, we claim, cannot be achieved by simple memorization.

In what follows, we look at the VQA task as a 'fill in the blank' problem and turn the question *which proportion of dogs are black?* into the query: ___ *dogs are black*.

The possible answers are selected from a range of linguistic quantifiers, namely *no, few, some, most*, and *all*. In order to assign the correct quantifier, the model must be able to focus on the relevant objects (the quantifier's **restrictor**) and to quantify which objects in this restricted domain have the queried property (the quantifier's **scope**). We show that a bag-of-word model is not sufficient for such a task, and propose an alternative,

linguistically motivated NN model, which outperforms the state-of-the-art VQA model of [Zhou et al., 2015a].

Chapter 2

Related Work

Visual question answering involves a combination of Computer Vision and Natural Language Processing and this topic received more attention since deep learning has showed very good results. Both these fields are very challenging for different reasons.

In Computational Vision, if we have a small image with 256*256 resolution and with 256 pixel values, we have 2^{524288} possible images which is an incredible amount of possibilities. We have visual object variations (different viewpoints, scales, deformations), intra-class variation, inter-class overlap. All these factors show that Computational Vision is a very challenging field.

Natural Language Processing(NLP) is an extremely complex task: it has to deal with synonymy, ambiguity, it is very high dimensional. If we consider only English, we have around 150000 words and we have to learn 150000 classifiers with a lot of sparse data.

It is impressive the amount of great results considering all the arguments presented above, but VQA implies good models for both fields and also a good understanding of the bridge between them.

2.1 Neural networks for Visual Question Answering.

A Visual Question Answering (VQA) system takes as input an image and a natural language question about the image and generates a natural language answer as output. This type of system requires different artifficial inteligence capabilities: object detection, detail recognition, knowledge and commonsense reasoning.

Most of the deep learning innovations are because of three reasons: structured data

availability, software creation and adaptation, and hardware evolution. The improvements in all of these areas lead to the impressive results that we have nowadays.

The first step in the direction of VQA is exploring image captioning and description [Karpathy and Fei-Fei, 2015], [Xu et al., 2015], [Vinyals et al., 2015], [Kiros et al., 2014]. For this task, the system receives a visual input and it has to produce a textual description as output. This means that it has to map the visual information into a linguistic answer which is the first step for a good VQA system.

A common approach which works very well for image description is presented in [Vinyals et al., 2015]. They apply a Convolutional Neural Network(CNN) over the input. The result of the CNN is a vector which will contain the visual features of the image. In order to produce a textual answer, the output of the CNN is connected to a Sequential Neural Network like Recurrent Neural Network (RNN) or Long Short Term Memory (LSTM). These networks are considered state of the art for language modelling and they are the best fit to produce sentences which will reflect the information from the image.

Another important technique applied in state-of-the-art image captioning systems is an attention module over the input. [Xu et al., 2015] adds a step to the previous models which focuses in specific regions of the images which contain the important information to formulate a correct answer. The results increased significantly with the addition of the new module.

A step forward to challenge the artificial neural networks is Visual Question Answering. This adds complexity to image description because it receives two types of input: textual and visual, and they are in different spaces. Because of the rich information needed for this task, the first challenge was developing and finding suitable datasets.

[Malinowski and Fritz, 2014] present the necessity of a visual turing test in order to develop performant systems, and they introduce DAQUAR dataset which is a collection of real images and a big variation of questions and answers. They also expose the biggest challenges of VQA: scalability, concept ambiguity, attributes, ambiguity in reference resolution, and common sense knowledge. The dataset created by [Malinowski and Fritz, 2014] opened the ways to different systems and models to be applied in visual question answering, but it also encouraged the development of other

datasets which will challenge and increase the complexity of this task.

[Lin et al., 2014a] developed one of the most used datasets in computational vision and multimodal applications: MS-Coco. This dataset is not specifically built for VQA, but it makes available a lot of annotated data: 300000 images with 2 million of instances and each image has 5 associated captions. The complexity and variation of the dataset challenges the research in the field, but it helps also to generate other specific tasks and datasets. [Antol et al., 2015b] uses a part of the information from MS-Coco and it generates a VQA specific dataset with open ended questions which require a higher level of understanding. Also, an interesting approach followed by [Antol et al., 2015b] is to add abstract images which will show if there is a difference of the results if you analyze different spaces.

An interesting approach for VQA is presented in [Xiong et al., 2016]: Dynamic memory networks. They show that these networks can be applied on different tasks: VQA, textual question answering, sentiment analysis, pos-tagging. Their claim is that a system able to solve general question answering is one of the most important steps now because most of AI/NLP tasks can be transformed into QA tasks. Also, their model uses a module called episodic memory which is a combination of an attention mechanism and a memory update mechanism. The attention module is responsible for describing the context and the memory update module is responsible to change the information from the memory based on the context and previous memories. This design simulates on a simple level how the human brain is processing this task which is following the claim that the ultimate goal is to understand and replicate the brain mechanisms.

Even though the philosophy of creating a universal QA system is very appealing and it has its justification, I consider that there are specific tasks which can not be incorporated in an universal system. An example which is very different to the others is answering questions which involve counting. When you have to answer questions which involve counting, you can not focus only in a specific area of the input, but you have to find relations beyond this. [Zhou et al., 2015b] show big differences in results between 'how many' questions and questions that involve 'yes/no' which is caused by the different ways to answer. Their analysis shows that you can answer the later class of questions with a

very simple model, but you need reasoning for 'how many' questions. Also, the task of answering counting questions is challenging at a dataset level because it is harder to offer a big variety of scenarios and answers.

2.2 Computational models of quantifiers

The goal of this thesis is to tackle the task of answering counting questions with a visual input using a quantifier. An important part of the project is the interaction between quantifiers and VQA. The problem of quantification is considered a theoretical subject in comparison with the other parts of the project. A first attempt to go beyond theory in the topic of quantification was made by Van Benthem [van Benthem, 1986] who describes an implementation of logical quantifiers using automatas. Automatas are considered one of the most flexible structures that can implement theoretical logic problems, but they do not work with any linguistic or visual input and they lack any connection with the pragmatic part of the task.

The appearance and evolution of neural networks opened the ways of different approaches for tasks which were treated only from a logical point of view. These new architectures give a possibility to use the inputs in a natural way closing the bridge with the human processing of the same tasks. Because of this, we can look at the task in a different way and use also visual and linguistic information.

From their beginning, computational models were used for object enumeration and [Rajapakse et al., 2005] shows that there is a strong connection between the features used by these models for 'numerosity judgements' and the features which play an important role in the quantification processing at humans. They show that there are three ways of dealing with counting: a fast and precised processing of small groups of less than five objects (this process is called subitizing), a slow process of serial counting for groups of objects with cardinality between 5 and 9, and an estimation process for larger groups of objects. These processes are linked to each other, but they are different from the point of view of processing.

[Dehaene and Changeux, 1993] and [Feigenson et al., 2004] present that there are different intervals in the human life when we learn the three ways of counting presented

above. [Feigenson et al., 2004] show that we develop an approximate number system (ANS) which gives an estimation of a large number of objects before we learn to count precisely and we also tend to use ANS often when we deal with a big group of objects.

A good overview of the differences and similarities between counting and quantification in the human brain is presented in [Hurewitz et al., 2006]. Their hypothesis is that children see the number words as quantificational expressions without a strong connection towards the cardinality of the set. This hypothesis is supported by behavioural experiments where children with age between two and three can not understand cardinality, but they can understand quantification like *some* and *many*. These experiments show not only that we acquire quantification before counting, but also that we use our knowledge of the semantics of the quantifiers to create the representations of the number semantics. From another perspective, [Olm et al., 2014] present fMRI studies that show the regions of the brain used for quantification and counting. There are clear differences between these two processes: the number knowledge is processed using the parietal lobe and processing quantifiers prompt activations in the prefrontal cortex which plays an important role in simple decision making according to [Troiani et al., 2009].

There is also a lot of work about quantification from a theoretical point of view ([Montague, 1973], [Cann, 1993], [Peters and Westerståhl, 2006]) which analyses the logical and linguistic aspects of this subject, but the important step in our work is the transition from a theoretical space where quantifiers were described in most of the previous work to a distributional, computational space. [Herbelot and Vecchi, 2015] showed that the transition between these two worlds is possible and that we can generate vector representations with set overlap information which have high quality. Their experiments prove that we can use distributional semantic models to encapsulate theoretical information about quantifiers and that these two spaces are not independent. [Linzen et al., 2016] uses the claims proved by [Herbelot and Vecchi, 2015] and it shows that logical properties like monotonicity or force of quantificational words are embedded in distributional semantic representations with accuracy reaching almost 100%.

As we have seen in this chapter, there is a lot of work done for understanding the quantification process, especially from a theoretical point of view, but the research from

the last few years shows that we have to use also the textual information surrounding this process. We consider that the next step is to use multimodal information for quantification (visual, textual).

Chapter 3

Our Proposal

Our task is learning how to answer to questions which refer to an image with multiple objects. The answer has to describe a vague quantity of objects with specific properties with one of the quantifiers: 'no', 'few', 'some' , 'most', 'all'. As we have seen before, the question is presented in a textual form and the scenario is presented through a visual representation. The information is represented different in these two environments and it is a hard task to close the bridges between them.

The approaches based on deep neural networks developed in the last five years showed very promising results in computational vision in various applications. These results influenced the usage of deep neural networks in computational linguistics and they increased the performances for a big variety of tasks. The precisions of the new models are impressive when we analyse the two different fields, but the next goal is to discover good models which can work with information from both fields and understand the links between vision and language. This would lead to systems closer to the human brain, which processes and understands the information together, even if it comes from different sources.

The multimodal architectures are one of the most important research topics now and they showed their impact in very different areas, as presented by
[Desmond Elliott and Lazaridou, 2016]: disambiguation, coreference resolution, visual lexical entailment, predicting concreteness, metaphor detection, referential grounding.
[Desmond Elliott and Lazaridou, 2016] show that there are different embeddings suited for these architectures and their combination lead to better results than considering them separately. Because of the advancements in both fields, there are different choices for

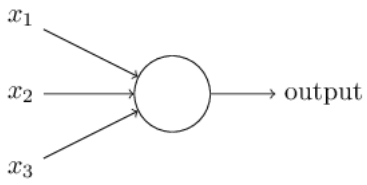

Figure 3.1: An artificial neuron: `http://neuralnetworksanddeeplearning.com/`

embeddings, but they should combine in a natural and intuitive way. This leads to the fact that we have to choose the architectures which encapsulate the information better, not the ones that perform better in their own independent task.

3.1 Background

The models and structures used for this project are different variations of neural networks. The concepts behind neural networks were descibed in the 1950s [Rosenblatt, 1958], but they started to be used in the last decade because the development of more computational power.

The basic concept behind a neural network is an artificial neuron (Figure 3.1). An artificial neuron is a structure which receives a series of inputs $x_1, x_2 \dots x_n$ and outputs a value y calculated using the input. y is calculated applying a non-linear function over the input and each input value is weighted by an associated weight $w_1, w_2, \dots x_n$. For example, the output of a sigmoid neuron is calculated using: $\frac{1}{1+exp(-\sum_j w_j * x_j + b}$. Another important non-linear function used for artificial neurons is the rectified linear unit function (ReLU): $max(0, x)$ which is a very simple and efficient function used mostly in computer vision. The structure of an artificial neuron is simple which gives a lot of flexibility in the design of the networks.

An artificial neuron is a simple structure which is not able to do complex decisions. In order to increase the power of the model, we can use multiple artificial neurons which will constitute an artificial neural network 3.2. As we can see, the neurons are structured in different sets and the neurons in each set do not connect to each other. Each of these sets is considered a layer. As we can see in Figure 3.2, the first layer receives as input

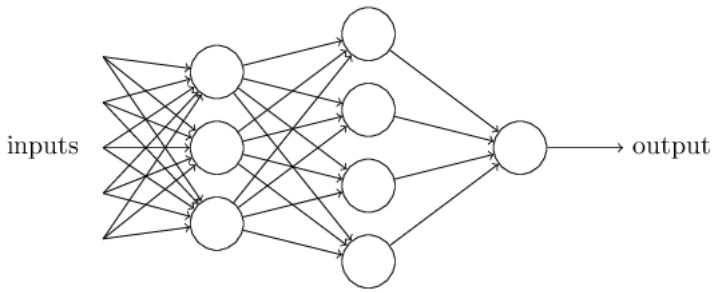

Figure 3.2: A simple neural network: `http://neuralnetworksanddeeplearning.com/`

the raw information which leads to the fact that this layer extracts general information about the input. Because the next layer receives as input the output of the first layer, the patterns captured in the second layer will be more subtle. This is the reason that the networks with more layers have better results.

Another important characteristic of the artificial neural network is the end-to-end property of the network. A neural network is end-to-end if it receives the raw input (the pixels of an image, the raw text) and it outputs the desired answer (sentence, category). This implies that the network does not need any other external information and it can work independently. This is an ideal scenario, but this requires a lot of computational power and it increases the complexity of the model.

One of the most important steps for our task is finding how to learn semantic representations for visual and linguistic information because we want to compensate for the lack of the end-to-end property using state of the art embeddings. Computers discriminate different entities using their semantic representations. The semantic representations are called word embeddings for textual information and image embeddings for visual information.

3.1.1 Distributional Semantics

Because of the importance of the topic, multiple methods to learn semantic information were developed: feature-based models which capture some specific characteristics using rules defined by humans [McRae et al., 2005], semantic networks which build graphs where the vertices represent differenct concepts and the edges are semantic relations between these concepts [Miller, 1995] and semantic spaces, which encode the representations

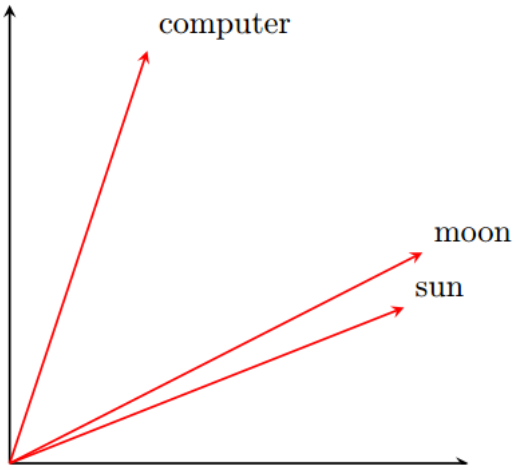

Figure 3.3: Example of distributional semantics vectors

using vectors. The last approach is based on the distributional approach: "words that occur in the same contexts tend to have similar meanings" [Pantel, 2005]. In the distributional semantic spaces (DSM), concepts that are semantically related to each other will be closer to each other in the semantic space. For example, we can visualize in Figure 3.3 that the vectors of "sun" and "moon" are closer than the vectors "sun" and "computer". This is the result of the fact that the words sun and moon tend to appear in the same context, which is the basic concept of DSMs.

It was shown that raw cooccurrence counts do not give a good representation and distributional semantic models achieve better results when different transformations are applied to the cooccurrence counts. These transformations were mostly done unsupervised, based on some independent considerations. In the last few years, it was shown that a supervised approach where the weights maximize the probability of the contexts gives very good results.

[Baroni et al., 2014] compares different distributional semantic models: methods based on counting like: pointwise mutual information, local mutual information and methods based on predicting. They show that the predicting models outperform the traditional models on most of the tasks and they recommend using the predict vectors for future experiments.

The state of the art predicting models are presented in [Mikolov et al., 2013]. These

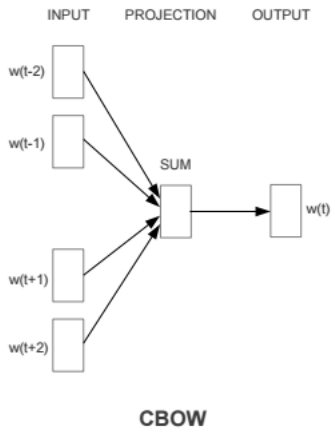

Figure 3.4: Models developed by [Mikolov et al., 2013]

approaches are based on previous neural network models [Bengio et al., 2003], but they are trained more efficiently and this reduces the necessary time and computing power. Because of these improvements, these models can be trained using more data and this leads to an improvement in the results. Figure 3.4 shows the basic steps implemented for the architectures from [Mikolov et al., 2013].

The continuous bag of words(CBOW) model projects all the words into the same position and this leads to the fact that word order does not influence the final prediction. Comparing with other approaches, CBOW also uses a symmetric window (the context) around the target word for prediction which means that it uses words from the "future". This approach is basicaly a log-linear classifier with a context including n previous words and n future words and the training is based on the task of classifying correctly the word in the middle. Also, because frequent words like 'the', 'and' do not carry a lot of contextual information, they will not have a positive impact on the model and the 'word2vec' toolkit discards words in the training data proportional with their frequency.

On the other hand, the second approach, the Skip-gram model predicts the context using the target word as input for the network.

As it was shown before by [Baroni et al., 2014], these methods manage to caption different and complex relations between words and they reach state-of-the-art results in different tasks:

- semantic relatedness: how close are words like 'queen' and 'king' in the semantic space.
- synonym detection.
- concept categorization: clustering the vectors in natural categories.
- analogy: answers to analogy questions: "brother" - "sister" + "grandson" = "granddaughter"

These models showed their power in a multitude of applications: topic modelling, named entity recognition, discourse analysis, question answering. They have very good results because they have an ability to encode the properties and characteristics of the target entities. These arguments lead us to use the continuous bag of word model with the parameters presented in [Baroni et al., 2014] to embed our linguistic input.

3.1.2 Visual embeddings

The previous section introduced different ways to represent linguistic information. Different studies show that the human brain does not use only textual input to represent the information, but it considers all the modalities when it builds lexical representations.

The representations extracted from images were mostly used in computational vision, but in the last few years they showed very good results in multimodal applications. The most successful approach for obtaining image emebeddings is by using Convolutional neural networks (CNN).

A convolutional neural network is a feed forward artificial neural network which was inspired by the structure of the visual cortex. The cells in the visual cortex are receptive to a small portion of the visual field which is called the receptive field. These portions are put together in order to cover the whole visual field. These cells behave as local filters over the input image. The CNNs used this structure for its layers and that conducts to a very good performance of CNNs in computation vision tasks.

The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's and the most well known is the LeNet architecture (Figure 3.5) that was used to read zip codes, digits [LeCun et al., 1995]. The first work that popularized

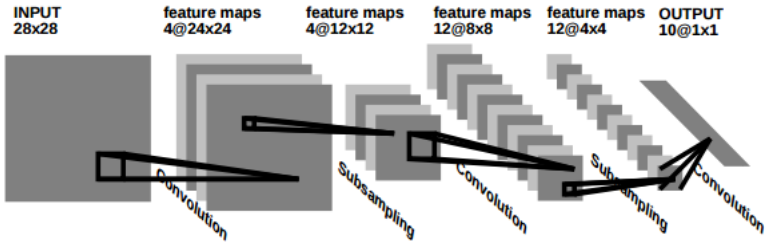

Figure 3.5: LeNet model used in [LeCun et al., 1995]

CNNs in Computer Vision was AlexNet, described in [Krizhevsky et al., 2012] which outperformed significantly the runner up of ImageNet challenge ILSVRC 2012: an annual challenge for object detection in images (with a top-5 error of 16% in comparison with 26%). It was based on the same principles used for LeNet, but it was bigger and deeper, adding more layers and filters to the initial design (see Figure 3.6). These changes were possible because the increase of computational power. The layers which constitute the basics of CNNs are:

- Convolutional layers: each neuron of these layers is connected only with a rectangular section of the previous layer and the weights of each section are the same.
- Pooling layers: compute the maximum or the average of the neurons in subregions. Most of the time extracting the maximum value shows the best results and it leads to dimensionality reduction.
- Fully connected layers: these layers connect every neuron with all the neurons from the previous layer.
- Loss layer: it applies a softmax function most of the time to produce a distribution over the categories used for classification.

The first part of the CNN simulates the visual cortex and it is constituted by multiple pairs of Convolutional and Pooling layers. At the end, we have a series of fully connected layers which ends with a loss layer. The technique used for training is backpropagation which is a common method for artificial neural networks.

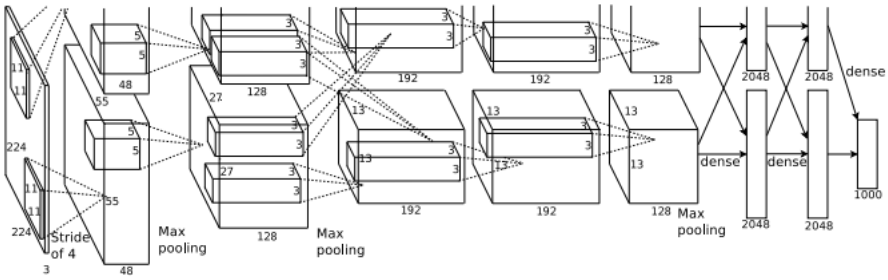

Figure 3.6: AlexNet model used in [Krizhevsky et al., 2012]

Another important detail of the convolutional neural network is the activation function used after each layer: ReLU (rectified linear unit) which is defined as: $f(x) = max(0, x)$. This function is very simple and it gives very good results. Because of its simplicity, it leads to reduced computation power and time compared with other acitvation functions.

AlexNet opened the ways to other approaches with amazing results in computational vision. The three most important systems developed using the concepts of AlexNet are: GoogLeNet [Szegedy et al., 2015], VGGNet [Simonyan and Zisserman, 2014], ResNet [He et al., 2015].

GoogleNet is the winner of ILSVRC 2014 and the innovation of this architecture is the inception module which reduces the number of parameters compared to AlexNet (from 60 million to 4 million). Also another important modification is the change from fully connected layers to pooling layers.

ResNet is the winner of ILSVRC 2015 and it is built using 152 layers, but the architecture is optimized with shortcut connections and extensive use of batch normalization. As well as GoogleNet, it does not have any fully connected layers at the end.

VggNet is the runner-up of ILSVRC 2014 and the success of this model is based on the depth of the network: 16 Convolutional/Fully Connected layers (19 in a later version) and it is more expensive to evaluate because of the big amount of parameters (140 millions). VggNet has slightly worse classification performance than GoogleNet and ResNet, but its features outperform the others in multiple transfer learning tasks because of the presence of the fully connected layers. Because of the last argument, VggNet is the most used model for extracting features from images and it was our choice for embedding the visual

information.

3.1.3 Memory Network

As I presented in the previous chapter, there are different architectures that are considered state-of-the-art methods for visual question answering. In previous works, it is shown that a model based on memory networks aids spatial attention and object identification. Another strong point of these models is that they are very flexible. They can be easily adapted to different tasks.

Memory networks are recurrent neural networks with an explicit attention mechanism that selects parts of the information stored in the memory [Xu and Saenko, 2015]. The original model was developed by [Weston et al., 2014] and it was described as a very good model for question answering. The model presented in this section is based on the memory network presented in [Sukhbaatar et al., 2015b] which showed its good results and flexibility in tasks such as: (synthetic) question answering and language modeling.

The approach presented by [Sukhbaatar et al., 2015b] is taking a discrete set of inputs: $x_1, x_2, \ldots, x_n$ which will be stored in the memory and a query q and outputs an answer a.

Figure 3.7 shows the steps followed by the memory network. In the first part, it is described how the simple model is behaving (without any recurrent steps):

- it calculates the similarity between the query and the embedding of each partition from the input. This step has as output a probability distribution where each value shows how related the query is to the corresponding sentence. The role of this step is to calculate how close to the query is each part of the input separetely and to determine where to focus for the next steps.

- the second step multiplies another embedding of the input (which can encode some spatial or temporal information) with the weights calculated at the previous step. These weighted embeddings are summed and this gives a single representation of the input relative to the query. This part of the model is used to detect different patterns in the structure of the input relative to the position of the information in memory cells and it leads to a very good information gist.

- the representation calculated before is summed with the initial query and a softmax function is applied to the summed vector. The result of softmax is used to predict the answer. The last vector is a representation of the input relative to the query and this vector will lead to the correct answer.

Intuitively, the memory networks can capture different characteristics of the input. This is facilitated by the fact that the input is split in the memory cells using diferent criterions and the model will focus its 'attention' more on some of the memory cells. For example, if you have a long text as input, the memory wil be populated considering a temporal variable, so the model will encode the information relative to the order of the sentences. In the case of an image as input, the image can be partitioned and each partition will be introduced in a different memory cell which will lead to a model which will consider the position in the initial image as an important variable. Following the previous examples, we can notice that the memory network is a model which analyses not only the input, but also its structure. An important detail for this neural network is the choice of the partition for the input data because this adds another layer of reasoning and it should be specific for the task.

As we can see in Figure 3.7, this model offers the possibility of a model with multiple hops. In this case, one hop is represented by the simple model presented above and its output is used as input for a similar model. This step is recurrent which builds a stack of simple memory networks that share the weights. This strategy leads to a deeper understanding of the relation between the query and the input which will increase performance for difficult cases. The stack of the simple models helps for discovering more complicated patterns, but it has also the downside that it can propragate the error generated in the first layers.

This model has behind the same strategy as recurrent neural networks, partitioning the input, but the RNN and LSTM models are unstable over long timescales. In comparison, the memory network uses a memory structure which shows more stability over the input information. This stability is because of the clear partition of the input considering time parameters which gives a consistency for the sequence analysis. [Sukhbaatar et al., 2015b] showed much better results when it was compared with systems based on LSTMs and

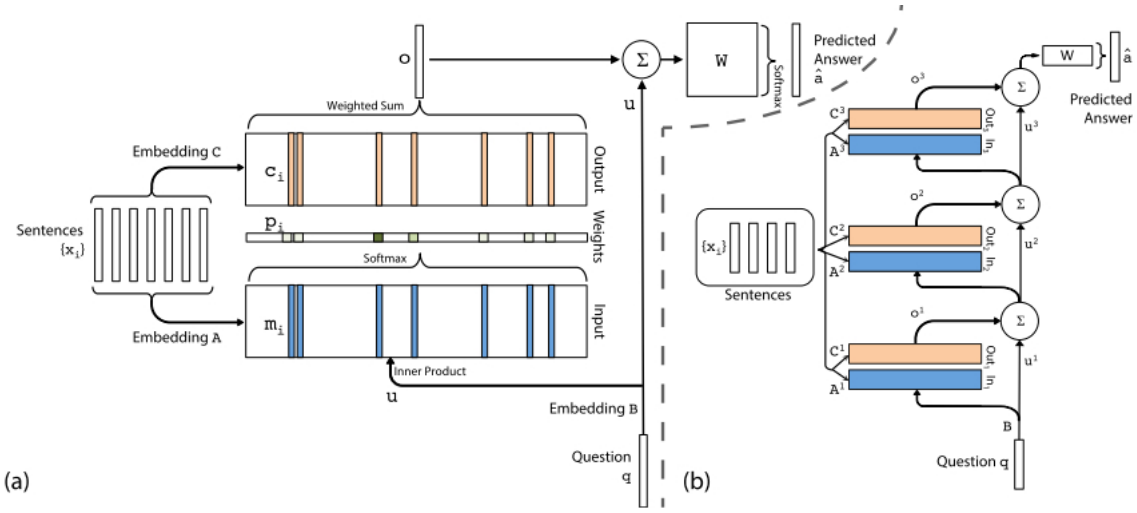

Figure 3.7: the model presented in [Sukhbaatar et al., 2015b]. a) presents a single layer version of the model. b) presents a three layer version of the model.

Structurally Constrained Recurrent Nets (SCRN).

Applications of Memory Networks

Even though the work of [Sukhbaatar et al., 2015b] is very recent, it was already applied in different topics of computational linguistics.

One of the applications of memory networks is learning from games [Sukhbaatar et al., 2015a]. This variation uses reinforcement learning to test the performance of memory networks on reasoning and planning in a newly developed environment.

The attention of the memory networks is one of the most important parts of the model and this is shown by different research directions that prove the importance of a good attention module inside neural networks. The attention module shows its contribution in entity extraction [Verga et al., 2016] and deep generative models [Li et al., 2016] and it is a primary key in understanding how and why the model works.

The most important adaptations of memory networks are applied to visual question answering. The model presented in [Xu and Saenko, 2015] gives a good overview of how the memory network is adapted for this task and why it is a suitable system. The attention module plays an important role in understanding what the system focuses on when it predicts an answer. This solves a problem that deep neural networks have: explaining

the reasoning behind a prediction.

One of the most important models adapted from memory networks is the dynamic memory network presented in [Xiong et al., 2016] which adds a mechanism similar to RNNs to help the reasoning and a better understanding of the input. This architecture shows very good results on different visual question answering scenarios, but it has an architecture which is not suitable for the task presented in this thesis because it treats the input sequentially and it loses the possible properties of the entire input.

Another recent application of memory networks is text understanding through a word prediction task in a broad context [Paperno et al., 2016]. In this case, the results of the memory network are not good and the reason for that is the limited capacity of the model to remember from the past. It is a model which can capture the information in a narrow context, but it can not find the connections in a broader context.

3.2 Our Model

The current task is to predict the correct quantifier when we have a visual scenario as input and a linguistic query.

Compared with the models presented in the previous section, the system designed for this task has to split its attention on different parts of the scenarios and it has to do this for multiple features. If the query is *How many dogs are black ?*, the model has to focus first on the parts of the scenario where there is a dog and on the second step to focus on the black objects. The other systems used for visual question answering use their attention module to focus on a primary region, relevant for the query. For this task, we are using the attention module to find how spread are the positive matches and not the exact position of the objects.

If we have a visual scene which contains different objects, we consider the images used as input in Figure 3.8 the corresponding bounding boxes of each object. Each object will be represented by a cell in the memory. A preliminary step is to find appropriate embeddings for the input and for the query. Because the input is a visual scenario, the approach is to use visual features extracted with state of the art convolutional neural networks. In this case, the visual features are extracted using VGG-19 [Simonyan and Zisserman, 2014]

which showed very good results on visual object identification. The representation of the linguistic query is generated using word2vec[Mikolov et al., 2013]. The arguments for these choices are presented at the beginning of the chapter.

When the input is queried with the question *How many dogs are black ?*, the first intuitive step is to focus and find the dogs. This is done in the first part of the model, when it is calculated the similarity between the linguistic representation of the restrictor(*dog*) and each visual representation of the objects from the input. The model decides if a bounding box contains a visual representation of the restrictor if the dot product between the corresponding embeddings is high enough. The dot product values for each bounding box are stored in the similarity vector 1 from Figure 3.8 after the whole vector is normalized. The most used activation function after the dot product is softmax, but this function spikes only for one value most of the time which means that the model would focus only on one object and it will not detect multiple positive matches. This step is similar to the initial design of the memory network, but it spreads the positive matches more in order to focus on more than one areas.

After the step presented above, the model will know approximately which cells in the memory contain the restrictor. This information is used in two different parts of the model. The first part is building Gist 1 which is a sum of the input vectors weighted by the values from similarity vector 1. This gives us the level of *dog-ness* in the visual scenario. The second part multiplies the visual input vectors by their corresponding similarity values. These new vectors (*weighted vectors* in Figure 3.8) will be used as visual input for the second step of the model. By multiplying the vectors, the model will give more power to the objects which were a positive match with the restrictor (if the restrictor is *dog*, the images with dogs will have bigger values for this step). The second step calculates the similarity between the scope (in the query used above, the scope would be *black*) and the new visual vectors. The method used for this step is similar to the method presented for extracting the images which contain the restrictor. After the dot product between the linguistic embedding of the scope (in this case, the embedding of *black*) and the weighted vectors, the model normalizes the vector and this result is represented by *similarity vector 2* in Figure 3.8. The values of this vector show how much *black dogness* is in each memory

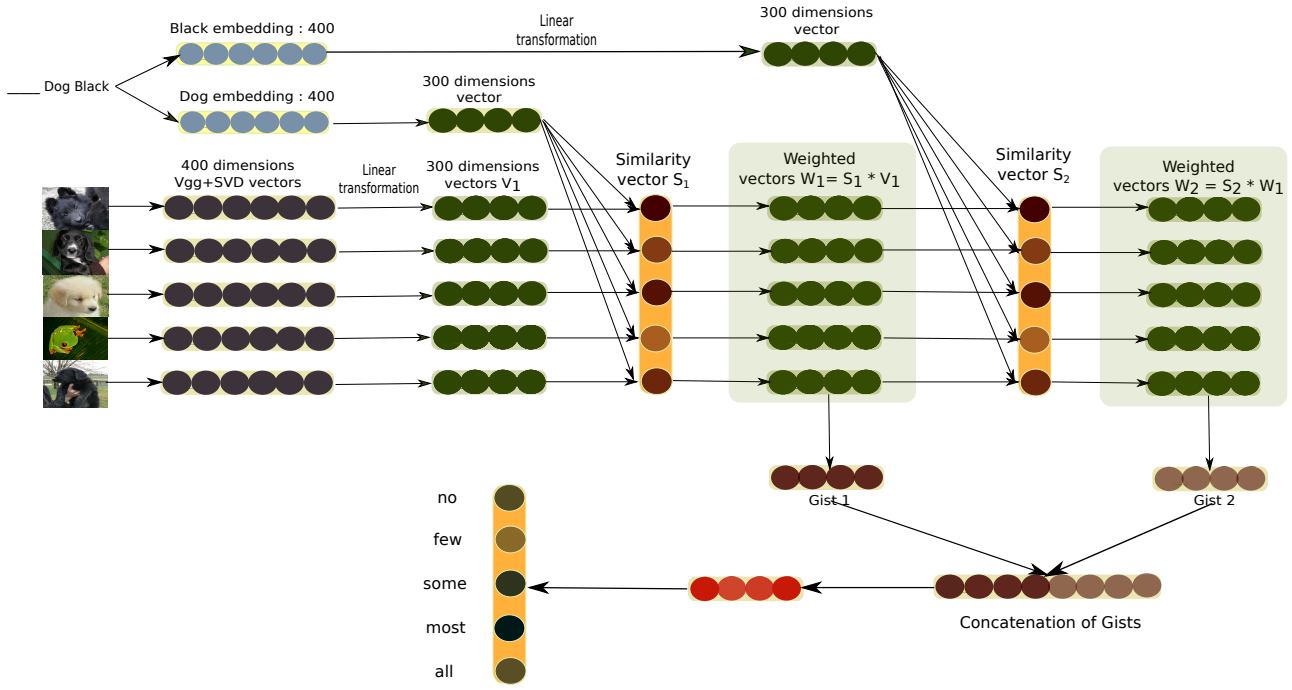

Figure 3.8: Memory network adaptation for the quantification task

cell. Similar to the first part, the model calculates a summary. The second summary (Gist 2) is a sum of the weighted vectors multiplied by the values in the similarity vector 2. This will represent the match between the visual input and the entire query.

This step is reflecting the power of the memory network to discover different variables: in the original model, we can see that the second step is encoding a temporal variable in its input, we are adapting the model and we encode the *restrictor* variable in the input of the second part of the model.

It is intuitive that in order to choose the correct quantifier, humans tend to calculate an approximate ratio between the objects that contain the restrictor (how many dogs are in the scenario) and the objects that are a positive match for the entire query (how many black dogs are in the scenario). The answer for the first question is stored in *Gist 1* and the

answer for the second question is in *Gist 2*. By concatenating these two gists and applying a linear transformation + activation function(softmax) on top of the concatenation, the model learns how to predict the correct quantifier by analysing the ratio between the two gists. If they are not similar at all, it means that the dogs in the scenario are not black and the correct answer would be *no*. If the gists are the same, this shows that all dogs are black, so the correct prediction is *all*.

Even though the last part resembles the general structure of the memory network, there is a slight change which is task specific: instead of summing the vectors, we are concatenating them because we want to compare the information stored in them, not only to merge them into a unified representation.

The model presented above is a general description of the system used for the visual quantification task. This architecture is considered suitable because it does not depend on an accurate counting method, which in most scenarios is a harder task. Also, it uses an approach which is intuitively close to how humans solve a similar task. It has been observed in children that they do not need to count in order to quantify. Another important positive factor of this model is that it can be adapted to calculate the attention in multiple hops (as shown in [Sukhbaatar et al., 2015b]) which might improve the object and property identification.

Chapter 4

Experiments

In the setup presented in this chapter, the questions ask how many objects of a certain type have a certain property, eg,. *How many dogs are black?*. The answers are selected from a range of linguistic quantifiers, namely *no, few, some, most*, and *all*. In order to assign the correct quantifier, the model must be able to focus on the relevant objects (the quantifier's 'restrictor') and to quantify which objects in this restricted domain have the queried property (the quantifier's 'scope'). We show that a bag-of-word model is not sufficient for such a task, and propose an alternative, linguistically motivated NN model, which outperforms the state-of-the-art VQA model of [Zhou et al., 2015a].

The task of visual question answering is a very hard task, which requires a high level of understanding and reasoning. On the other hand, this task is not as hard as a counting task where the model should predict the correct number of objects with some specific features. The quantification task is strongly related with the counting task, but it does not require a precise counting module.

4.1 Dataset

One of the biggest challenges for this task is to find a suitable dataset.

Because of the nature of the task, the visual input has to follow some requirements:

- the visual scene has to contain a relative big number of labeled objects in order to offer:
  - variability of the objects (dogs, cats, and trees appear in the same image when

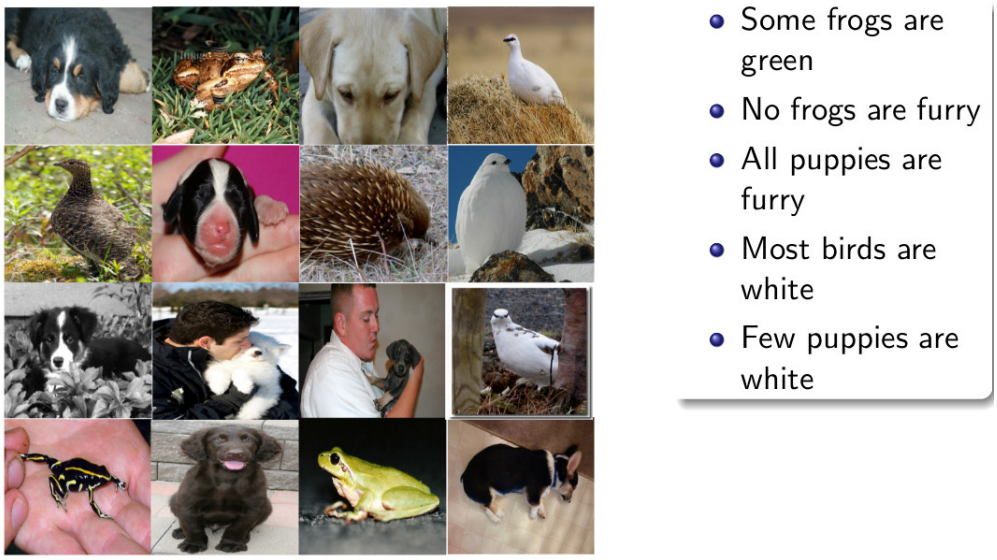

Figure 4.1: Dataset example

we analyse real scenarios)

- variability of the properties of the objects (black, white, brown, and spotted dogs appear in the same visual scene in real life)

- we need to have gold standard answers for each visual scenario. This requires a correct quantifier relative to the restrictor (an object label) and the scope (a property of the selected object).

There is no dataset which provides such a rich annotation so we chose to generate a dataset which simulates visual scenes and it is easy to annotate with the correct answer. Figure 4.1 presents a generated visual scene with some correct quantifiers relative to the example. The next subsections will explain the resources used for creating the dataset and the steps followed for this process.

4.1.1 Visual input

The central part of the dataset is the visual input. There are two different image databases which are used as resources: ImageNet [Deng et al., 2009] and MS-COCO [Lin et al., 2014b].

ImageNet is the biggest image database and it gives the opportunity for a multitude of breakthroughs in computational vision. It contains 14197122 images with labeled objects. It is organized according to the WordNet hierarchy [Miller, 1995]. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100000 synsets in WordNet, the majority of them being nouns. ImageNet contains images for 21841 synsets and it provides in average 1000 images for each one of them.

Another important feature of ImageNet is that around 9600 images contain labels for the properties of the objects. The properties are spread among different categories: color (white, red, green, black, etc.), patterns (spotted, striped), shape (rectangular, long, round, square), texture (wooden, shiny, metallic, etc.).

Considering the features presented above, ImageNet is a suitable database for our task but the problem is that each image contains only one object. This can be solved by putting together images from ImageNet to form an artificial visual scene. If the images for a scenario are randomly selected, we can have a fish and an airplane appearing in the same scenario, which is very improbable. MS-COCO will be used to calculate the probability of two objects to be in the same visual scene.

MS-COCO is an image recognition, segmentation and captioning dataset. It gathers complex everyday scenes containing common objects in their natural context. MS-COCO contains more than 300000 images and each image has 5 written caption descriptions.

The number of objects in the same image is not big enough to be suitable for our task, but we can use the captions to calculate what the probability of two objects is to appear together in natural context.

The PMI distance between two objects is calculated as follows:

$$PMI(o1, o2) = log \frac{f(o1, o2) * N}{f(o1) * f(o2)} \tag{4.1}$$

where $o1$ and $o2$ are two objects, $f(o1, o2)$ counts how many times the *words* $o1$ and $o2$ appear in the caption of the same image, $f(o)$ counts how many times o appears in the captions of MS-COCO dataset overall, and N is the number of words in all captions. Labels that are not in the captions are assigned uniformly distributed values in relation

to all other objects.

The most important detail related to the visual input is what we use as input for a corresponding image. We considered that the best approach is to extract visual representations using a technique capitalizing on Convolutional Neural Networks [Simonyan and Zisserman, 2014]. Specifically, we use the VGG-19 model pretrained on the ImageNet ILSVRC data [Russakovsky et al., 2015] and the MatConvNet [Vedaldi and Lenc, 2015] toolbox for features extraction. Arguments for this choice are presented in Section 3.1.2. Each image is represented by a 4096-dimension vector extracted from the 7th fully connected layer (fc7) and it is subsequently reduced to a 400-dimension vector by applying Singular Value Decomposition (SVD). These 400-dimension vectors are the visual input for our models. This approach was chosen because it is not computationally expensive and it still gives a good visual features representation.

4.1.2 Linguistic representation

Another important part of the dataset is to find a suitable embedding of the query. As it is presented in Figure 4.1, the restrictor and the scope are represented by words.

Section 3.1.1 shows that context-based predicting models outperform all the other approach when they are used for generating linguistic embeddings. We follow this reasoning when we choose the corresponding embedding for our queries.

As we presented before, the query is expressed as two words: $word_1$, the restrictor of the quantifier, and $word_2$, the scope. Each word is represented by a 400-dimension vector built with the word2vec CBOW architecture [Mikolov et al., 2013], using the parameters that were shown to perform best in [Baroni et al., 2014]. The corpus used for building the semantic space is a 2.8 billion tokens concatenation of the web-based UKWaC, a mid-2009 dump of the English Wikipedia, and the British National Corpus (BNC).

4.1.3 Scenario generation

We design a synthetic dataset using the resources presented in the previous subsections. The dataset contains 10000 scenarios made up by 16 different images extracted from ImageNet (see Figure 4.1). Each image used for the scenario has been annotated by

humans with one label corresponding to an individual object (*dog*, *wine*, *pizza*, etc.) and several properties associated with it (*black*, *spotted*, *furry*, etc.). The rationale for using 16 images per scenario is that this number allows us to have a fairly large amount of variability with respect to both the number of different objects in the same scenario (from 1 to 11) and the number of objects associated with each quantifier (e.g., *some* can be associated with quantities ranging from 2 to 11). This prevents the model from learning an association between a quantifier and a defined quantity instead of properly quantifying.

As a first step, we select all images from ImageNet that were annotated with at least one property associated with that object. In total, 9600 images representing 203 unique objects and 24 unique properties are selected. As a second step, we carry out a filtering of the objects based on both the number of images available for that object and the frequency of the linguistic label in the UkWaC corpus [Baroni et al., 2009]. The way we control for the variability of scenarios, queries and associated quantifiers depends on having many different images for one concept. The viability of the system also depends on having high quality linguistic vectors for each concept, which means discarding vectors that were trained on a small amount of data. We end up keeping items that have at least 150 occurrences in the corpus and at least 16 different images. This filtering process results in 161 different objects associated with 7324 images. In order to generate scenarios that are as realistic as possible, we use the PMI distance presented in 4.1.1 to decide if two objects can appear in the same scene.

For generating a scenario, we go through the following steps:

1. We randomly choose a label l from the set of 161 objects (e.g. *dog*) and we make it the restrictor of the quantification.

2. We choose a property p from the set of 24 properties (e.g. *black*) and we make it the scope of the quantification.

3. We select n_1 images that contain objects with label l and n_2 images that contain objects with the label l and property p, so that $6 \leq n_1 \leq 16$ and $0 \leq n_2 \leq n_1$.

4. We fill the empty 'cells' of the scenario $(16 - n_1)$ with images *not* labelled l by using the PMI distances as a proxy of their likelihood of co-occurrence with l.

5. Using the ratio between n_1 and n_2, we calculate which quantifier is applicable to the generated scenario, following some predetermined rules. That is, we ignore any semantic and pragmatic effects on the interpretation of quantifiers in context, and assume that they can be expressed as a fixed relation between the cardinality of their restrictor and scope. This is of course a simplification which will have to be addressed in future work. We set *no* and *all* to be correct when, respectively, no n_1 objects or all n_1 objects have the p property. For *most* and *few* we use prevalence estimates reported by [Khemlani et al., 2009] for *majority* and *low-prevalence* predications: we assign the former to ratios equal or greater than 70%, the latter to ratios up to 17%. All ratios falling between these two values are assigned to *some*. To illustrate, out of $n_1 = 6$ objects with label l, we assign *no* to cases with $n_2 = 0$, *few* when $n_2 = 1$ $(1/6 = 0.167)$, *some* when $2 \leq n_2 \leq 4$, *most* when $n_2 = 5$ $(5/6 = 0.833)$, and *all* when $n_2 = 6$.

6. We generate a query from l and p (e.g., *how many dogs are black*).

When generating the scenarios, we constrain the number of n_1 images labelled with the restrictor to be equal or greater than 6. Based on the ratios we used for defining the quantifiers, this number is the minimum for covering all quantification cases (see example above). This means that models can not gain accuracy by simply learning a correlation between low values of n_1 and the unsuitability of *few/most* in the associated scenarios.

4.1.4 Dataset variations and statistics

Because we want to analyze the rationale behind the scenario generation and we want to check the importance of the visual and the linguistic information in the system, we design multiple variations of the dataset: one in which we check the system's ability to deal with new linguistic input, using the same visual scenarios at training and test time ('unseen queries'); one where new visual scenarios are associated with previously encountered queries ('unseen scenarios'); and one 'uncontrolled', in which the model receives a

mixture of seen and unseen material (both visual and linguistic).

All the scenarios and queries in these dataset variations are built with the procedure described in the previous section. Each variation follows the same partition of the data: 7000 cases for training, 1000 for validation and 2000 for testing.

The characteristics of each setting of the dataset are:

Unseen queries: the queries in the test set are unseen, they are generated from those in the training set by keeping the quantifier and its restrictor (object) and changing its scope (property). That is, given a scenario and the query *some dogs are black*, we generate the new query *some dogs are furry* – where *furry* does not appear in training in combination with *dog*. This dataset consists of 10K scenarios, with an average of 2K scenarios per quantifier (2054 for *few*, 1943 for *most*, 2054 for *some*, 1890 for *all*, and 2059 for *no*) and 754.4 object-property combinations per quantifier (866 for *few*, 497 for *most*, 648 for *some*, 478 for *all*, 1283 for *no*).

Unseen scenarios: the scenarios in the test set are unseen and the queries are in the training set. The dataset consists of 10K scenarios with an average of 2K scenarios per quantifier (1998 for *few*, 1957 for *most*, 1972 for *some*, 1991 for *all*, 2082 for *no*) and 684 object-property combinations per quantifier (751 for *few*, 469 for *most*, 570 for *some*, 454 for *all*, 1176 for *no*).

Uncontrolled: this dataset does not follow any constraint. It consists of 10K scenarios; 2K scenarios for each quantifier and on average 827.6 queries (object-property combinations) per quantifier (905 for *few*, 508 for *most*, 665 for *some*, 517 for *all*, 1543 for *no*).

Except for the **unseen queries** dataset, which uses in the test set, scenarios from the training set, each query in the other datasets has a different test scenario.

The data used for dataset generation includes 161 unique concepts (e.g. dog, table) and 24 unique properties (e.g. furry, red). The frequency with which each concept occurs ranges from 1159 times (*bonsai*) to 8293 (*table*) (average 2981, standard deviation 1093). For each unique concept, an average of 48 images (standard deviation 99) have been used, with the distribution ranging from 13 (there are 13 unique images for the concept *pasta*) to 1104 (1104 unique images for *dog*). For building the scenarios, a total amount of 7735

unique images have been used. Each unique image can appear more than once in the dataset. In particular, the frequency of unique images ranges from 1 to 596 (average 55, standard deviation 62), meaning that there are images that occurs only once in the dataset, with other occurring up to 596 times in the full dataset.

Focusing on the annotation of unique images, the most used concept-property combination is *furry dog* (769 times), followed by *brown dog* (471) and *white dog* (464). It is worth noting that the concept *dog* is represented by 1104 unique images in the dataset, but only 769 out of them are annotated as *furry*. This means that more than 300 images of dogs are not annotated as *furry*, even though it is likely that the depicted dogs are furry as well. That is, in our setting we assume a property to be present in a given image only if the image is annotated with that property. The frequency of concept-property combinations ranges from 1 (e.g. *pink salmon*, *white ape*) to 769, with an average of 13.5 (SD 37). Focusing on properties only, *furry* is the most used one with 2936 unique images being labeled with that property, followed by *brown* (2782), and *smooth* (2266). The less frequent property is violet which occurs in 24 unique images. The average frequency is 801 (SD 837).

Each concept is associated with a different number of properties, with the number ranging from 2 properties to 18 (average 8, SD 3.4). The former case is represented by concepts as bonsai or lion, which are associated only with green, black and brown, furry, respectively. The latter is represented by concepts like dog or filter, which are associated with 18 unique properties. For example, dog is associated with furry, brown, white, black, smooth, gray, long, spotted and 9 others.

Focusing on the three different settings described above, there is a small difference regarding the number of unique images that have been used in testing data. For uncontrolled set we used 7124 unique images, 7171 for unseen scenarios set and 7125 for unseen queries (Table 4.1).

When we generated the datasets, we controled the frequency of the quantifiers and the structure of the scenarios. Other important parameters that can influence the final results are the frequency of objects or properties in the query. We decided to not control this because we wanted to create a more free environment.

	uncontrolled	**unseen scenario**	**unseen queries**	**total**
scenarios	10,000	10,000	10,000	30,000
images (tokens)	160,000	160,000	160,000	380,000
unique images (types)	7124	7171	7125	7735
unique concepts (types)	161	161	161	161

Table 4.1: Dataset structure

information	**unseen queries**	**unseen scenarios**	**uncontrolled**
mean frequency	43.4	43.5	43.8
standard deviation	10.5	10.5	10.6
highest frequency	rice - 68	boa - 69	wildcat - 67
lowest frequency	manatee - 18	cheetah - 16	zebra - 16

Table 4.2: Statistics of the objects frequency in the queries for the training set

The dataset contains images that can be splitted in 161 object categories. The objects can be very different (bat, fondue) or it can be very hard to distuinguish them (dog, puppy). As we can see in Table 4.2, the frequencies are consistent over all the dataset settings, so we can consider that this is not a reason for the higher performance. Also, we can observe that all the objects appear at least 16 times in queries in the training set which exposes each object for a fair amount of times to learn the mappings between objects and images.

The second variable in the training set queries is the frequency of the properties. The images are annotated with 24 properties in total, which behave as scope in the query. Table 4.3 presents the statistics of the properties in the training data. As we have seen in the previous table, the numbers are consistent over the different datasets which shows that the variation of the components is constant in all the experiments. The big difference

information	**unseen queries**	**unseen scenarios**	**uncontrolled**
mean frequency	291.7	291.7	291.7
standard deviation	173.3	168.05	166.2
highest frequency	brown - 718	brown - 704	brown - 704
lowest frequency	violet - 94	violet - 91	violet - 80

Table 4.3: Statistics of the properties frequency in the queries for the training set

property groups	unseen queries	unseen scenarios	uncontrolled
color	3672	3633	3602
pattern	303	321	315
shape	996	1026	1009
texture	2029	2020	2074

Table 4.4: The frequency of each property group in the training data

between the properties ('brown' has 718 appearances and 'violet' has 94 appearances) is caused by the number of images that are annotated with the specific property: 'brown' is associated with 2782 images and 'violet' is associated with 24 images. This big difference is caused by the annotation of ImageNet and it is reflected in the dataset structure.

Another important fact is that the properties belong to four bigger categories:

- **Color**: black, blue, brown, gray, green, orange, pink, red, violet, white, yellow
- **Pattern**: spotted, striped
- **Shape**: long, round, rectangular, square
- **Texture**: furry, smooth, rough, shiny, metallic, wooden, wet

Table 4.4 presents the distribution of the queries from the training data over the four property categories. There is a big difference between **color** and **pattern**, but this is justified by the number of the properties in each category: we have 11 properties in the **color** class and only 2 properties in the **pattern** class.

The statistics presented above show the consistency of the datasets and that the results reflect the nature of the task and not the structure of the input.

4.2 Experimental setup

We experiment with three different models:

1. a baseline, iBOWIMG, which is a state-of-the-art VQA system;
2. our proposed architecture, the quantification memory network (qMN), dedicated to quantification questions, described in Section 3.2.

3. an implementation of qMN with multiple hops, presented in Figure 4.3.

The models are all implemented in Torch 7 using the neural network package (*nn*) and the neural network graph package (*nngraph*). We chose Torch because it offers a very good support for neural network models and it is very fast, a very important factor when we train our models. The dataset is generated using Python 2.7 because of the simplicity of the programming language and because it offers data structures which are suitable for this task.

iBOWIMG

iBOWIMG is the visual QA model of [Zhou et al., 2015a]. In the original system,[1] the network is given as input an image and a user question, and must return a linguistic answer. The question is first converted to a one-hot bag-of-words vector, which is further transformed into a 'word feature' embedding. This linguistic embedding is concatenated with an 'image feature' obtained from a convolutional neural network (CNN). The combined features are sent to a softmax layer which predicts the answer by assigning appropriate weights to an output layer, where each node corresponds to a word in the overall vocabulary. The output step can be seen as a multi-class logistic regression model. The adaptations made for our task are presented in Figure 4.2.

We modify the original system in different parts, but the rationale behind it is the same.

First, our scenarios must be converted into a single image. The popular approaches in related work for this operation are: average or concatenation. In order to not lose some details regarding some parts of the input, we chose to define the input as a concatenaton of the 16 image vectors corresponding to our individual entities which gives a 6400-dimension vector. Also, we chose to use image embeddings and not one hot vectors because the input has a high complexity which is better captured by the embeddings.

Second, the output vocabulary is set to five nodes, corresponding to our five quantifiers of interest.

Quantification Memory Network (qMN)

This is the model we propose for the quantification task. The model and its structure

[1] Available from `https://github.com/metalbubble/VQAbaseline/`.

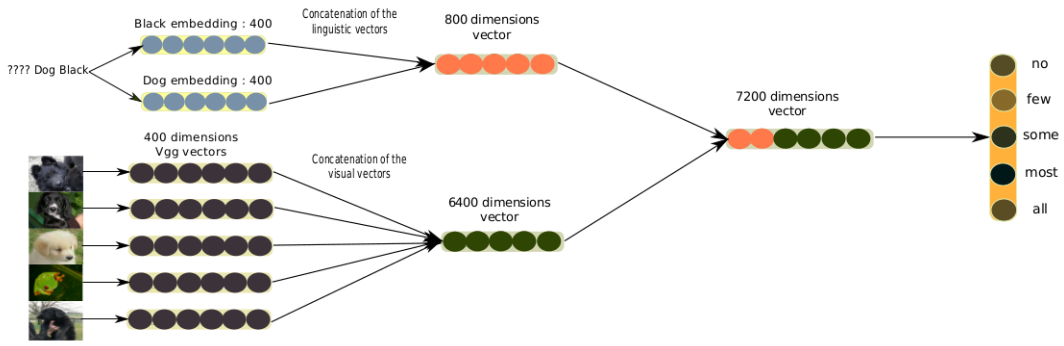

Figure 4.2: iBOWIMG model diagram

is presented in Section 3.2 Our model aims at quantifying over a set of images (i.e. a scenario like the one shown in Figure 4.1), given a linguistic query of the type *How many dogs are black?*.

A graphic representation of the model is shown in Figure 3.8 and the general ideas of the model are presented in Section 3.1.3. Specifically, in the model:

- both the visual and the linguistic vectors are linearly mapped to a 300-dimension space, and then fed into 16 memory cells;
- we calculate the dot product between the visual vectors and the restrictor vector (eg., *dog*) to obtain a 16-dimension vector that is further normalized using the Euclidean norm (Similarity Vector 1 in Figure 3.8);
- Gist 1 is calculated by summing the visual vectors weighted by the similarity values in Similarity Vector 1. Gist 1 represents how much 'dogness' is found in the given scenario ;
- we then calculate the weighted vectors for each individual by multiplying the memory cells with the associated similarity values in Similarity Vector 1. This gives us a representation of the amount of 'dogness' in each object;
- we calculate the dot product between the obtained weighted vectors and the scope vector (eg., *black*). The resulting 16-dimension vector is further normalized using the Euclidean norm; This vector is represented by *Similarity vector 2* in Figure 3.8.

- Gist 2 is obtained by summing the memory cells weighted by the similarity values in Similarity Vector 2. Gist 2 represents how much 'black-dogness' is found in the given scenario.

- Gist 1 and Gist 2 are concatenated in a single 600-dimension vector that is further linearly transformed into a 5-dimension vector;

- we apply a softmax classifier on top of the resulting vector that returns the probability distribution over the quantifiers.

Quantification Memory Network with multiple hops

This approach is summarized by Figure 4.3 and it is based on the memory network with multiple hops presented in Section 3.1.3.

The model calculates multiple times the gists. At each step, it uses the previous gists as input and this approach should give a better understanding of the relation between queries and images.

As we can see in Figure 4.3, this variation has some parts which are different compared to the model presented in [Sukhbaatar et al., 2015b] because it calculates recurrently and separately Gist 1 and Gist 2. Gist 1 should describe the relation between images and restrictor and Gist 2 should describe the relation between images and query. Because their role is different, we do not apply the recurrent steps on their concatenation and we keep it separate.

This variation of the model should learn deeper relations, but it is also more sensitive on the amount of data used for training. The model does not see an image so many times (around 20 times during training) and this can cause more confusion if the complexity of the model is increased.

The models are tested in all three different settings: 'unseen queries', unseen scenarios' and 'uncontrolled', in order to analyse the task from different angles. In all cases, each scenario-query combination has only one correct answer.

For each dataset, 70%, 10% and 20% of the data are used for training, validation, and test, respectively.

models	unseen queries	unseen scenarios	uncontrolled
iBOWIMG	46.5	36.4	30.8
qMN	54.1	**44.0**	**38.9**
qMN-2hops	**58.1**	**44.0**	35.0
qMN-3hops	52.7	43.6	36.5

Table 4.5: Results: all numbers are accuracy scores.

4.3 Results

The models analysed in this section are the models presented in the previous section for all three settings of the dataset. This variation in model and dataset gives a better understanding of the task and the suitable approach for it.

As it can be seen in Table 5.6, our **qMN** models significantly outperform the **iBOWIMG** baseline in all three datasets (it increases accuracy by around 8% in each setting). When we compare the results of the simple qMN model with the results of qMN with multiple hops, we observe that the precision drops few percentages for more complicated datasets when the complexity of the model is increased. In the case of the unseen queries, the precision for the model with 2 hops is higher with 4% compared to the simple qMN model. Even though we have this fluctuation of the precision when we use different variations of the model, all the variations have better results than the baseline.

As far as the **unseen queries** are concerned, **qMN** achieves an accuracy of 54.1% against the 46.5% of the baseline. **qMN with 2 hops** reaches a precision of 58.1% which is the best result obtained for this task. Overall, the correct quantifier is always numerically prevalent over the others. In particular, the **qMN** models work very well in predicting *no* and *all*, with the task becoming slightly harder for *few* (partly confounded with *no*), *most* (partly confounded with *all*), and *some* (equally confounded with all the other competitors). A more detailed distribution of the predictions can be seen in Table 4.6.

This trend can be further understood by looking at the percentages of quantifiers which are correctly predicted by each model, reported in Table 4.7. The baseline model, in contrast, is effective in predicting *all* and *no* (for the latter, it achieves an accuracy of

Table 4.6: Confusion matrices for the Unseen queries test

Unseen queries											
qMN						**iBOWIMG**					
	some	all	no	few	most		some	all	no	few	most
some	187	71	53	51	72	some	112	139	60	64	59
all	33	226	27	13	38	all	34	207	61	4	31
no	27	8	334	52	10	no	4	8	416	1	2
few	61	43	129	165	36	few	87	77	128	104	38
most	52	107	23	11	171	most	56	148	49	19	92
qMN-2hops						**qMN-3hops**					
	some	all	no	few	most		some	all	no	few	most
some	203	36	56	53	86	some	142	82	53	62	95
all	28	224	26	18	41	all	28	221	33	11	44
no	11	22	373	19	6	no	6	1	424	0	0
few	58	43	119	156	58	few	71	58	128	131	46
most	37	81	23	17	206	most	49	122	31	26	136

96%), but the performance for the other quantifiers is very bad, since the model predicts more wrong answers than correct ones for *some*, *few*, and *most*. This means that the overall accuracy is biased by the extremely good performance on *all* and even more on *no*, but the model does not work properly for the other quantifiers, where the percentages of correctly guessed answers are significantly lower compared to the **qMN** model (see Table 4.7). Also, we can see the same behavior for the **qMN with 3 hops** model, but the results are more uniform.

In the **unseen scenarios** dataset, the **qMN** models and the baseline model achieve an overall accuracy of 44.0% and 36.4%, respectively. The simple qMN is very effective in predicting *no* and *all*, whereas it has some confusion with both *some* (partly confounded with *most*) and *most* (partly confounded with *all*). For the cases where *few* is the target quantifier, instead, the model predicts more *no* than *few* answers, and the percentage of correctly predicted *few* is indeed lower compared to the other cases, as can be see in Table 4.7. The other two variations of qMN show a more uniform distribution of the results over the 5 quantifiers, with lower precision for *all* and *no*, but with higher precision for the others. The baseline model, in contrast, always assigns more correct answers to the target quantifier than to the competitors. However, a general confusion is in place

for all the quantifiers, with *no* and *all* achieving significantly lower accuracy compared to the qMN models.

In the **uncontrolled** dataset, the simple **qMN** achieves an overall accuracy of 38.9% against the 30.8% accuracy of the **iBOWIMG** baseline and it is also better with 2-3% compared with the models with multiple hops. The **qMN** model is again very accurate for *no* and *all* and it works fairly well for both *few* and *most*. But the accuracy drops for *some* (18.16%), where the model assigns more answers to other quantifiers than to the target, and whose low accuracy probably affects the overall performance of the model. A different pattern of results is achieved by the baseline, which is fairly accurate in predicting both *no* and *all*, but works quite badly in all other cases. That is, *few*, *some* and *most* are equally hard to be predicted. Even though the results for the models with multiple hops are worse than the simple model, they are more uniformly distributed over the quantifiers. For these models, the precision for *some* increases which leads to a smaller gap between the precision of different quantifiers.

Notably, the **iBOWIMG** baseline performs particularly badly on the quantifiers *few* and *most* in all setups. We believe that this shows the limits of a network without reasoning abilities. Those cases require fine-grained decisions to be made about the extent to which the restrictor and scope of the quantifier overlap. In contrast, *all* and *no* (the 'easiest' cases for the baseline across setups) do not require any hard reasoning: the composition of the linguistic query words (e.g. *black* + *dog*) can be straightforwardly matched to the presence or absence of the corresponding visual feature ('black-dogness'). The **qMN** models, in contrast, turns out to be generally effective in predicting all quantifiers across the different datasets. That is, the overall accuracy is equally distributed over all quantifiers. By focusing on the errors the model makes across the datasets, moreover, it can be seen that it 'reasonably' confounds, for example, *few* with *no* but not with *some*, *most* and *all*, thus indicating that it is learning to quantify rather than simply keeping track of the correlations in the dataset. The same is not true for the baseline model, where the errors are much less interpretable (for example, *few* is confounded with *no* and *all* to almost the same extent).

The higher accuracy obtained on the **unseen queries** with respect to the other two

Table 4.7: Percentages of target quantifiers correctly predicted by each model

	Unseen queries			
	qMN	qMN-2hops	qMN-3hops	iBOWIMG
some	43.08	**46.77**	32.72	25.8
all	**67.06**	66.5	65.57	61.42
no	77.5	86.54	**98.3**	96.52
few	**38.01**	35.95	30.2	23.96
most	**46.97**	56.6	37.36	25.27
	Unseen scenarios			
	qMN	qMN-2hops	qMN-3hops	iBOWIMG
some	32.62	**41.97**	33.69	39.83
all	**50.51**	41.28	48.97	34.1
no	**67.99**	60.7	58.5	50.33
few	25.86	32.75	**33.74**	26.84
most	39.25	**40.58**	**40.58**	29.17
	Uncontrolled			
	qMN	qMN-2hops	qMN-3hops	iBOWIMG
some	18.16	25.37	**27.36**	22.13
all	**52.22**	39.6	38.11	40.34
no	**59.7**	53.23	58.46	49.5
few	**32.25**	24.3	27	21.25
most	**32.14**	**32.14**	31.37	20.4

tests confirms the need of reasoning required by the proposed task. As the baseline shows, NNs do quite well at capturing correlations, in other words they perform well at matching a word and a picture, which is what the models need for the **unseen queries**. Changing the scenario (**unseen scenario** and **uncontrolled** tests), instead, means changing the reasoning needed to answer the question.

When we analyze the results for the variations of the **qMN** model, we see that they do not follow the same pattern. In Table 4.7, we can see that the best precisions are reached by the qMN models. Another important fact is that for each dataset, each model has at least one quantifier where it reaches the best precision. This shows that different variations of the model manage to learn other relations. Even though the structure is the same, the differences between the distribution of the results show that another type of reasoning is reached when the model increases the complexity. Still, the best results are achieved by the simple model which can be explained by the difficulty to train a

more complex model with a relatively small dataset (7000 complex scenarios, each image appears around 20 times). Intuitively, the results should increase more for the complex models when the dataset size is increased. This is also suggested by the fact that the the best precision is achieved by **qMN** with 2 hops when is is exposed to a simpler dataset.

4.4 Conclusions

Following the results presented in the previous section, we show that visual question answering for quantification questions is a hard task which can not be solved just by memorizing correlations from the dataset.

The model proposed in this chapter achieves good results on all the setups used for experiments relative to a state-of-the-art model, but there is a lot of improvement that can be made. Also, the more complex **qMN** models do not show a significant improvement compared to the simple model. Intuitively, they will perform better with a bigger dataset because they need to be exposed to more information in order to develop deep reasoning.

The multiple hops **qMN** was inspired by [Sukhbaatar et al., 2015b] and adapted for the current task, but there are some variations of the hops which can lead to different results. These variations do not change the idea behind the model, but they combine the information in different ways and this can give us other insights into the task.

As we have seen in the previous section, the models have better results for *no* and *all*. This can be caused by how we define the quantifiers, and how we place the boundaries between them. We consider that the quantifiers are exclusive (we can not have two quantifiers for the same scene), but this is not how quantifiers are used by humans. The boundaries between them is more blurry and this should be another change for future works.

The generated dataset is as close as possible to real scenarios, but we can not say that it reflects real visual scenes. This change can go smoothly because each cell in the memory can be associated with a bounding box and there are algorithms which extract the bounding boxes from an image with high precision. The challenging part of this change is dataset generation, which requires a lot of resources: we need a lot of visual scenes, but they must be manually annotated with the correct quantifier.

The experiments and the results presented in this chapter open the studies for VQA on quantifier questions, but as presented above, there are still a lot of open questions for this task. They are very challenging, but they can lead to a better understanding of the relations between linguistic and visual information.

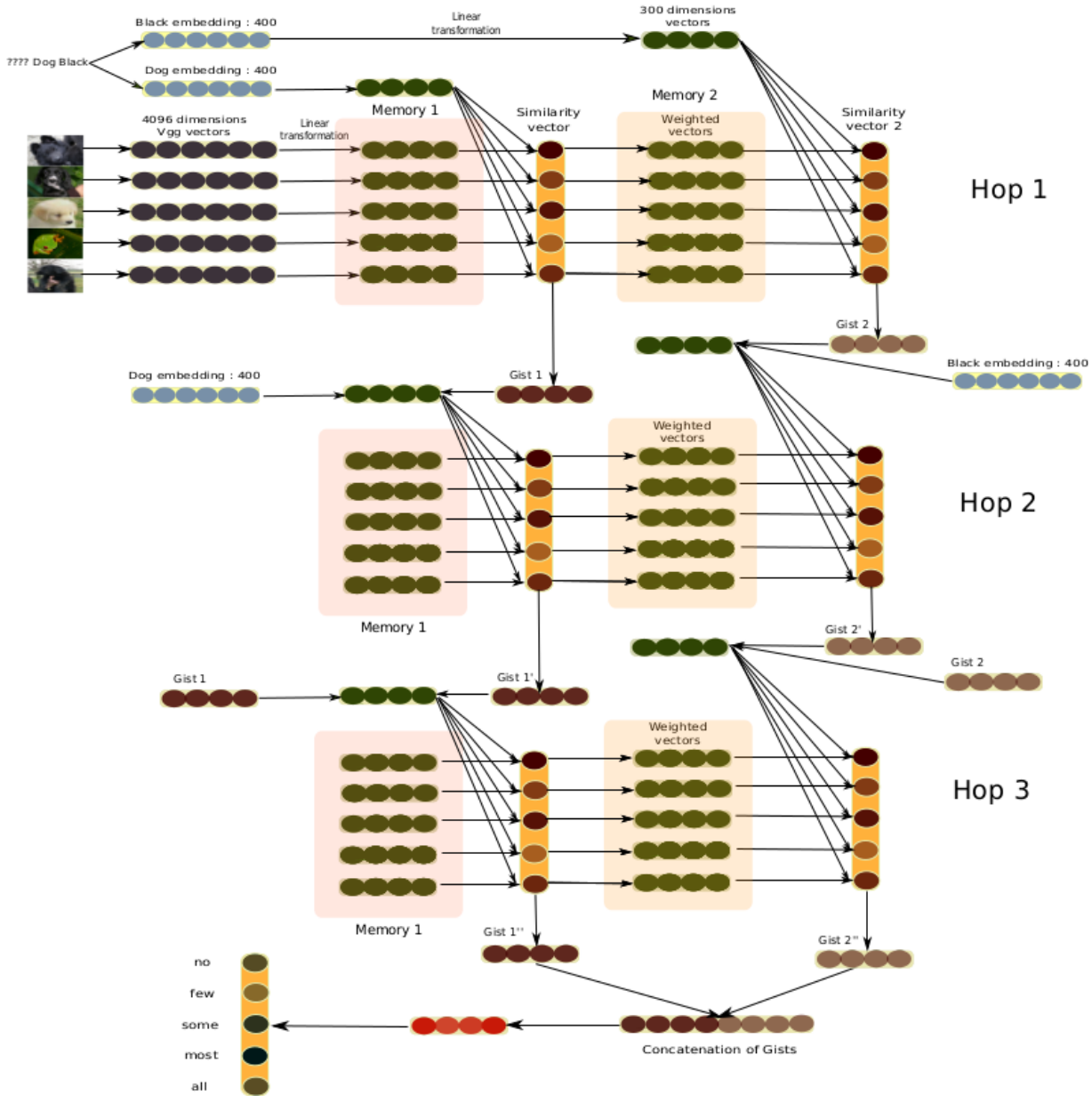

Figure 4.3: qMN model with multiple hops

Chapter 5

Analysis

As we have seen in the previous chapter, the proposed model shows better results than a simple neural network model.

This chapter gives an in-depth analysis of the predictions of the system and it gives a different, more theoretically focused perspective on the task in the second part.

5.1 Qualitative analysis

This section presents more details about the results, and it checks if the models learn specific patterns and if they are more biased to specific cases.

5.1.1 Predictions analysis

More information and insight about the results can be discovered when the predictions are analyzed from multiple points of view.

The results in the previous chapter were presented using the precision overall and over the quantifiers, because this is the main goal of the task. But we can see other characteristics when we analyze the precision for each object, property or property group.

First of all, the diagrams 5.1, 5.2, 5.3 present the frequency of the objects in the test data for all three datasets. We can observe that even though we have restrictions for two of the datasets, the distribution remains the same. Also, this shows that all 161 objects were used in the test data. Ideally we would want all the objects to appear 12 or 13 times in the test data to give a uniform image, but this can not be done due to the different amount of images for each object label.

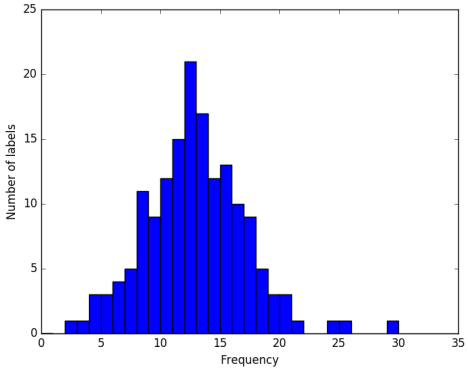

Figure 5.1: The distribution of the objects in the test set for the uncontrolled dataset

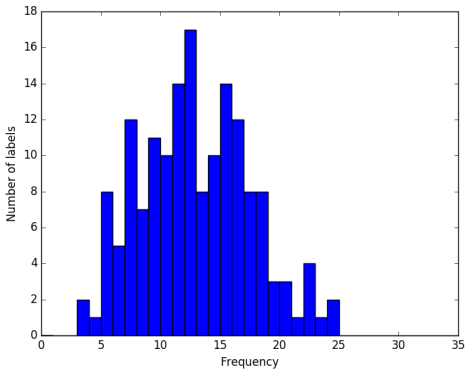

Figure 5.2: The distribution of the objects in the test set for the 'unseen scenarios' dataset

An interesting piece of information is if there are objects that tend to give good results independently from the rest of the information. A scenario like this can be caused by the power of the embedding or by the quality of the images for a specific object label. Table 5.1 presents the best 5 objects for each dataset. The objects show a very good variation, only *orangutan* appearing in two of the lists. Also, we can see that we have animate and as well as inanimate objects. The list contains animals, objects and food.

On the other hand, Table 5.2 presents the worst 5 objects for each dataset. Comparing with the previous table, the results seem to belong to the same classes, with only one exception. We can see that we have in the rankings, objects that are liquid, like lemonade, wine and syrup, which is an interesting result, but it is expected to have more noise when you deal with images that are annotated with these labels.

unseen queries	**unseen scenarios**	**uncontrolled**
panther - (15, 15, 100%)	cheetah - (5, 5, 100%)	bonsai - (8, 8, 100%)
lion - (7, 7, 100%)	manatee - (10, 9, 90%)	bottle - (8, 7, 87.5%)
panda - (6, 6, 100%)	sofa - (6, 5, 83%)	orangutan - (5, 4, 80%)
key - (11, 11, 100%)	buckle - (12, 10, 83%)	ferret - (8, 6, 75%)
orangutan - (5, 5, 100%)	wombat - (11, 9, 81%)	fondue - (7, 5, 71%)

Table 5.1: Objects with good results for each dataset - the tuples are (total number, correct predictions, precision)

unseen queries	**unseen scenarios**	**uncontrolled**
cow - (11, 0, 0%)	lemonade - (9, 0, 0%)	wine - (8, 0, 0%)
gnu - (9, 1, 11%)	deer - (14, 1, 7%)	syrup - (16, 1, 6%)
rodent - (14, 2, 14%)	squirrel - (12, 1, 8%)	salad - (12, 1, 8%)
marimba - (6, 1, 16%)	goat - (7, 1, 14%)	bench - (12, 1, 8%)
sunflower - (5, 1, 20%)	bread - (12, 2, 16%)	basketball - (9, 1, 11%)

Table 5.2: Objects with bad results for each dataset - the tuples are (total number, correct predictions, precision)

unseen queries	**unseen scenarios**	**uncontrolled**
green - (20, 14, 70%)	metallic - (40, 32, 80%)	violet - (26, 18, 69%)
spotted - (427, 294, 68%)	violet - (34, 27, 79%)	square - (30, 19, 63%)
orange - (35, 24, 68%)	pink - (37, 24, 65%)	spotted - (42, 25, 59%)
yellow - (127, 84, 66%)	square - (25, 16, 64%)	pink - (28, 16, 57%)
wooden - (31, 20, 64%)	green - (86, 51, 59%)	striped - (35, 19, 54%)

Table 5.3: Properties with good results for each dataset - the tuples are (total number, correct predictions, precision)

unseen queries	**unseen scenarios**	**uncontrolled**
pink - (36, 7, 19%)	rough - (94, 30, 32%)	shiny - (99, 24, 24%)
long - (132, 42, 31%)	shiny - (89, 29, 33%)	brown - (186, 49, 26%)
blue - (38, 9, 32%)	brown - (183, 60, 33%)	gray - (112, 33, 29%)
shiny - (33, 13, 39%)	white - (173, 58, 34%)	yellow - (68, 21, 30%)
red - (20, 8, 40%)	black - (148, 54, 36%)	white - (184, 60, 33%)

Table 5.4: Properties with bad results for each dataset - the tuples are (total number, correct predictions, precision)

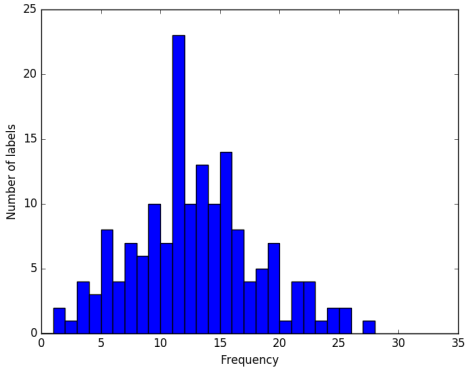

Figure 5.3: The distribution of the objects in the test set for the 'unseen queries' dataset

Similar to the best and worst object categories presented above, I also extracted the best and the worst properties from the test data. The top five list of each dataset is presented in Tables 5.3 and 5.4.

These tables show some consistency in the rankings, which is important when you analyze the results. First of all, it seems that properties that have more associated images give lower precisions. We can see that *brown*, *shiny* and *white* appear constantly among the worst properties, and these are the labels with the biggest number of images. On the other hand, properties like *violet* or *square* have a low quantity of images, and they give the best precisions. These statistics can be explained by the fact that if you increase the number of images associated with a property, that property will become more noisy and it will be more difficult to map the visual and the textual information. Interestingly, these tendencies are nullified in the **unseen queries** dataset which can be a reason for the higher precision.

Another interesting fact that can be extracted from Tables 5.3 and 5.4 is that the precision of the extremes does not increase when the general perfomance is increasing. For example, the **unseen queries** dataset has a precision 20% greater than the **uncontrolled** dataset, but the best properties have the same precision. This means that the bigger performance of the **unseen queries** closes the gap between the best and the worst properties, an encouraging phenomen.

Another statistic covers the precision for each property group. As we have seen before,

property groups	unseen queries	unseen scenarios	uncontrolled
color	(659, 337, 51%)	(1002, 428, 42%)	(989, 350, 35%)
pattern	(454, 307, 67%)	(99, 52, 52%)	(77, 44, 57%)
shape	(285, 131, 46%)	(303, 142, 49%)	(304, 123, 40%)
texture	(602, 289, 48%)	(596, 268, 45%)	(630, 244, 38%)

Table 5.5: The precision of each property group in the test data the tuples are (total number, correct predictions, precision)

the properties are split into four different categories. We can see in Table 5.5 that the results are close to each other with a higher precision for the **pattern** category. This category has only two instances, and neither has many associated images. So these results reflect the claim from the previous paragraphs that properties with a lower number of images perform better.

In the diagrams presented in Figures 5.4, 5.5, 5.6, we show the precision of the simple model(qMN) relative to the frequency of the images that are correct compared with the query. This information is key to understanding the rationale behind our model and to verifying if it is treating the task as a problem of matching the quantifier with specific numbers or rather if it is developing a deeper understanding of quantification.

As we can see in Figure 5.4, the precision for the uncontrolled dataset does not follow specific patterns. We can see an increasing precision for *most* and for *all* when the number of the correct matches is larger, which we can say happens intuitively to humans. It is easier to say that **all dogs are black** when all images are black dogs, compared with the scenarios in which only half of the images are black dogs and there is more noise around the target. This is a behaviour that it is reflected by the diagram but that is not very powerful, which shows that the model's learning is not limited to some basic associations. Also, the precision does not increase constantly relative to the frequency.

The diagrams for the *unseen queries* dataset (Diagram 5.5) and for the *unseen scenarios* dataset (Diagram 5.6) reflect the increased precision overall, but they show improvements for all the frequencies, which indicates that the model is not biased for specific cases and that it has a generally better behaviour. These facts show again that our results reflect the initial premise and that the proposed model is suitable for the task.

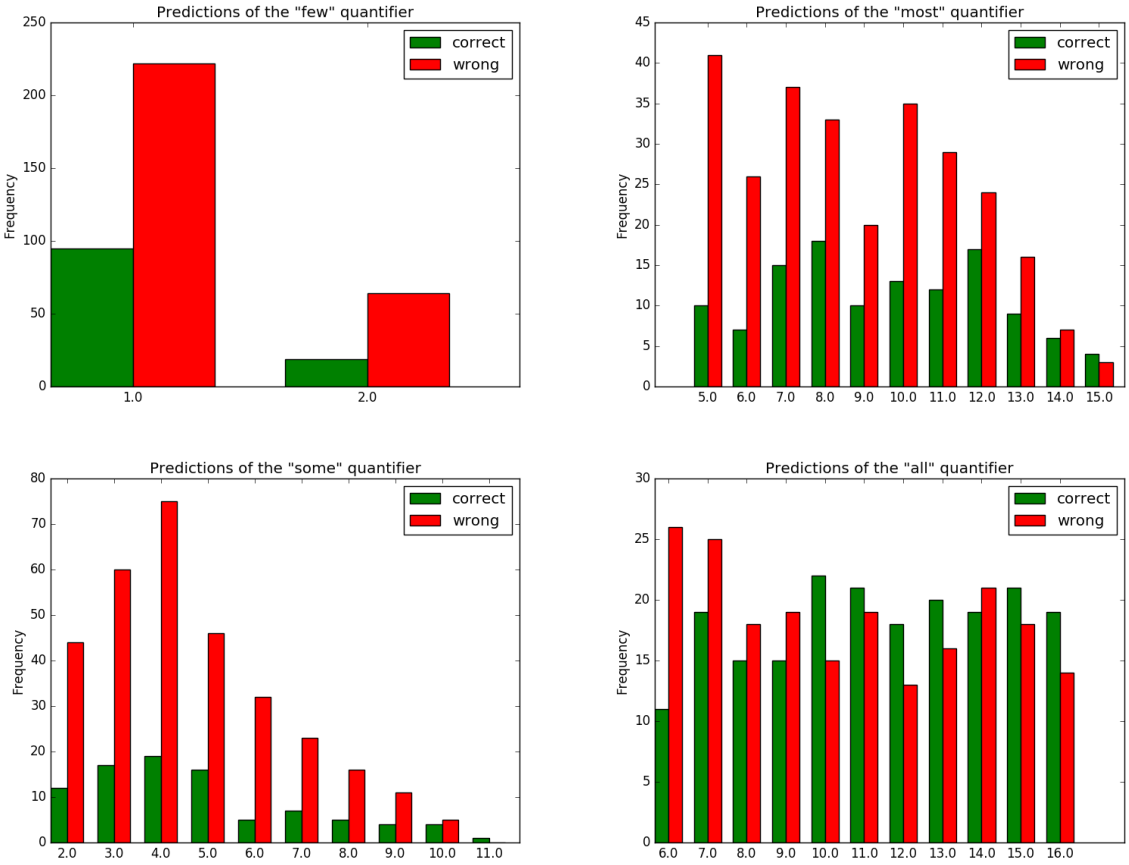

Figure 5.4: Precision on the uncontrolled dataset relative to the frequency of the correct match

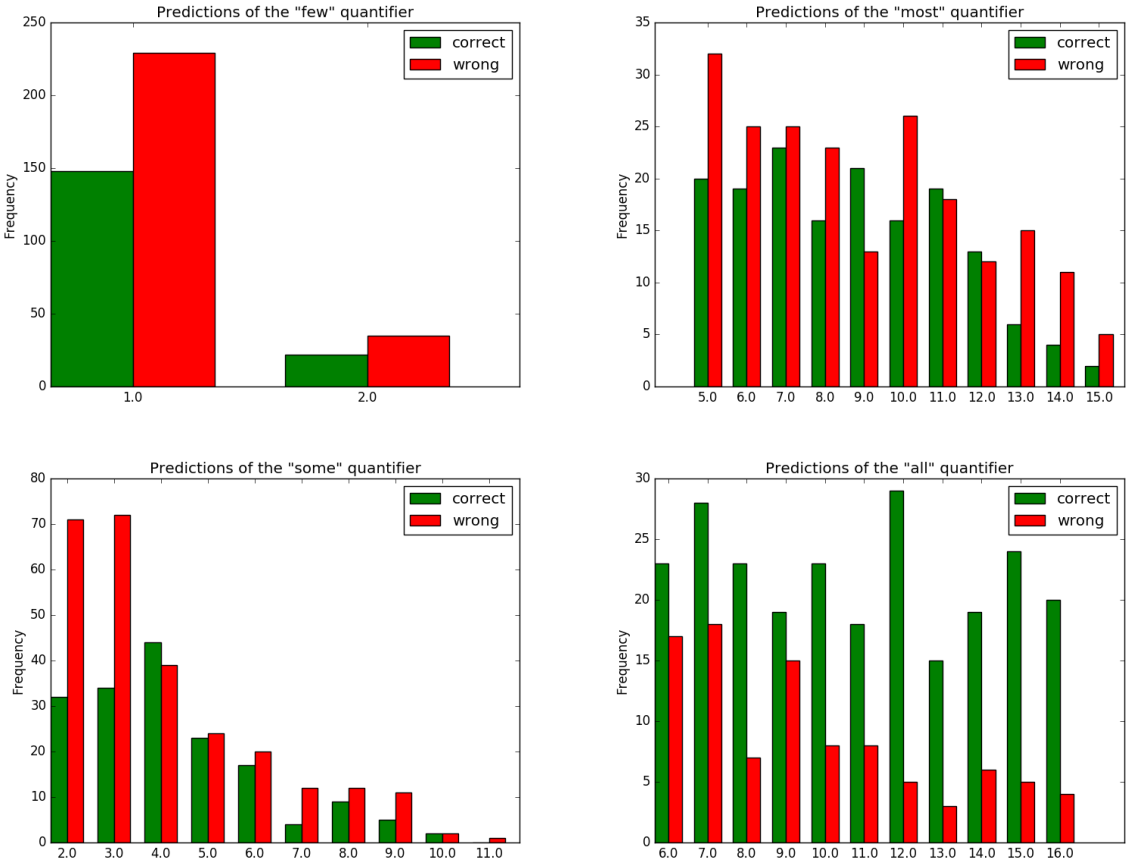

Figure 5.5: Precision on the *unseen queries* dataset relative to the frequency of the correct match

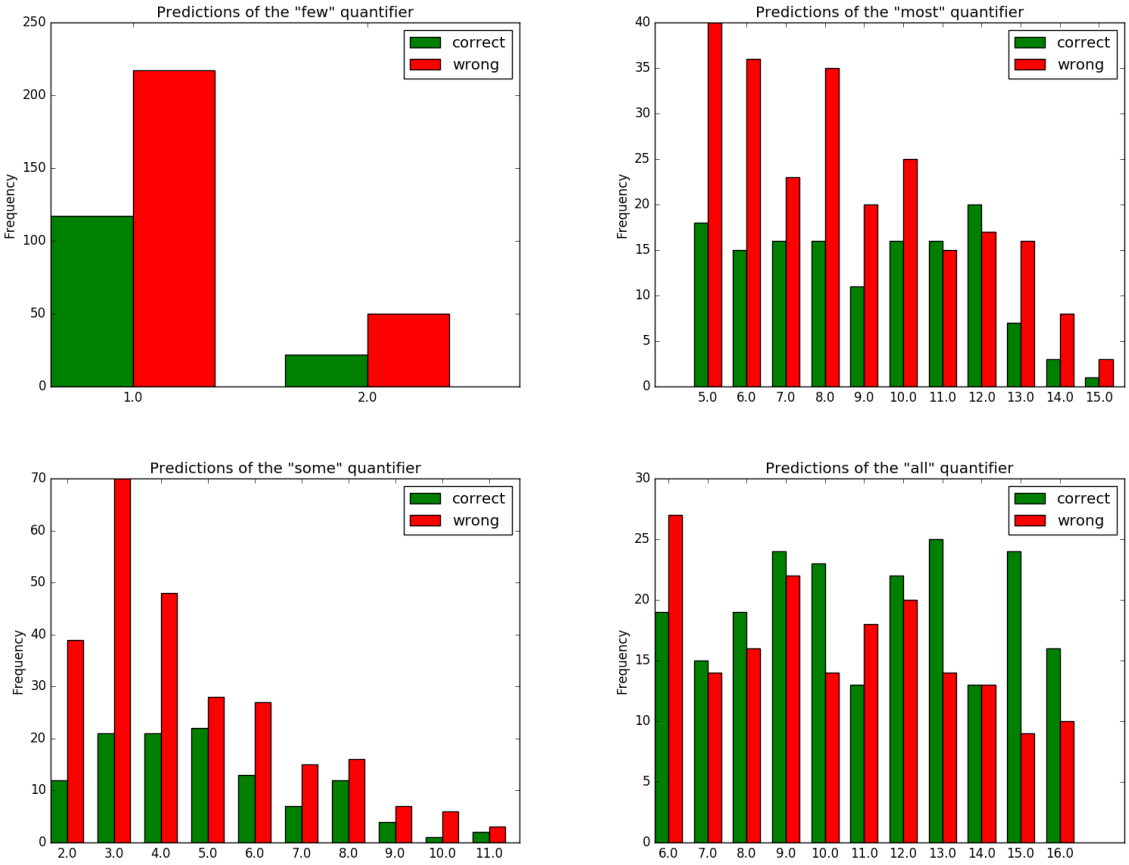

Figure 5.6: Precision on the *unseen scenarios* dataset relative to the frequency of the correct match

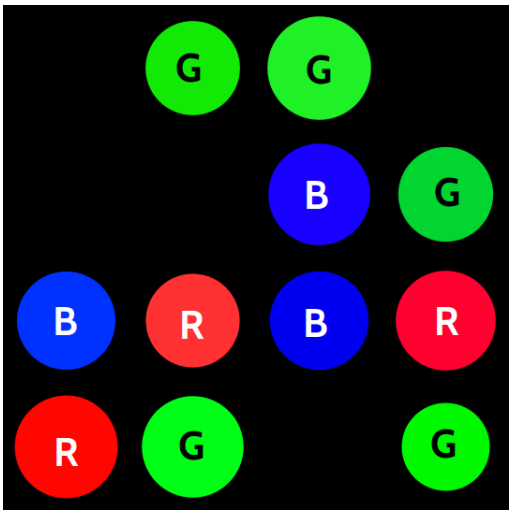

Figure 5.7: Input example

5.2 Experiments on logical quantifiers

Because the previous task is dependent on learning the relations between visual input and a linguistic query, we decided to conduct an experiment which analyses the problem of quantifiers from a more theoretical point of view. For this study, we also fix the restrictor and we focus on the scope.

In this case, the input is a set of circles with different colors (see Figure 5.7). The query is a visual representation of a circle with a specific color. The answers can be only no/some/all in this setup. We design this setup to test different models which are easier to adapt for this task, compared with the previous one. The models developed for this chapter were previously presented in [Sorodoc et al., 2016].

This chapter presents a simpler setup which shows different insights on quantifiers. The main hypothesis analyzed in this chapter is that a quantification model does not need to rely on exact number information, but it can predict the correct answer using the gist of the queried property in the image, thus simulating the human ANS.

5.2.1 Artificial dataset

Images. In order to focus on the quantification task, barring any effect from data preprocessing, we create an artificial dataset with clear visual properties (see Figure 5.7).

This dataset consists of images with 1 to 16 circles of 15 different colors, and we generate all possible combinations of different numbers of circles (from 1 to 16) with all possible combinations of colors. Figure 5.7 presents one of the images in the dataset.

Image representation. In order to avoid effects from visual pre-processing, the dataset is presented to the quantification network with (automatically produced) gold standard information about image segmentation and object identification. Each scene is represented by a grid with 16 cells and each cell can contain a circle (see Figure 5.7).

Instead of calculating a visual vector of a circle with a specific color (the approach described in the previous chapter), we chose to associate each circle-color combination with 20-dimension vectors that are randomly generated and normalized to unit norm. The circles are identical in shape and size, and this means that the differences between vectors express the color property of the objects. The dataset should not include 'confusable' objects and this requirement is solved by constraining the vectors to have a low pairwise similarity. The cosine similarity between the vectors of two different circles does not exceed 0.7. On the other hand, to prevent overfitting, we add a small amount of noise to all vectors, generated for each dimension from a Gaussian distribution with mean 0 and variance 1/5 of the original variance of that dimension. Intuitively, the Gaussian noise simulates natural variations in a given property, e.g., two lemons being of slightly different shades of yellow. This is applied to both training and test data.

Finally, we can have *empty cells* in a visual scene, viz. parts of the image with no object in it (e.g., in Figure 5.7 there are 5 empty cells.) These are similarly represented by a vector, randomly generated so that it is orthogonal to all the other object vectors. The chosen vector is orthogonal because we consider that there is no similarity between an empty cell and a cell with a circle. The dot product between the corresponding vectors is zero, which will not influence our models.

Queries Each image in the dataset is associated with a *query*, i.e., the property we want to quantify over, and the task of the model is to associate the correct *quantifier* with the query for the image. For instance, the query associated with the image in Figure 5.7 is *green* and the correct quantifier is *some*.

Some encodes "at least one but not all circles have color X", *all* encodes "all cir-

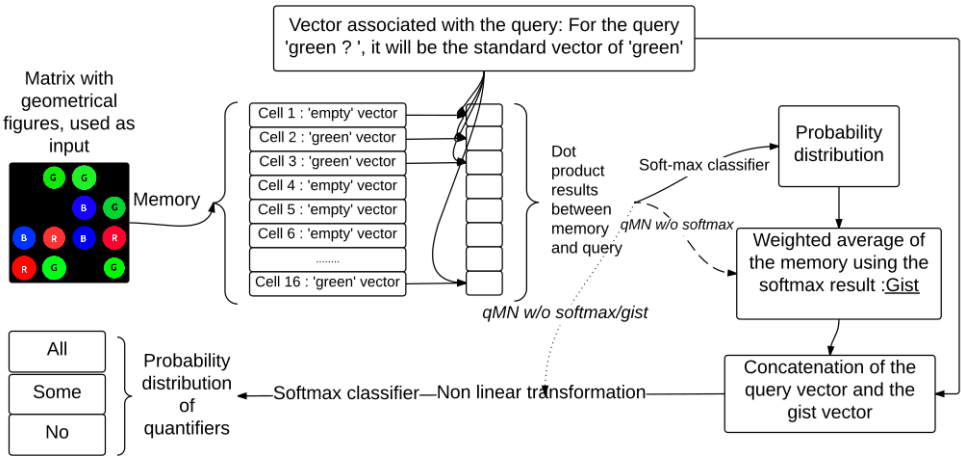

Figure 5.8: Quantification Memory Network model

cles have color X" and *no* "no circle has color X". Our dataset contains 5000 <image, query, quantifier> datapoints split equally amongst the three quantifiers. Although the *all* quantifier generates fewer images than *no* and *some*, it is possible to create balanced data by producing noisy variations of a same image. This dataset will be used to evaluate our models and analyze their behavior.

5.2.2 Models

We experiment with a variety of models, described below.

Quantification Memory Network (qMN):

This model is a simple version of the model presented in Section 3.2 and used in the previous chapter. The changes of the model are caused by the different datasets and by the different task. On the other hand, the model follows the same principles, and it is based on the same ideas.

As shown in Figure 3.8, the model consists of a memory with 16 slots, one for each image cell. It computes the dot product between each memory slot and the vector query, obtaining 16 scores, which are then fed into a softmax classifier to derive a valid probability distribution. These normalized scores are used to derive the "gist" of the image (a 20-D vector), by computing a weighted sum over cell vectors in the memory slots, where the weights are taken from the probability distribution that is output by the softmax classifier. Finally, a non-linear transformation with a ReLU activation is applied over

the concatenation of the "gist" and query vectors. The vector dimensionality is reduced to 3 by linear transformation and a softmax classifier is applied on top of that, deriving a probability distribution over the three quantifiers. The "gist" vector is an aggregate of the memory, and information about individual objects is lost, such that the model is not able to count. However, the similarity between the "gist" and the query reflects the ratio (rather than the exact number) of objects of that color in the image. To make this explicit, in the case of 'all' , the gist and query vectors will be almost identical, in the case of 'no' there will be hardly any trace of the query in the gist, making them different, and in the case of 'some' query and gist will be somewhat similar.

Counting model:

The complexity of the task is not very high and we can predict that a simple rule-based counting model will reach 100%.

Instead, we try to define a neural network model, which can be compared more easily with qMN and has a design based on counting.

Despite the logical interpretation of quantifiers as ratios between two magnitudes, it is unclear whether this logical operation is easily learnable in a visual model. In this setup, we build for each image a 16-D feature vector, one dimension for each of the 15 colors plus one for the empty cell. To each dimension we assign a value encoding the frequency of the color in the image scaled by the similarity of that color to the query (recall that, because of the added Gaussian noise, a given yellow circle may not be identical to the query *yellow*). This way, the quantity of objects of a given color is encoded in the dimensions of the vector as if the model were counting.

The query is represented by a one-hot 16-D vector, encoding the color the model is asked to quantify over.

The feature and query vectors are concatenated. As in the qMN model, we then apply a linear transformation followed by a ReLU activation and a softmax classifier.

Recurrent Neural Network (RNN):

As an alternative model with a visual memory, we also implement an RNN that uses the hidden state to encode information about the image's gist. At each timestep, the RNN receives as input first the query vector followed by each of the 16 objects vectors.

At the last timestep, the hidden layer is fed to a linear transformation, reducing its size to 3, on top of which a softmax classifier is applied to obtain a probability distribution over the quantifiers. As opposed to the qMN, the RNN does not explicitly model the similarity between the query and the color of the objects in the image.

We further present a small ablation study to track the effect of the various components of qMN.

To analyse the effects of the MN components, we have evaluated two other versions of it: the simplest version **qMN-softmax/gist** and the version without SoftMax **qMN-softmax** (Highlighted in Figure 3.8.)

qMN-softmax/gist is designed to show if a model can learn only from the similarities between the query and the the image without being exposed again to the query and without building a generic representation of the input. The input for the nonlinear transformation is a vector which represents the similarity between each cell and the query vector and it has a dimension equal to the number of slots in the memory. Due to the nature of the input, we can not expose it to the query again in a meaningful way as is done for qMN.This model should learn how to count the positive matches relative to the negative matches.

qMN-softmax skips the softmax activation function over the dot products. We made this change because we wanted to observe what is happening if we are not so drastic with the dot product. This variation is not so biased towards the positive matches and the results of the dot product are not transposed in a probability distribution vector (which is done by the softmax classifier in the first model).

All models are trained with cross-entropy to predict the correct quantifier.

5.2.3 Setup

We divide the 5K data points into training, validation and test set (70%, 10% and 20%). We test the models in 3 experimental setups. In all setups, given a datapoint <image, query, quantifier>, the models receive the image and the query and are evaluated in terms of accuracy in predicting the quantifier.

The first setup, **familiar**, is the simplest, and tests whether models are able to quantify

previously observed ("familiar") colors and quantities. In the **unseen quantities** setup, we create training and test sets so that there is no overlap with respect to the number of objects in the image: 4, 9 or 13 objects are used at test time and all other quantities at training/validation time (i.e., 1-3, 5-8, 10-12, 14-16). Finally, in the **unseen colors** setup, we make sure training and test sets differ with respect to objects' color: The models are trained/validated on 10 colors and tested on 5 additional, unseen colors.

We expect that the use of the gist in our model, which implements **global** quantification over objects of a certain property, will allow it to generalize well when tested against unseen quantities.

In addition to these three experimental setups, we tested both a linear and non-linear version of the **Counting** and **qMN** models. We wanted to verify the hypothesis that non-linearity is crucial to classifying *some*, by virtue of its similarity to the XOR operator. (To see the analogy between the two operators, consider the simplest case of having just two objects: then *some* is true iff the two objects are different.) Indeed, we found that the linear versions were outperformed by the non-linear ones, and the results presented in the next section were calculated using non-linearity.

5.2.4 Results

As shown in Table 5.6, having exact number information is not necessary for learning to quantify: The **qMN** model, which does not explicitly count, is more accurate than the **Counting** model in all test conditions. Even though both models outperform the **RNN** model when tested on unseen number of objects, only the **qMN** model truly generalizes the learnt quantification operation.

The performance of all models drops when tested on unseen colors, though **qMN** still performs best and the decrease in performance in **Counting** is much worse than in the **qMN** model (-53.7 vs. -34). The results for the ablation models in Table 5.6 show that both the softmax and the "gist" are crucial elements of the model and removing them causes significant performance drop in all test conditions.

By looking at the confusion matrices for the **qMN** model (Table 5.7) we observe that there is generally more confusion between *no* and *some* than in pairs involving *all*; the gist

Models	familiar	unseen quantities	unseen colors
RNN	65.7	62.0	49.7
Counting	86.5	78.4	32.8
qMN	**88.8**	**97.0**	**54.9**
-softmax	85.9	66.6	54.4
-softmax/gist	51.4	51.8	44.4

Table 5.6: Model accuracies (in %).

familiar			
	some	all	no
some	255	1	62
all	0	335	0
no	49	0	298
unseen quantities			
	some	all	no
some	321	8	5
all	0	333	0
no	6	0	327
unseen colors			
	some	all	no
some	115	13	206
all	64	166	103
no	63	2	268

Table 5.7: Confusion matrices.

for *some* is an average of potentially several different colors, and thus less straightforwardly interpretable. In the 'familiar' test, most of the errors come from situations in which the model confused "some" with "no" (see Table 5.7) and the image contains just one or at most two occurrences of the queried color. Hence, the increase in performance from the familiar to the unseen quantity test (+8.2) is due to the absence of very small cardinalities in the image (the least is four items.)

As for *all*, in both the 'familiar' and the 'unseen quantities' conditions it is nearly always classified correctly. This is to be expected because in this case, the "gist" computation produces a vector which should be almost equivalent to the query (minus the effect of noise).

When moving to unseen properties, the performance decreases, which is expected be-

cause of the increasing complexity of the dataset, but the proposed model still outperforms the other models by a large margin.

5.2.5 Conclusions

Compared with the experiments presented in 4, this chapter presents an analysis at a more theoretical level of quantification questions. Even though the experimental setup is different, the results show that models based on memory networks are suitable for this task because they reach the best performances compared with other state-of-the-art models.

Furthermore, we were able to tentatively replicate results in Cognitive Science showing the primacy of ANS over counting for quantifier learning in humans, with a computational model learning the meaning of *some*, *no*, and *all* from visual scenes. We show that counting is not necessary in order to reach a high precision for quantification questions.

Because of the reduced complexity of the experimental setup, we can afford to check if there are some redundant parts in the proposed model by ablation. The results show that all the components are important and that by eliminating the gist, the model no longer gives the expected predictions.

Moreover, we aim to study whether NN models can provide us with vector representations of quantifiers needed in many NLP and CV tasks. Furthermore, though we believe the model can scale to larger magnitudes since the number of memory cells is parametric, we will test our conjecture to find evidence for it.

Chapter 6

Conclusions and further work

The work presented in this thesis represents the first study on visual question answering with quantification questions. This is a very challenging task and it requires an architecture which is capable of reasoning and not just finding correlations in the dataset. An important result of this study is the creation of two viable datasets which can be applied for different tasks: prediction of the correct number of objects, object identification, property identification, etc.

This study shows possible directions, but the topic is open and it can show very important insights which will make the gaps between linguistics, computational vision and neuroscience smaller.

There are two directions for improvement on this task, but they are very related. The first direction is to improve the dataset. The final goal is to work with real visual scenes, annotated by humans because this will eliminate any doubts regarding the dataset. It can lead also to a better comparison with the human experience on this task because we are exposed to real-world scenes, not to bounding boxes. The second direction is to implement different models that can be applied to the task. There is a lot of work on the field of deep learning and especially on visual question answering and it is interesting to check if the state-of-the-art models can be adapted for this task and, if not, what is the reason behind it.

The **qMN** model and its variations presented in this thesis show that they are built following a good philosophy, but because of their flexibility some parts can be changed and this might increase the performance. Also, a better understanding of the complex

models used in Chapter 4 can lead to a complex architecture that manages to find more non-superficial relations in the dataset.

Another goal of this task is to generate a quantifier representation with visual and linguistic input. The human representation of the quantifiers is not based only on linguistic information and this encourages a multimodal model that can generate this representation.

Another research direction worth exploring and very related to this study is how to predict quantifiers using grounded knowledge (we know that *most dogs have four legs* without seeing all the dogs in the world). This direction involves a better understanding of the reasoning behind semantic memory and how it combines with episodic memory.

Chapter 7

Appendix

7.0.1 Objects

The list of the 161 objects we have used in our dataset is below.

acorn, antelope, ape, baboon, backboard, badger, bag, ball, basketball, bat, bathtub, bear, bench, bird, bison, blackboard, boa, bonsai, bottle, bovine, box, bread, bridge, buckle, building, button cabinet, cake, cat, cavy, chair, cheese, cheetah, chili, chimpanzee chinchilla, civet, cocktail, comb, cow, cross, daisy, deer, dog, drawer, drum, ferret, ferris wheel, field, filter, flag, flan, fondue, fork, fox, frog, fruit, gazelle, gerbil, giraffe, gnu, goat, gorilla, grass, grassland, hamster, hare, hominid, honey, hoop, horse, hyena jaguar, kangaroo, ketchup, key, keyboard, koala, lamp, lemming, lemonade, lemur, lion, liqueur, livestock, lizard, lollipop, macaque, mallet, manatee, margarine, marimba, mink, monkey, mouse, mousse mule, orangutan, orca, otter, panda, panther, parfait, pasta, paste pizza, platypus, pony, pool, porpoise, porridge, possum, pot pudding, puppy, rabbit, raccoon, rack, racket, rat, rice, rodent roller coaster, ruler, sail, salad, salmon, sauce, sauerkraut, saute sheep, simian, ski, skunk, sofa, spider, squirrel, stew, stick strawberry, sugar, sunflower, syrup, table, tea, tiramisu toilet seat, towel, tower, transporter, van, vegetarianism, weasel, whale, wildcat, window, wine, wombat, yolk and zebra.

7.0.2 Properties

The 24 ImageNet properties we have used are the following:

Color:	black, blue, brown, gray, green, orange, pink, red, violet, white, yellow
Pattern:	spotted, striped
Shape:	long, round, rectangular, square
Texture:	furry, smooth, rough, shiny, metallic, wooden, wet

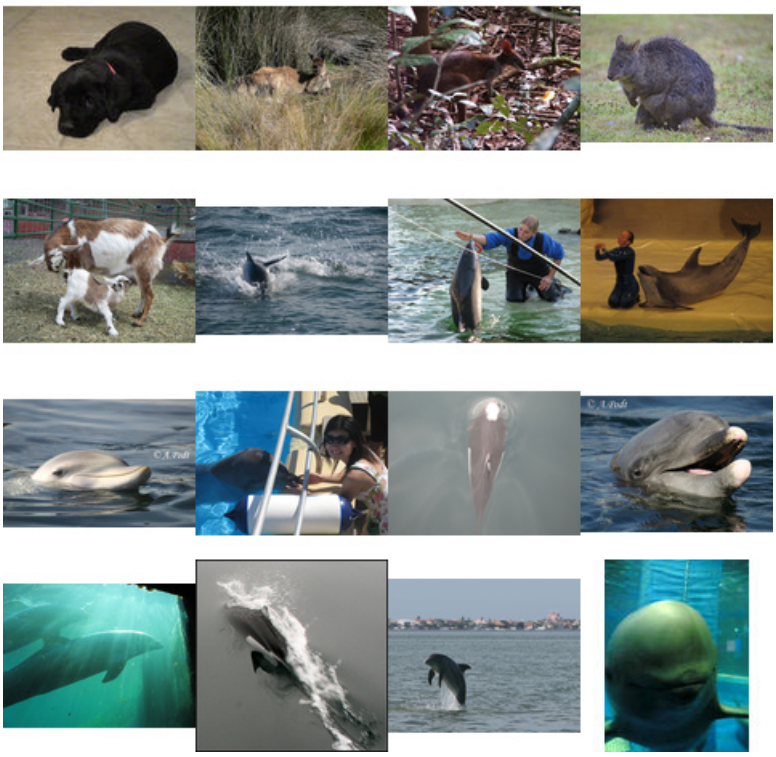

Figure 7.1: Scenario 1 No porpoises are gray - correct

7.0.3 Examples of some scenarios with good and bad predictions for each dataset

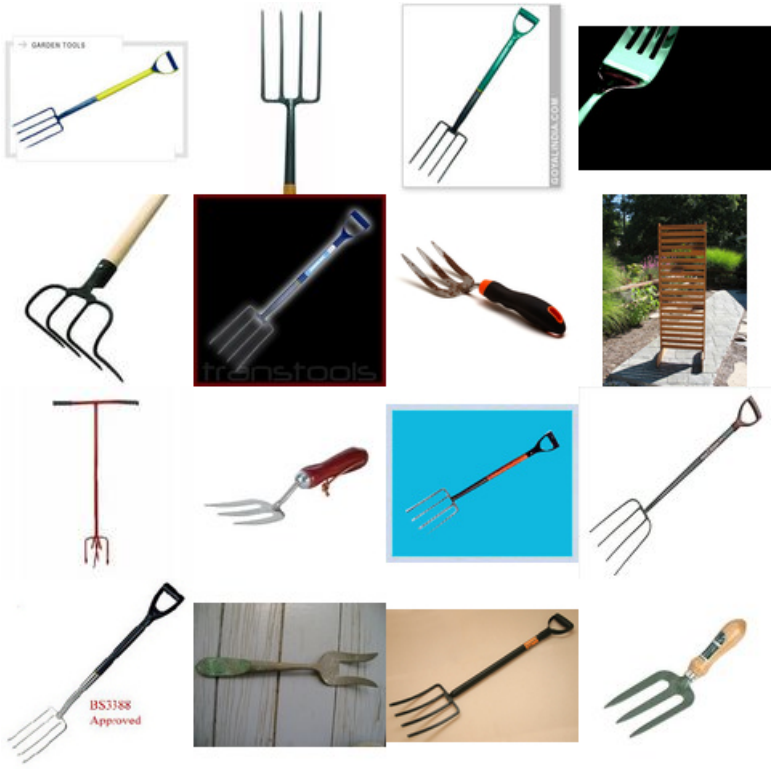

Figure 7.2: Scenario 2 Some forks are long - correct

Figure 7.3: Scenario 3 No pots are spotted - correct

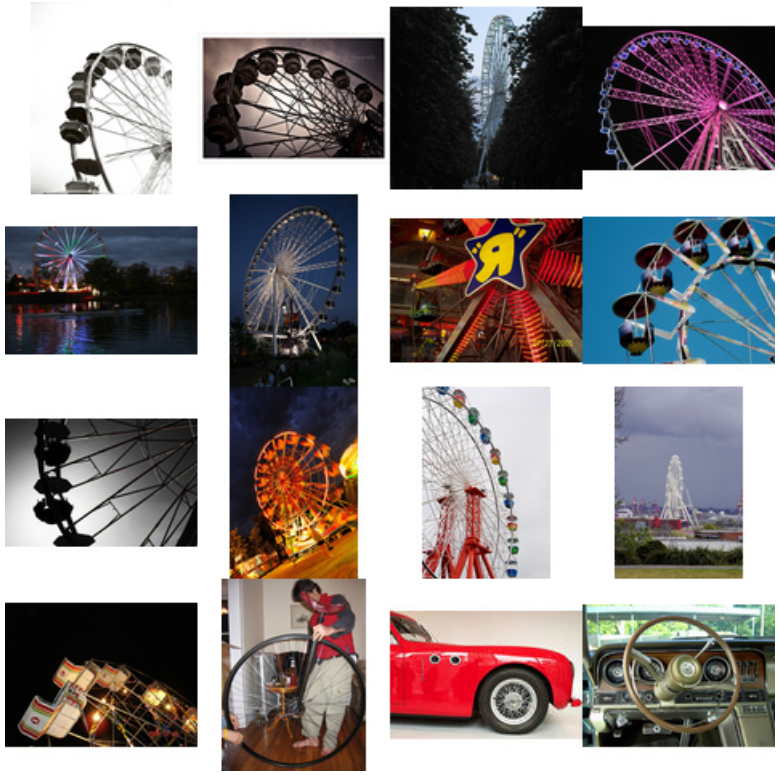

Figure 7.4: Scenario 4 Some ferris-wheel are red - wrong prediction *few*

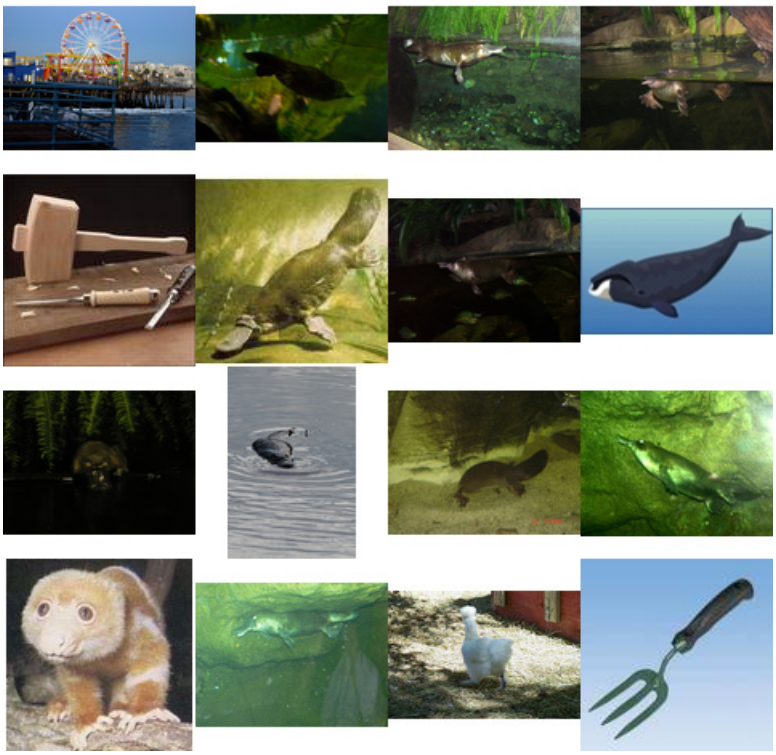

Figure 7.5: Scenario 5 Few platypuses are spotted - wrong prediction *no*

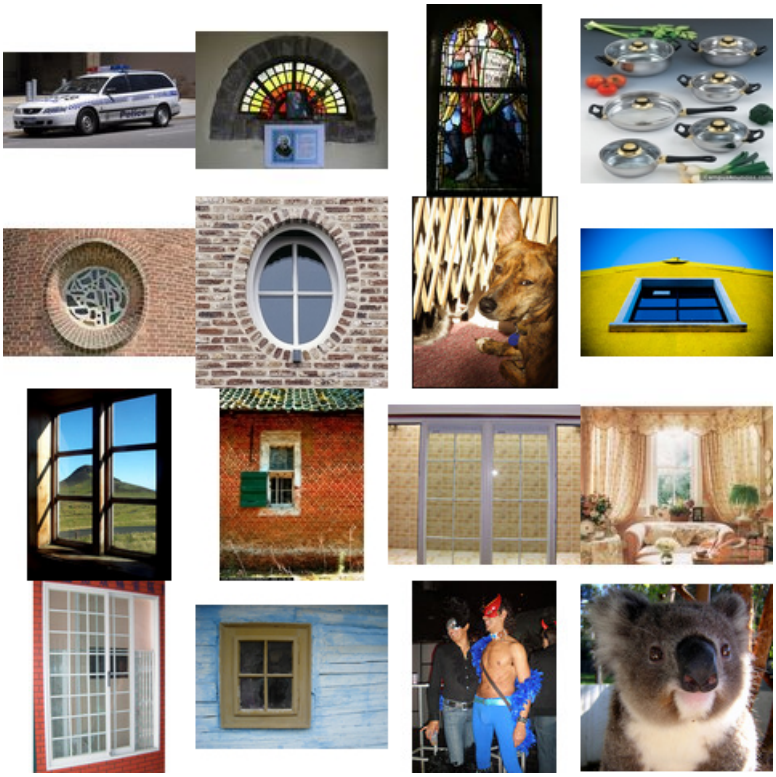

Figure 7.6: Scenario 6 Some windows are yellow - wrong prediction *few*

Bibliography

[Antol et al., 2015a] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015a). Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

[Antol et al., 2015b] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015b). Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

[Baroni et al., 2009] Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

[Baroni et al., 2014] Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

[Cann, 1993] Cann, R. (1993). *Formal semantics: an introduction*. Cambridge University Press.

[Dehaene and Changeux, 1993] Dehaene, S. and Changeux, J. (1993). Development of elementary numerical abilities: A neuronal model. *Journal of Cognitive Neuroscience*, 5.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

[Desmond Elliott and Lazaridou, 2016] Desmond Elliott, D. K. and Lazaridou, A. (2016). Multimodal learning and reasoning.

[Feigenson et al., 2004] Feigenson, L., Dehaene, S., and Spelke, E. (2004). Core systems of number. *Trends in cognitive sciences*, 8(7):307–314.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[Herbelot and Vecchi, 2015] Herbelot, A. and Vecchi, E. M. (2015). Building a shared world: Mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. `https://www.aclweb.org/anthology/W/W13/W13-0204.pdf`.

[Hurewitz et al., 2006] Hurewitz, F., Papafragou, A., Gleitman, L., and Gelman, R. (2006). Asymmetries in the acquisition of numbers and quantifiers. *Language learning and development*, 2(2):77–96.

[Karpathy and Fei-Fei, 2015] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

[Khemlani et al., 2009] Khemlani, S., Leslie, S.-J., and Glucksberg, S. (2009). Generics, prevalence, and default inferences. In *Proceedings of the 31st annual conference of the Cognitive Science Society*, pages 443–448. Cognitive Science Society Austin, TX.

[Kiros et al., 2014] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[LeCun et al., 1995] LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276.

[Li et al., 2016] Li, C., Zhu, J., and Zhang, B. (2016). Learning to generate with memory. *arXiv preprint arXiv:1602.07416*.

[Lin et al., 2014a] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014a). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer.

[Lin et al., 2014b] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C. L. (2014b). Microsoft coco: Common objects in context. In *Microsoft COCO: Common Objects in Context*.

[Linzen et al., 2016] Linzen, T., Dupoux, E., and Spector, B. (2016). Quantificational features in distributional word representations. *c2016 The* SEM 2016 Organizing Committee. All papers c2016 their respective authors. This proceedings volume and all papers therein are licensed under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons. org/licenses/by/4.0*, page 1.

[Malinowski and Fritz, 2014] Malinowski, M. and Fritz, M. (2014). Towards a visual turing challenge. *arXiv preprint arXiv:1410.8027*.

[McRae et al., 2005] McRae, K., Cree, G. S., Seidenberg, M. S., and McNorgan, C. (2005). Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Montague, 1973] Montague, R. (1973). The proper treatment of quantification in ordinary english. In *Approaches to natural language*, pages 221–242. Springer.

[Olm et al., 2014] Olm, C. A., McMillan, C. T., Spotorno, N., Clark, R., and Grossman, M. (2014). The relative contributions of frontal and parietal cortex for generalized quantifier comprehension. *Frontiers in human neuroscience*, 8.

[Pantel, 2005] Pantel, P. (2005). Inducing ontological co-occurrence vectors. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 125–132. Association for Computational Linguistics.

[Paperno et al., 2016] Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. (2016). The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

[Peters and Westerståhl, 2006] Peters, S. and Westerståhl, D. (2006). *Quantifiers in language and logic*. Oxford University Press.

[Rajapakse et al., 2005] Rajapakse, R., Cangelosi, A., Conventry, K., Newstead, S., and Bacon, A. (2005). Grounding linguistic quantifiers in perception: Experiments on numerosity judgments. In *Proceeding of the 2nd Language and Technology Conference*, Poland.

[Ren et al., 2015] Ren, M., Kiros, R., and Zemel, R. (2015). Image question answering: A visual semantic em- bedding model and a new dataset. In *In International Conference on Machine Learningt Deep Learning Workshop*.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet

large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

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

[Sorodoc et al., 2016] Sorodoc, I., Lazaridou, A., Boleda, G., Herbelot, A., Pezzelle, S., and Bernardi, R. (2016). look, some green circles!: Learning to quantify from images. In *Proceedings of the 5th Workshop on Vision and Language at ACL*.

[Sukhbaatar et al., 2015a] Sukhbaatar, S., Szlam, A., Synnaeve, G., Chintala, S., and Fergus, R. (2015a). Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*.

[Sukhbaatar et al., 2015b] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015b). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[Troiani et al., 2009] Troiani, V., Peelle, J. E., Clark, R., and Grossman, M. (2009). Is it logical to count on quantifiers? dissociable neural networks underlying numerical and logical quantifiers. *Neuropsychologia*, 47(1):104–111.

[van Benthem, 1986] van Benthem, J. (1986). *Essays in Logical Semantics*. Reidel Publishing Co, Dordrecht, The Netherlands.

[Vedaldi and Lenc, 2015] Vedaldi, A. and Lenc, K. (2015). *MatConvNet – Convolutional Neural Networks for MATLAB*. Proceeding of the ACM Int. Conf. on Multimedia.

[Verga et al., 2016] Verga, P., Neelakantan, A., and McCallum, A. (2016). Generalizing to unseen entities and entity pairs with row-less universal schema. *arXiv preprint arXiv:1606.05804*.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

[Weston et al., 2014] Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

[Xiong et al., 2016] Xiong, C., Merity, S., and Socher, R. (2016). Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*.

[Xu and Saenko, 2015] Xu, H. and Saenko, K. (2015). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

[Zhou et al., 2015a] Zhou, B., Tian, Y., Suhkbaatar, S., Szlam, A., and Fergus, R. (2015a). Simple baseline for visual question answering. Technical report, arXiv:1512.02167, 2015.

[Zhou et al., 2015b] Zhou, B., Tian, Y., Sukhbaatar, S., Szlam, A., and Fergus, R. (2015b). Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*.