

POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Semantics-Aware VQA

*A scene-graph-based approach to enable
commonsense reasoning*

Supervisor

Prof. Elena Baralis

Dott. Andrea Pasini

Candidate

Andrea Detommaso

December 2020

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 1.1 | VQA applications | 6 |
| 1.2 | Objective | 8 |
| 1.2.1 | Thesis outline | 10 |
| 2 | Preliminary knowledge | 11 |
| 2.1 | Visual question answering | 11 |
| 2.1.1 | Datasets for VQA | 13 |
| 2.1.2 | Evaluation metrics | 15 |
| 2.2 | Convolutional Neural Networks | 16 |
| 2.3 | Recurrent Neural Networks | 18 |
| 2.4 | Long Short Term Memory | 20 |
| 2.5 | Scene graphs | 24 |
| 2.6 | External commonsense knowledge base | 25 |
| 2.7 | Natural Language Processing | 26 |
| 2.7.1 | NLP pipeline | 27 |
| 3 | Related works | 28 |
| 3.1 | Attribute extraction and LSTM | 28 |
| 3.2 | Usage of external Knowledge bases | 29 |
| 3.3 | Graph attention networks | 31 |
| 3.4 | Contribution | 32 |
| 4 | Semantics-aware VQA | 34 |
| 4.1 | Preliminary definitions | 36 |
| 4.2 | Scene graph embedding | 39 |
| 4.2.1 | Embedding computation | 39 |
| 4.2.2 | Scene graph weighting | 42 |
| 4.3 | Attribute extraction | 44 |
| 4.3.1 | Attribute computation | 45 |
| 4.3.2 | Attribute weighting | 45 |
| 4.3.3 | Extra attributes computation | 47 |

| | | |
|----------|---|-----------|
| 4.4 | Time series encoding | 49 |
| 4.5 | LSTM backbone | 51 |
| 5 | Dataset, evaluation and implementation details | 53 |
| 5.1 | VQA dataset | 53 |
| 5.1.1 | ConceptNet | 56 |
| 5.2 | Evaluation and metric | 57 |
| 5.3 | Implementation Details | 58 |
| 6 | Results | 59 |
| 6.1 | Comparison of the proposed methods | 60 |
| 6.2 | Qualitative results | 66 |
| 7 | Conclusions and future works | 75 |

Abstract

In this work, we investigate a novel approach to the Visual Question Answering task, a research area where a system has to answer a question expressed in natural language about an image. Most of the existing works have a standard limitation: they are bounded by the poor ability to reason about the image's context. Therefore, we consider the use of scene graphs derived from images: a synthetic representation of an image where graph nodes represent objects entities and the graph's edges show object relationships. Furthermore, we investigate the use of an ontological knowledge base as a way of improving the reasoning capacity of the system. The dataset used for our experiments is Visual7W, a collection of 40'000 questions related to the Visual Genome dataset. These pictures are enriched with scene graphs and further annotations. Our empirical studies show how scene graphs can enhance the reasoning capacity of the system, especially in spatial terms. Moreover, the work shows how the usage of an external knowledge base improves the capacity of our system to infer the image's context.

Chapter 1

Introduction

In the last decade, we observed a sensational development in Machine Learning and Artificial Intelligence fields. Crowd-sourcing permitted fast advancements in new research areas that started attracting great interest. Indeed, the availability of high quality labeled data provided by crowd-sourcing platforms, favored the discovery of new machine learning models. The ones related to computer vision and Natural language processing shifted the focus on the possibility of creating systems capable of answering natural language questions about surrounding words. The studies conducted in this thesis examine in depth one of these multidisciplinary research fields, called Visual question Answering. VQA is an Artificial Intelligence research problem, which involves Natural Language Processing, Computer Vision and Knowledge Reasoning. The main purpose of this task, is answering text-based questions about the contents of images. Questions can be related to the object's features, relationships between objects and the image's context. Current architectures obtain acceptable results, indeed they can reasoning on a wide range of question/answer pairs. These models obtain a high accuracy on the datasets with which they are trained on. Nevertheless, the majority of them are limited by the poor ability of reasoning about the image's context, picking an answer among a limited number of options. A robust VQA system should reason about the image's

context and should learn a commonsense knowledge that humans give for granted. Humans can easily locate objects inside an image, understanding their positions in the space and the relationships that exist among them. With all these information, we can understand the image's context. This task is not trivial for an AI system. A question asked in a different way or an unclear image can be misleading for the algorithm. Therefore, this work aims at overcoming these limitations by enriching our model with external data that could improve the image's understanding.

1.1 VQA applications

Visual Question Answering has many real life and research applications. It provides information about real word and web images. Therefore, it can be integrated into image retrieval systems, without using meta data or tags. VQA can also be used for improving human interaction with computers, as it improves visual content comprehension. *Abhishek Das et al* for example, developed the chat-bot *Visual Dialog* [8]. The latter is a AI model capable of holding a meaningful natural language dialog about visual content with humans. Figure 1.1 shows an example of conversation with Visual Dialog about an image. The latter depicts a cat drinking water out of a coffee mug. The AI system is questioned about the visual features contained in the picture (e.g., Which color is the mug?).

This type of visual intelligence system can be exploited for many purposes: they can be integrated in the already existent AI assistant, like Alexa, asking questions about visual contents (e.g., security cameras or baby monitors). Moreover, these AI models can assist data analysts making decisions about surveillance data[24]. With the same purpose, these systems can be used for answering questions related to a specific domain, like the ones gathered by *Bigham et al.* [3] to help visually impaired people.

Regarding the research area, VQA is a way of evaluating AI models for both scene understanding and reasoning capacity. Indeed, this task requires visual recognition,

logical reasoning and external knowledge about the world. *Abhishek Das et al.* for example studied in their work [7] how humans recognize objects or scenes and how human attention works, analyzing the differences with the reasoning process of an AI system. Figure 1.2 depicts an example of this work, where the attention system highlights the pixels of the image that answer a specific question (e.g., blue for the tennis racket and red for the floor).

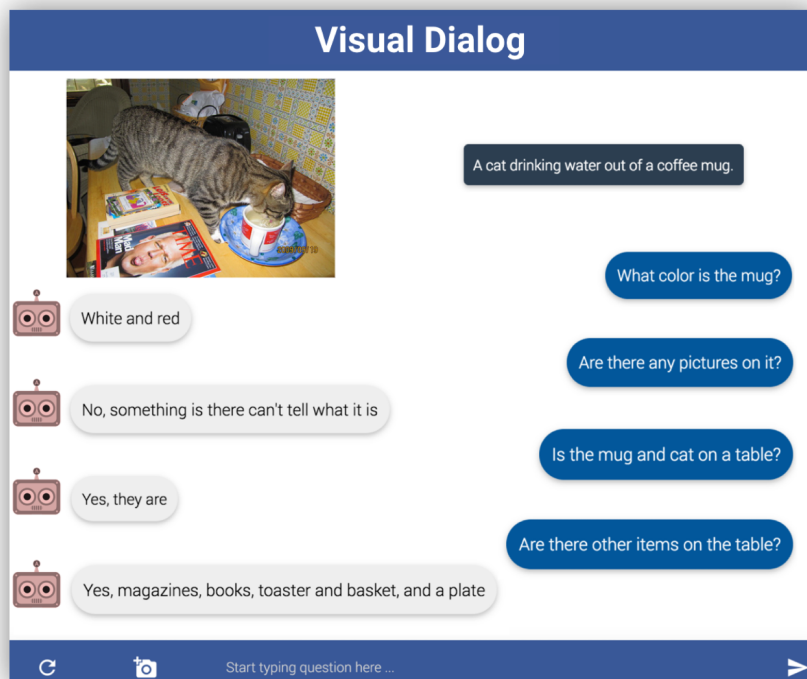


Figure 1.1: A dialog with the chat-bot *visual dialog*. It is an example of how the VQA task can improve human-machine interaction [8]

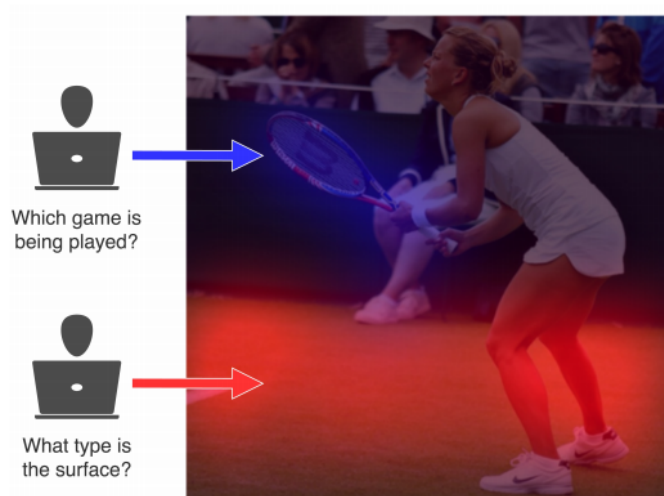


Figure 1.2: A research area where VQA task is involved. The figure depicts human attention regions concerning different questions [7]

1.2 Objective

This work aims at improving the way a VQA system makes commonsense inferences. The existing state-of-the-art architectures achieve acceptable results, especially on questions that do not require a high grade of reasoning. Nevertheless, enhancing the system’s ability to reason about the image’s context is still an open research problem. To achieve this goal, we exploit graph-structured external knowledge: instead of considering exclusively visual feature’s extraction, we propose both to use scene graphs related to pictures and an external commonsense knowledge base to extract new concepts and measure the relevance of the extracted knowledge concerning to question context. Concepts extracted from the images and the ones related to every Question/Answer couple are used to train a LSTM Network. The input time series take into account both question/answer concepts and the extracted knowledge.

In our work, the multiple choice task is turned into a binary classification fashion, assigning to the right answer the label “1” and the label “0” to the other ones.

At test time, the right answer is considered the one characterized by the highest probability of belonging to “1” class.

The architecture is trained with Visual7W dataset (see Section 2.1.1). We made this choice because of the possibility of using scene graphs related to the dataset’s images. The model is trained only with *why* questions, the ones involving common-sense knowledge the most. In proof of this, human performance without images on this subset is remarkably high, indicating that many why questions encode a fair amount of common sense that humans are able to infer without visual cues.

Our results demonstrate how scene graphs can infer essential information of images, improving the reasoning capacity of the system, especially in spatial terms. Furthermore, the work shows how the usage of an external knowledge base improves the capacity of our system to infer the image’s context. On the other hand, this VQA-system has been created uniquely for answering *why* questions. Moreover, even if our technique shows an improvement in terms of reasoning capacity, existing architectures adopt more sophisticated models that still achieve higher performances. Nevertheless, after the study conducted in this work we will improve performances by integrating our intuitions into further types of VQA-systems.

1.2.1 Thesis outline

The following chapters in this thesis are organized as follows.

Chapter 2 (Preliminary Knowledge) provides the background knowledge required to understand the work and the implementation.

Chapter 3 (Related works and contributions) goes over some of the most significant related works, which constitute a start point for our proposed implementation. It briefly analyzes their advancements and their limitations. Moreover, it explains the scientific contributions that our model provides.

Chapter 4 (Semantic aware VQA) provides a formal description of our architecture, describing the modules of which it is composed.

Chapter 5 (Dataset) treats the preliminary analysis made on the dataset, the evaluation methods adopted and the experimental setting.

Chapter 6 (Results) goes through an in-depth analysis of results, under a qualitative and quantitative point of view.

Chapter 2

Preliminary knowledge

In the following sections, we will first analyze the main characteristics of VQA models, then the different dataset openly available to work with this task. Afterward, we will provide the main techniques for a quantitative evaluation. Then, we will introduce the basic notions related to our architecture: the machine learning network we adopted for this work and the external data exploited for our method: Scene graphs and ConceptNet.

2.1 Visual question answering

As previously mentioned, VQA is a multidiscipline that combines Natural Language Processing and Computer vision. Given a free-form question expressed in natural language and related to an image, the VQA-system has to produce an answer. The image is a generic picture, that can represent any kind of context. The answers instead, can be expressed in two main forms that divide this research field in two groups:

- **Open-ended VQA:** as Figure 2.1 (b) shows, the model has to produce a free-form, natural language answer.

- **Multiple-choice VQA:** as Figure 2.1 (a) depicts, the system has to choose among multiple options.



Q: Why is the flowers and vase casting a shadow?

- A. The sun
- B. Their position
- C. They are in the way
- D. Lights



Q: What animal is in this picture?

A: A giraffe

Figure 2.1: Figures show an example of the two main types of VQA

The existing state-of-the-art architectures distinguish themselves for the algorithms, the architectures and the type of VQA treated. Nevertheless, the systems are built using a similar logic.

As Figure 2.2 shows, a typical VQA-system is composed of a series of key modules:

- **Input:** The system takes two inputs: one image and the related question.
- **Feature extractors:** These modules are used for extracting the key features from the inputs. Then, these information are projected into a higher-dimensional space: every feature is embedded into a vector, belonging to a R_n space, where n is equal to the vector size.
- **Algorithm:** Input features are combined and used to train a machine learning algorithm.

- **Classifier:** Finally, the previous output is used for predicting the correct answer.

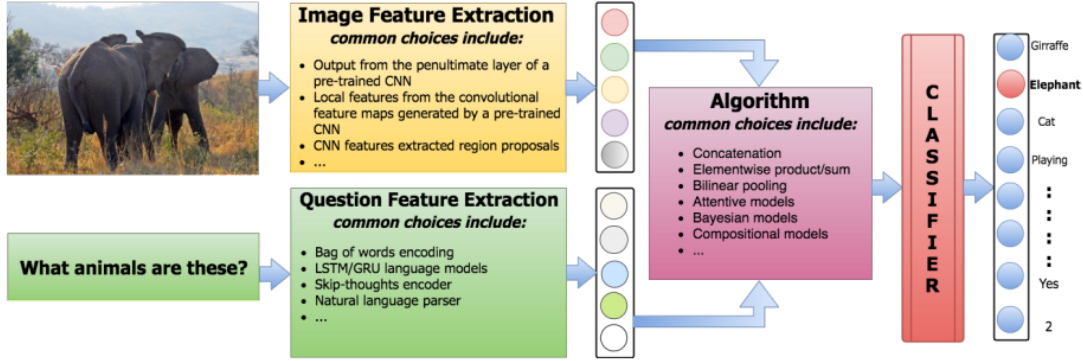


Figure 2.2: Standard VQA architecture [11]

2.1.1 Datasets for VQA

Starting from 2014, many VQA datasets have been created. These datasets are employed for training and evaluating a VQA system. A VQA dataset must possess some key characteristics. It needs to be large, in order to contain different images and questions related to many contexts. Otherwise, the algorithm could be biased by the distribution of the questions or answers. Furthermore, its evaluation scheme has to be hard to outflank, it must be a proof that the system is able to answer many types of questions.

The main datasets for VQA are:

- **DAQUAR[22]:** It is one of the smallest VQA datasets available. Indeed, it

consists of 6795 training and 5673 testing Question/Answer couples. Images are related to the NYU-DepthV2 Dataset and they contain exclusively indoor scenes. This peculiarity limits the variety of questions related to the pictures.

- **COCO-QA[27]**: COCO-QA contains 78,736 training and 38,948 testing Question/Answer pairs. The latters are created thanks to a Natural Language Processing algorithm that extracts them from the COCO image captions. The Answers are characterized by a single word and there are 435 possible answers. Because of this characteristic, the evaluation procedure is very simple.
- **The VQA Dataset[1]**: The dataset is composed of real images from COCO and cartoon images. Every picture is related to three questions. The latters are characterized by 10 possible answers. Many of them are subjective, indeed they do not have a single objective answer. Simple algorithms achieved 49.6 % accuracy on the dataset. This score was obtained without exploiting the images, but using only the question.
- **FMIQA[10]**: The Freestyle Multilingual Image Question Answering (FMIQA) dataset is composed of images from COCO. Questions/Answers pairs are generated by humans. Furthermore, in this case the answers can be represented by full sentences. Because of this characteristic, the answers can't be evaluated with common metrics.

Visual7W

Visual7W [15] contains 47,300 images from Visual Genomes. Questions belong to seven different categories: How, Who, What, Where, When, Who, Why and Which and they can be divided in two types: *telling* and *pointing*. The first one is related to the first 6 categories and the answer is text-based. The *pointing* questions are the ones that begin with *Which*, for these questions the algorithm has

to choose the correct object among the ones in the picture. The dataset contains both open-ended and multiple choice questions. In the second case, the algorithm has to choose between four options, written in natural language. Choosing among options is easier than generating a natural language answer. On the other hand, the usage of multiple choice questions is challenging because they consist of at least three answers that are possible for the given question.

As Table 2.3 shows, the main differences between these datasets rely on answer diversity, the answer’s type and length. Another difference consists in the type of annotations collected for every image. Visual Genome for example, includes also regions description and scene graphs, data that we estimated to be useful for training the VQA systems.

| | DAQUAR | COCO-QA | COCO-VQA | FM-IQA | Visual7W | Visual genome |
|-----------------------|---------------------|-----------|-----------|---------|-----------|---------------|
| Total Images | 1,449 | 123,287 | 204,721 | 120,360 | 47,300 | 108,000 |
| QA Pairs | 12,468 | 117,684 | 614,163 | 250,569 | 327,939 | 1,773,258 |
| Distinct Answers | 968 | 430 | 105,969 | N/A | 65,161 | 207,675 |
| % covered by top-1000 | 100 | 100 | 82.8 | N/A | 56.29 | 60.8 |
| Human Accuracy | 50.2 | N/A | 83.3 | N/A | 96.6 | N/A |
| Longest Question | 25 words | 24 words | 32 words | N/A | 24 words | 26 words |
| Longest Answer | 7 (list of 1 words) | 1 word | 17 words | N/A | 20 words | 24 words |
| Avg. Answer Length | 1.2 words | 1.0 words | 1.1 words | N/A | 2.0 words | 1.8 words |
| Evaluation Type | OE | OE | MC or OE | OE | MC or OE | OE |

Figure 2.3: VQA datasets comparison [13]

2.1.2 Evaluation metrics

VQA are evaluated with different metrics, based on the the dataset structure and peculiarities. Generally, metrics can be divided with respect to the VQA type:

- For **open-ended VQA** (see Section 2.1), many alternative methods have been adopted. Among them, the basic one is accuracy metric. Nevertheless, this method is often too rigid because the same concepts can be expressed in different ways. Consequently, many dataset adopt different methods. DAQUAR and COCO-QA employ the Wu-Palmer Similarity index [32].

Instead of comparing the given answer with the ground truth one, WUPS evaluates the similarity between word senses. In this way, this method overcomes the limitations related to accuracy metric. Nevertheless, WUPS is characterized by a series of imperfections. In some cases, it assigns high scores to even distant concepts. Another problem with WUPS is the inability of understanding the sense of entire sentences or phrasal answers, often exploited in the VQA Dataset and Visual7W. Therefore, this metric is often used in the cases the answer is composed of one word. An alternative is to collect multiple ground truth answers for the same question annotated by humans. This procedure has been adopted by the VQA dataset and DAQUAR-consensus. The final answer is obtained by evaluating the consensus among the human annotated ones. Therefore, achieving a high accuracy is really hard. Indeed, in these cases human agreement plays a key role, especially for questions starting with the pronoun *Why* (e.g., *Why the fork is on the table?*). Indeed, in these cases the answer is often subjective.

- For **multiple-choice VQA** (see Section 2.1), the simplest method is adopted and it is used by a part of the Visual7W, VQA Dataset and Visual Genome. In this case, the VQA system is required to choose which of the possible choices is the correct one.

2.2 Convolutional Neural Networks

VQA-system are generally composed of two main modules: an encoder (see feature extractors in Section 2.1) and a decoder. The first one is often a Convolution Neural Network (CNN), used for extracting from the image the most important features for the task, the second one is represented by a Recurrent Neural Network, responsible for generating the output sequence (i.e., the answer).

A CNN is a type of Neural Network that is able to take in input an image,

assigning to its objects a weight in order to understand their semantics and differentiate one from the others. These types of network provide a solution for the feature extraction task. Indeed, thanks to a sequence of operations they process the data present within an image, extracting its main characteristics. Figure 2.4 shows the general structure of a CNN, composed of a series of *convolutional* and *pooling* layers.

Convolutional layers:

Convolutional layers are the main blocks of a CNN. They take as input a tensor, condensing its information into a new one that is often characterized by a smaller shape. This operation is done performing a dot product between the image's tensor and a small matrix of values, called filter, that is "slided" over the entire image. Depending on the values contained within the filter, the convolution captures different types of information. Furthermore, while going deeper into the network, the features assume a more *abstract* meaning.

Pooling layers:

This operation aims at reducing the dimension of the output, mapping the input into a smaller space. It is exploited to diminish the computational power needed to process the image.

Dense layers:

These layers are composed of neurons, a mathematical operation that takes an input, multiplies it by a weight and then passes it through an activation function to the other neurons. In this case, each input neuron is connected to each output neuron.

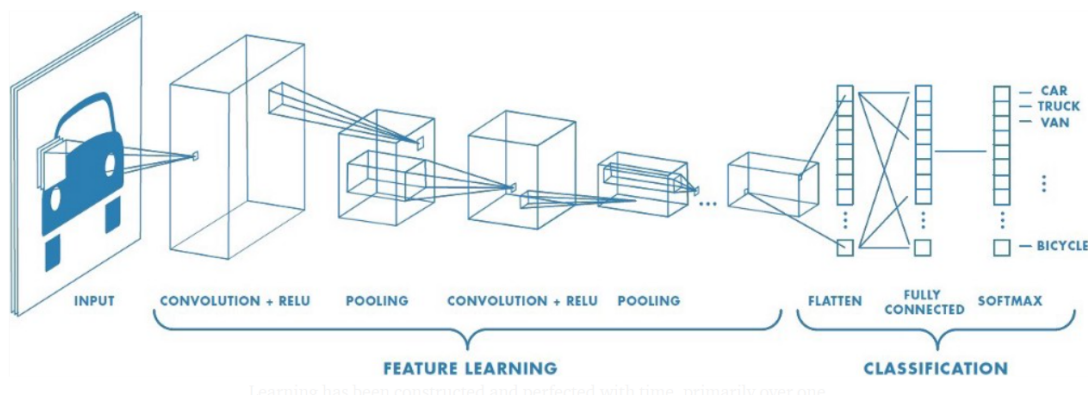


Figure 2.4: CNN architecture [6]

2.3 Recurrent Neural Networks

VQA task deals with sequences of words that compose both questions and answers. In order to understand their meaning, it is important for a VQA model to consider the order of the words inside a sentence. For these cases, where *temporality* of inputs plays a key role, a special type of network called Recurrent Neural Network (RNN) has been created. RNN recurs the output, that is handled together with the new input. Considering as input the previous outputs, each step maintains a relation with the previous and the following one. These architectures are trained using an input structure called Time Series, a sequence of chunks of information organized in steps. Time Series are often exploited for text sequencing tasks, but they are also employed in many other fields like statistics and forecastings. RNN is characterized by the presence of an internal state, called *HiddenState*. This part of the network is fundamental because it introduces the concept of *temporality*. Indeed, it can be considered as a memory unit that remembers what the network has processed during the previous steps of the series. This state is employed in order to provide a *context* for the computation of the input. To do so, together with the new input and the previous output, the state is exploited for the training

phase of the network. Figure 2.5 shows an example of RNN structure and its main components:

- **x**: constitutes the input time series. The latter is composed of pieces of encoded information (see x_i in Figure 2.5).
- **h**: constitutes the hidden state of the network. For each at time step i , this value is computed exploiting the state calculated in the previous time step (h_{i-1}) and the current input (x_i).
- **y**: constitutes the output vector. Each part of the vector y_i is obtained from the computation of the input and the hidden state related to the time step i .

The capability of the network to deal with *temporality* can be enhanced, increasing the number of hidden states. This technique provides a powerful network: the model recognizes easily the temporal relations among input words. The downside is an increasement of computational cost and a deeper architecture that affects the computation of the gradients. Indeed, a deeper system enhances the probability that during the training phase, the model's weights assume a value near to zero. Therefore, these procedure has often the opposite effect because the model is not able learn anymore. To overcome this issue, called *Vanishing gradient*, Long-Short Term Memory (LSTM) Networks were created.

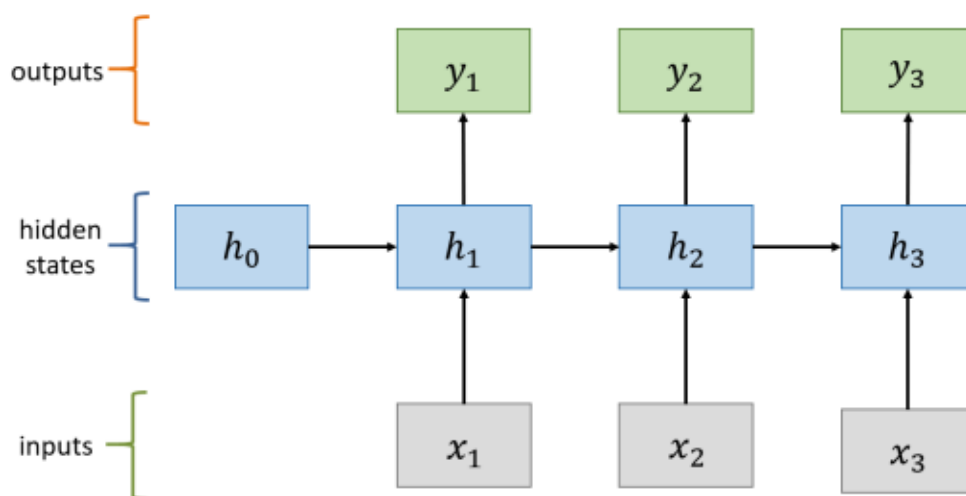


Figure 2.5: RNN unfolded architecture [26]

2.4 Long Short Term Memory

Traditional RNNs are employed for tasks related to the understanding of short temporal relationships. Indeed, they are unable to handle long ones because of how they exploit the information related to their hidden state. Furthermore, in Section 2.3 we explained why implementing a deeper network to face this problematic causes *vanishing gradient*. Long Short Term Memory were created to deal with tasks where long temporal relationships are required. This network shows a similar architecture with respect to RNNs one. On the other hand, the principal difference relies on the way the internal state is stored and updated. Indeed, the main problem of Recurrent neural networks is the absence of any filter for the chunks of information that update the internal state.

LSTMs aim at overcoming this problem, providing a mechanism that is able to keep or flush the hidden internal state. Figure 2.6 shows a high-view of this unit, called *cell*. As it depicts, a *cell* includes sigmoid (σ) and tangent (\tanh) functions. The first one has an output value between 0 and 1. In case the information should

be forgotten, the function outputs 0, otherwise the function outputs 1. The output of *tanh* activation function instead, sweeps between -1 and 1. A LSTM cell is composed of 3 sigmoid and 2 tanh activation functions. These functions are exploited to update the internal state and generate outputs. The activation functions can be subdivided into three main components called gates:

- **Forget Gate:** This gate decides if a piece of information will be removed from the cell state (see \mathbf{f}_t in Figure 2.6).
- **Input Gate:** This gate decides if the information contained in the input x_t can flow into the cell state h_t (see \mathbf{i}_t in Figure 2.6).
- **Output Gate:** This part is related to the computation of the output (see \mathbf{o}_t in Figure 2.6).

In the following section we describe in details these three components:

Forget Gate:

Figure 2.6 shows that this gate is the first one involved in the cell mechanism. It decides which pieces of information remove from the cell state C_{t-1} (i.e forget gate). The code below shows that given the current input x_t and the previous cell output h_{t-1} , a sigmoid activation function is exploited to compute an output value (i.e f_t) between 0 and 1:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \quad (2.1)$$

W_f and b_f represent the internal parameters of the cell.

Input Gate:

A new vector C_t of candidates ready to be inserted into the cell memory is computed exploiting the input x_t and the previous cell output h_{t-1} . As the code below shows (2.3), this procedure adopts a *tanh* activation. Furthermore, the *input gate*

removes some of the candidates employing another *sigmoid* function (2.2). This step is applied to the candidates C_t before updating the internal state.

The input gate i_t and the candidates C_t are computed as:

$$i_t = \sigma(W_i * [h_{t-1}; x_t] + b_i) \quad (2.2)$$

$$C_t = \tanh(W_C * [h_{t-1}; x_t] + b_C) \quad (2.3)$$

At this point, the forget gate removes the useless input data (f_t) selected in the previous step and the input gate selects the candidates that will be exploited for the computation of the new cell state:

$$C_t = \sigma(f_t * C_{t-1} + i_t * C_{t-1}) \quad (2.4)$$

Output Gate:

For every time step, this gate is employed for the computation of an output. The latter is based on a filtered version of the hidden state C_t . Indeed, also in this case a sigmoid activation function o_t will compute the same operation performed in the previous steps (2.1, 2.2). As (2.6) shows, this value will be multiplied with the state C_t , passed through the activation function \tanh . In this way, a new output h_t is computed.

$$o_t = \sigma(W_o * [h_{t-1}; x_t] + b_o) \quad (2.5)$$

$$h_t = o_t * \tanh(C_t) \quad (2.6)$$

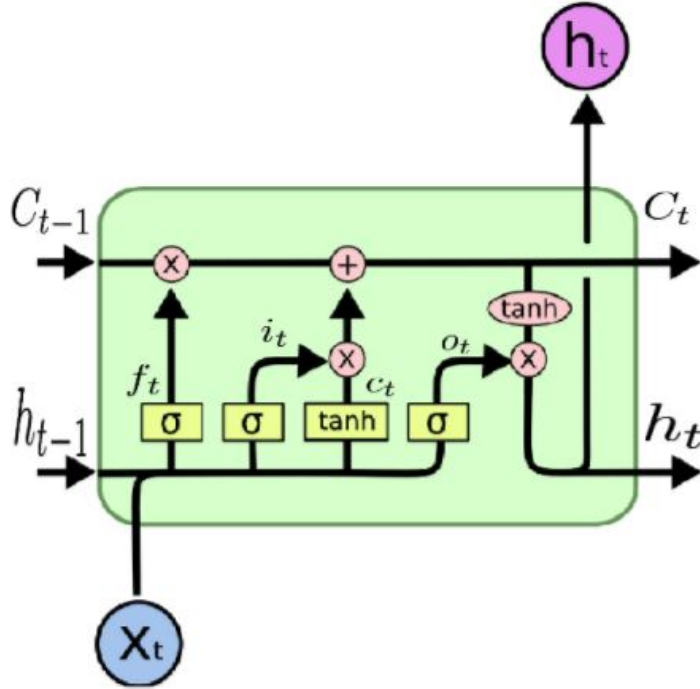


Figure 2.6: LSTM cell architecture [20]

Input structure and training

To train the network, input data is reshaped into 3-dimensional arrays. As Figure 2.7 shows, the dimensions are the number of data features, the number of time steps and the number of samples. This last dimension, called batch size, represents the number of samples that the model processes before its parameters are updated. In this case, batches are iteratively created sliding a window of a certain dimension on input time series. A LSTM remembers only what happened within a batch. Indeed, at the starting time point of every batch, states are initialized and set again to 0. To overcome this characteristic and make the model remember what happened in the previous batch, passing states from the previous batch to the next one, *Stateful LSTM* are used. Figure 2.7 shows time series of a *Stateful LSTM* having batches of dimension equal to 3 and series characterized by two features (x_1 , x_2).

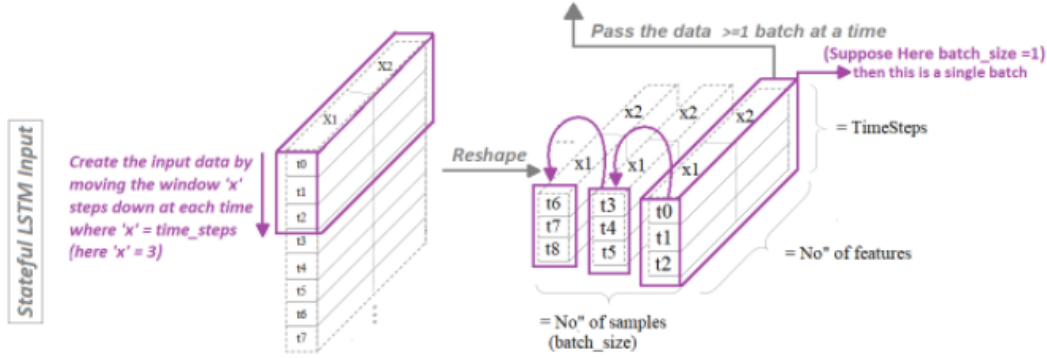


Figure 2.7: Stateful LSTM input time series [19]

The training phase requires also the choice of a gradient descent algorithm. Thanks to this process, network parameters are modified in order to minimize the error function of the model over the training data. There are many variants of this gradient descent algorithm characterized by specific peculiarities. The biggest difference among them is the momentum, a technique that adjust weight's update. Thanks to it, weight's adjustment is biased by previous update and the best weight setting is reached faster, preventing situations where the algorithm gets stuck into a local minimum. Some of them are Adagrad [9], AdaDelta [34], Adam [14] and Stochastic Gradient descent.

2.5 Scene graphs

One of the main problems of Visual Question Answering is to reason about semantic and spatial indications related to a question. To fully understand the visual semantics of images, we propose to integrate the data with which we train our network with a structured visual representation of pictures: scene graphs. A scene graph is a directed acyclic graph, that describes the relations among images objects, that are any type of entities depicted in the image (e.g., *car*, *person*, *guitar* in Figure 2.8). The nodes in the graph may represent objects, attributes and relationships. Objects are linked to their respective attribute nodes with a graph edge (e.g., *car*→*black* in Figure 2.8). The relationship's nodes link one object to another,

for example the Figure 2.8 shows the relation $Person \xrightarrow{driving} car$. Visual Genome dataset provides the biggest and detailed scene graphs dataset. Scene graphs have been shown to improve image retrieval, generation, video understanding [21] and visual reasoning [28]. Nevertheless, VQA systems that use SG usually integrate multiple techniques. Indeed, in a study of the Visual Genome dataset [25], only 40% of the questions could be answered with the usage of human annotated scene graphs.

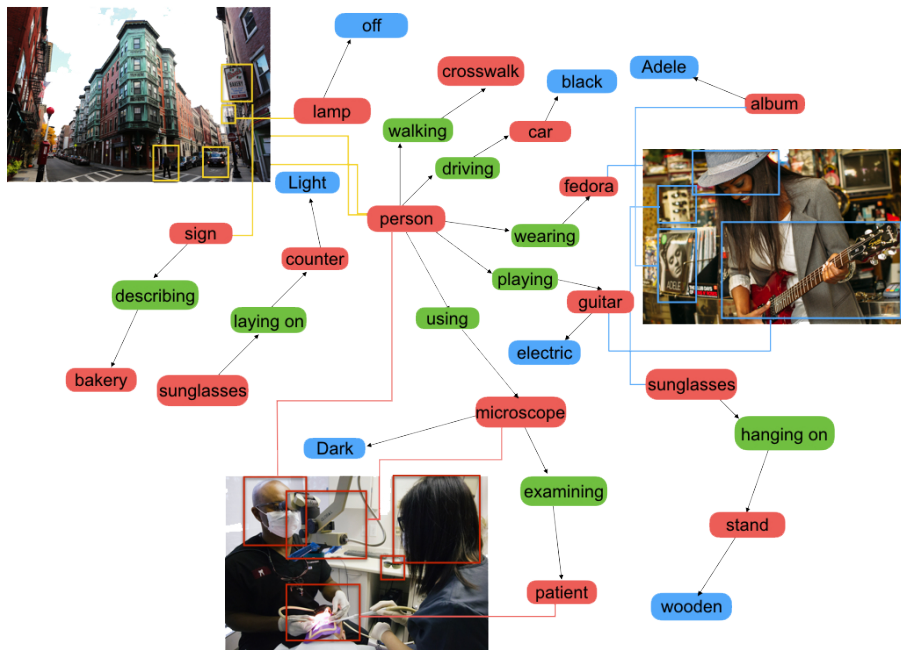


Figure 2.8: Scene graph [30]

2.6 External commonsense knowledge base

In this research area, we refer to ‘commonsense knowledge’ as the facts and notions possessed by most people (e.g., car is a vehicle in Figure 2.9). In other words, it is the most general kind of knowledge about the everyday world. ConceptNet [18] is a freely available large-scale commonsense knowledge base. It can be seen as a directed graph where nodes are represented by words and phrases in natural language, connected by weighted relations. ConceptNet exploits a set of 42 relations

(e.g., IsA, UsedFor, CapableOf, LocationOf in Figure 2.9). These relationships are chosen because they are unrelated to the language or the source of the terms they connect. In order to enhance the ability of reasoning beyond the image contents, many recent frameworks [2, 17] tried to exploit this knowledge. In other cases, it has been used as a way to extract reasoning paths among concepts.

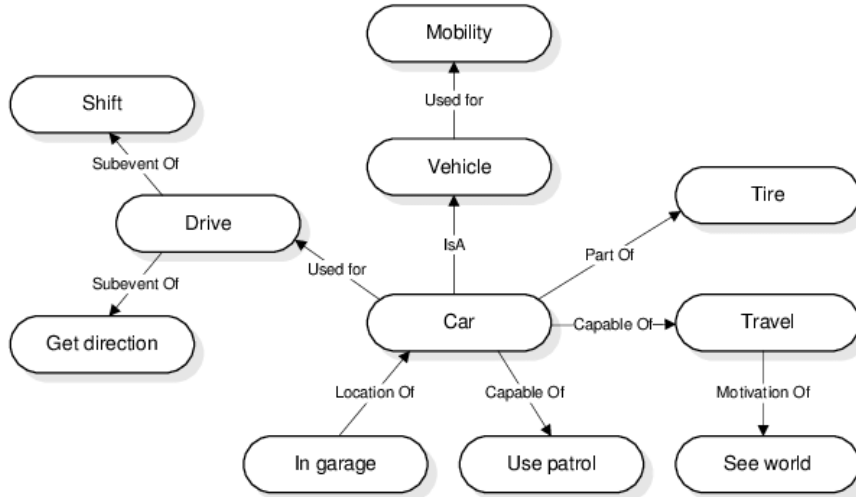


Figure 2.9: A frame of ConceptNet, related to ‘Car’ [18]

2.7 Natural Language Processing

Natural Language Processing is an area of research that involves techniques of linguistics and artificial intelligence. This field aims at studying the interaction between humans and computers. It focuses on how to adapt input data written into natural language into a computationally digestible format. A VQA-system has to both understand the sense of the question and generating a meaningful answer. For this reason, the model should be able to process the inputs and generate outputs. Of course, data have to be adapted to a format the VQA-system can elaborate. Therefore, an initial procedure transforms the inputs, expressed in natural language, into something that computer can understand: a vector of numbers. This process is performed not only by VQA models but almost by every NLP system.

To do so, a series of NLP procedures is adopted.

2.7.1 NLP pipeline

In the following section, we describe the preprocessing pipeline: a set of operations applied to translate a sentence in natural language to a vector of numbers.

Tokenization

The first step of the pipeline consists in the *Tokenization* of the input. The sentences are split into pieces, called tokens (e.g., the questions “Why the fork is on the table?” is tokenized as “Why”, “fork”, “on”, “table”). Thanks to this operation, every token can be analyzed and elaborated separately. For the VQA task, this phase is employed for both questions and answers.

Embedding

The second step of the pipeline consists in embedding the tokens. Word Embedding is an encoding technique, consisting on mapping each token to a vector of a predefined size of dimension n . In this way, tokens are projected into a multidimensional space where every vector belongs to R_n . The extracted tokens are converted into something that a model can understand: a vector of numbers. As the words are turned into vectors, the model is able to reason about tokens meaning, measuring their semantical distance. Word embeddings are usually learned using Neural Networks. In our work for example, fastText projects into a 300-dimensional semantic space. FastText is an artificial neural network that has been trained on a large number of texts. This operation has been performed in order to extract, with respect to its context, a word embedding for each word in the language vocabulary. FastText has the capacity of inferring semantic relationships between words. This method is employed not only to simplify computations, but also to exploit the multidimensional space R_n , where mathematical operations such as addition and multiplication hold. In this way, semantic vectors for sentences can be obtained.

Chapter 3

Related works

The most common VQA architectures exploits both textual and visual feature extractors and feed their output to an answer generator (see Section 2.1). Most of the times, these components are implemented with an encoder, a decoder (see Section 2.2) and a classifier. In the following, we provide a review of previous works in visual question answering, highlighting the differences with our work. We start by reviewing VQA systems based on attributes to describe the image, then we move to systems that exploit external knowledge bases to integrate the training process. Finally, we review recent solutions for identifying the important information of query images, called graph attention networks. At the end of this chapter, in Section 3.4 we highlight the contribution of our work.

3.1 Attribute extraction and LSTM

The early VQA-system, stressed the importance of understanding which are the relevant parts of the picture. In other words, “where to look” plays another key role for answering correctly. To face this task, the first approaches consisted in training ad-hoc networks [33]. Other methods, exploited the extracted information to gather further data from external knowledge. In particular, the architecture developed by Qi Wu et al. [31] paved the way to new original solutions. Their

VQA-system, depicted in Figure 3.1, exploits an ‘internal representation’ of the picture to extract attributes from the image. This attribute-based representation of a picture is exploited for mining from a knowledge base, DBpedia, further captions related to the context. This ensemble of information is embedded and used as input for a LSTM. Inspired by this approach, for every image we exploited an ensemble of key concepts to train our network.

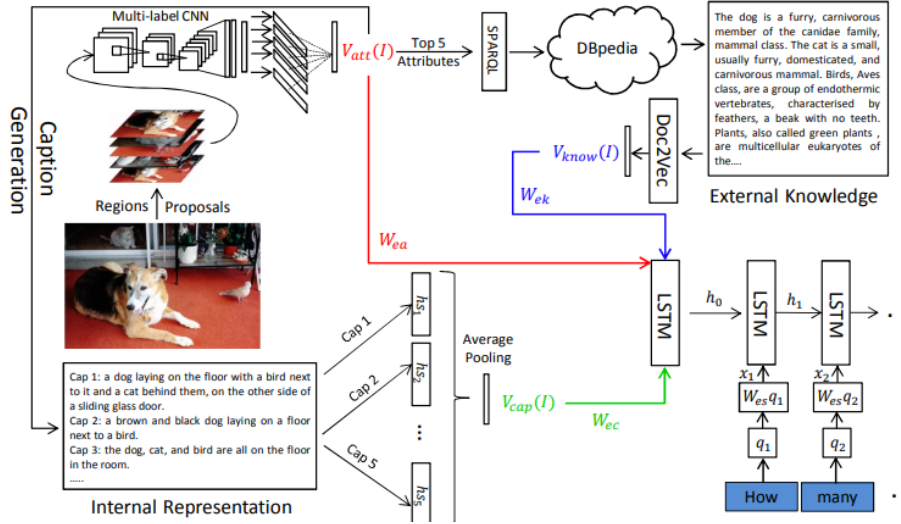


Figure 3.1: Figure shows the architecture designed by Qi Wu et al. [31].

3.2 Usage of external Knowledge bases

The architecture presented in Section 3.1, showed how external knowledge bases (KB) are used for the VQA task. In other cases, KBs are used to improve model’s capability of commonsense reasoning. ConceptNet is often adopted for this purpose and it has two fundamental usages. Many recent architectures [35, 16] use this

ontology to extract concepts in order to integrate the ones related to questions and answers. In other cases, this knowledge base is exploited as a way of connecting concepts. As *zhong* affirms in its work [35], it is possible that the lack of common-sense reasoning ability is caused by the absence of connections between concepts. Therefore, recent systems [2, 17] use this external KB as a way of finding reasoning paths that connect Questions and Answer tokens. In general, path extraction has been shown to be very effective for this type of task. Figure 3.2 for example, depicts the approach adopted in [2]. In this case, ConceptNet is employed for finding a reasoning path among the question and answer concepts that is consistent with a given context. In this work, we used this knowledge base in order to verify the existence of close relations between concepts. Leveraging on ConceptNet, we assigned weights to extract additional attributes and extend the attribute-ensemble, picking the ones that were more often present among the correlated terms.

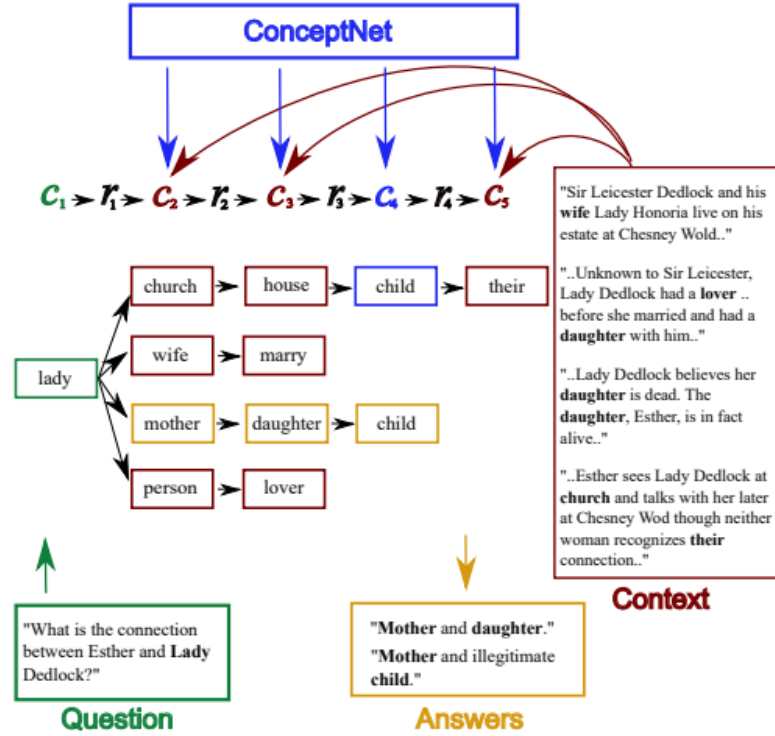


Figure 3.2: An example of ConceptNet usage [2]. In this case, this KB is employed for finding a path among the question and answer concepts

3.3 Graph attention networks

To solve the “where to look” problem, recent systems tend to adopt attention mechanisms. These approaches exploit mathematical operations with the aim of evaluating how much an input element relates to the others, generating an output that consider these interactions. Regarding graphs, a similar approach is used with Graph convolutional networks, a way of creating an embedding of the graph. For instance, Figure 3.3 depicts the technique adopted by Graph attention network (GAT) [29], a type of graph convolutional network, for the graph node h_1 . GAT computes a new representation of the node, defined as h'_1 , that leverages on node neighbors. Indeed, the Figure 3.3 shows that the computation of h'_1 involves a series

of coefficients, called Attention Coefficients, related to h_i neighbors. These terms determine the importance of a node with respect to another one. These techniques have been exploited in the most recent VQA architectures. For example, Marcel et al. [12] attempted to use Graph attention networks on scene graphs for its VQA model. With respect to this system, we tried to exploit a simpler Graph embedding technique. In our work, we embedded every node considering its neighbors and the relative weights that we assigned.

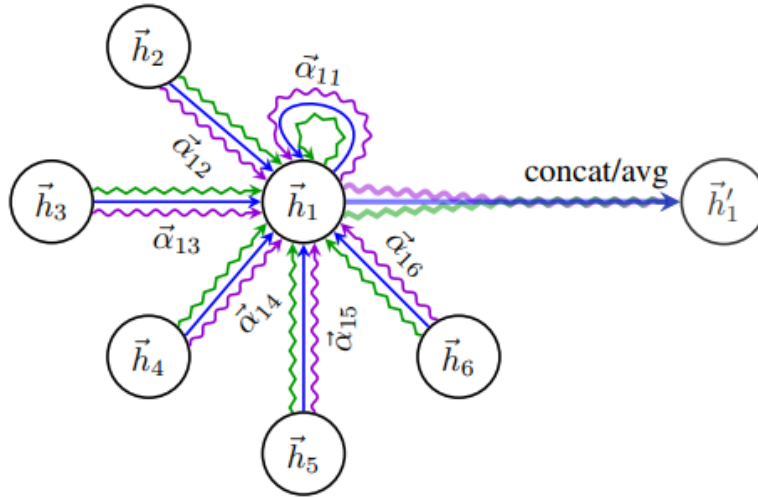


Figure 3.3: GAT convolutional mechanism [29]

3.4 Contribution

Recent frameworks focused their efforts toward the creation of complex architectures. We propose a straightforward approach (i.e., semantics-aware VQA) that aims at incorporating external knowledge to the image, including common sense

knowledge and scene graphs, into the VQA process. With respect to existing methods, we aim at solving the “where to look” (see Section 3.1) problem without attention modules. Indeed, we exploit ConceptNet ontology [18] to evidence the most pertinent image objects with the question.

Our contribution can be summarized as follows:

- We exploit a structure that consider as input: Questions/Answer concepts, an ensemble of attributes related to the picture and a scene graph.
- We propose to exploit ConceptNet to weight the importance of the concepts extracted from input pictures and the relations belonging to scene graphs.
- We propose a novel way of mining, from an external knowledge base, the most pertinent concepts with the image context.

In the following chapter, we will describe in detail the architecture of our model.

Chapter 4

Semantics-aware VQA

This work investigates a novel approach to the Visual Question Answering task. In particular, our goal is to improve the way a VQA model makes commonsense reasoning. Our model is expected to answer multiple-choice questions related to images, employing an architecture capable of exploiting many levels of knowledge. Indeed, our VQA system utilizes both visual features and scene graphs: a graph-structured external knowledge related to the image. A scene graph provides an alternative representation of an image, capable of storing picture’s semantics (see Section 2.5). In the following chapter, we will describe our architecture.

Figure 4.1 depicts a high level view of our model. The structure is composed of several modules, that will be described in detail in these sections. The input data used for training the model are the *input images*, the *multiple choice questions* and the *scene graphs*. To mine data from these sources, we build 3 modules: (I) *Answer selector*, (II) *Attribute extraction* and (III) *Scene graph embedding*. The *Answer selector* module takes as input a multiple choice question, its purpose is selecting one of the 4 answers and output a question/answer pair.

The *Attribute extraction* receives as input: the image, a question/answer pair and external data mined from the knowledge base ConceptNet. This module extracts from a picture a set of concepts related to its context that we ablate as *Attributes*.

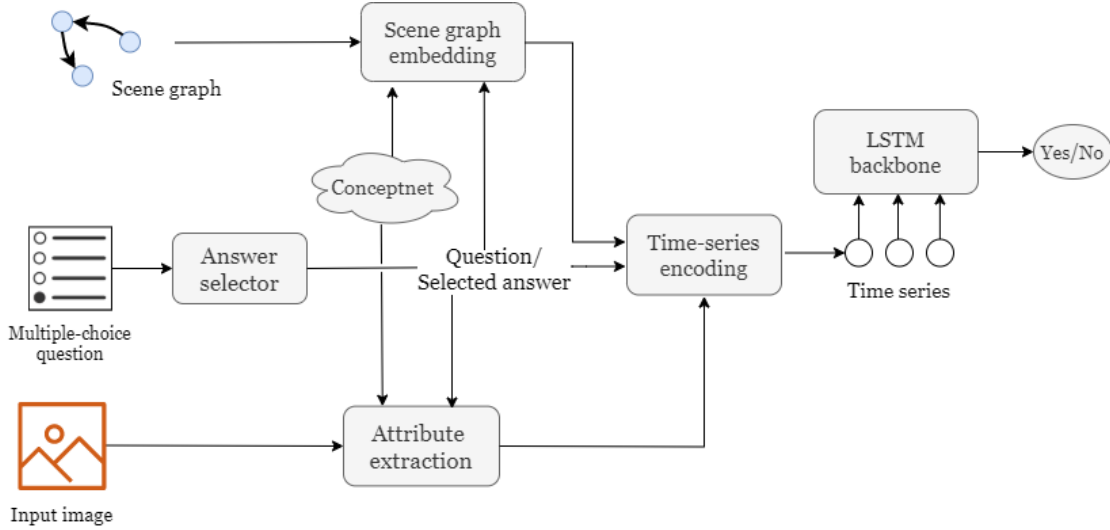


Figure 4.1: Network architecture

The selected answer and ConceptNet are exploited to mine further attributes and to enrich every attribute with a value, called weight, that indicates the attribute’s relevance in that context.

The *Scene graph embedding* module takes as input the Scene graph related to the input image, the information from the knowledge base ConceptNet and a question/answer pair. This part of the model computes a scene graph embedding. The latter is formed by a set of data contained in the scene graph. Also in this case, scene graph elements are enriched with a value called weight that indicates their relevance in that context.

The attributes, the scene graph embedding, the question/answer pair constitute the input data of the model’s *backbone*. The latter is represented by a *LSTM* network (see Section 2.4). This particular type of architecture takes into account the notion of input temporality, fundamental for the VQA task to understand the sense of a question and formulate an answer. The model’s output is a binary label assigned to the answer selected by *Answer selector* module to indicate if that option is the correct one. As we explained in Section 2.4, this type of network is trained exploiting an input structure called *Time series*, a sequence of pieces of

information organized in steps. In our model, *Time series encoding* module has the aim of computing the input time series. It processes the input data ensemble, reshaping and adapting these information into a computationally digestible format.

4.1 Preliminary definitions

Multiple choice questions

They constitute the samples extracted from visual7W dataset. Every question is expressed in natural language and it belongs to the type *why*. Furthermore, it is characterized by 4 possible answers. Both questions and answers, written into natural language, contain key concepts that we will extract and exploit in the architecture. These concepts have to be converted into a computationally digestible format. This operation is performed leveraging on FastText embedder (see Section 2.7) which projects a concept into a 300-dimensional semantic space.

Let Q be the question, we ablate $[q_1, \dots, q_i, \dots, q_n]$ the concepts extracted from Q . We define $[\hat{q}_1, \dots, \hat{q}_i, \dots, \hat{q}_n]$ the embedding of these concepts obtained leveraging on FastText. Let $[AN_1, \dots, AN_4]$ be the 4 answer options related to Q . We define $[an_1, \dots, an_i, \dots, an_n]$ the concepts extracted from AN_i . We define $[\hat{an}_1, \dots, \hat{an}_i, \dots, \hat{an}_n]$ the embedding of these concepts obtained employing FastText.

Scene graphs

A scene graph is a directed acyclic graph, that describes the relations among image objects, that are any type of entities depicted in the image being represented (see Section 2.5). We decided to integrate this knowledge to the data related to the Question and Answer couples to improve the model’s reasoning. Indeed, relations could improve the inference of commonsense knowledge base related to a

context. For example, Figure 4.2 shows that the depicted scene graph is sufficient for inferring a part of the image’s context.

Scene graph nodes are divided into: vertices, relationships and attributes. Furthermore, we also define relations with the triplets $\langle \text{vertex}, \text{relationship}, \text{vertex} \rangle$ inside the graph. A vertex is a node of the graph that represents any entity or object inside an image (e.g., child, shirt in Figure 4.2). An attribute is a node that represents an object feature (e.g., blue in Figure 4.2). A relationship instead, is a node that links two vertices corresponding to two objects. A relationship is characterized by a label that describes the type of bound that connects the two vertices. For instance, Figure 4.2 depicts the relation triplet $\langle \text{child}, \text{wear}, \text{shirt} \rangle$ where the relationship label *wear* is an action. The Figure shows also that a relationship can specify the mutual position of two entities (e.g., $\langle \text{child}, \text{on}, \text{snow} \rangle$).

Let $G = (V, E)$ be a scene graph where V is the set of nodes of any of the three types (i.e., vertex, attribute, relationship). We refer to relations r_i as the triplets $\langle v_i, r_i, v_j \rangle$ where $v_i \in V$ and $v_j \in V$ are two vertices (i.e., objects) and l is the label of a relationship node connecting v_i and v_j with two edges in E . Also in this case, we define \hat{v} and \hat{r}_i the embedding of these terms obtained exploiting FastText.

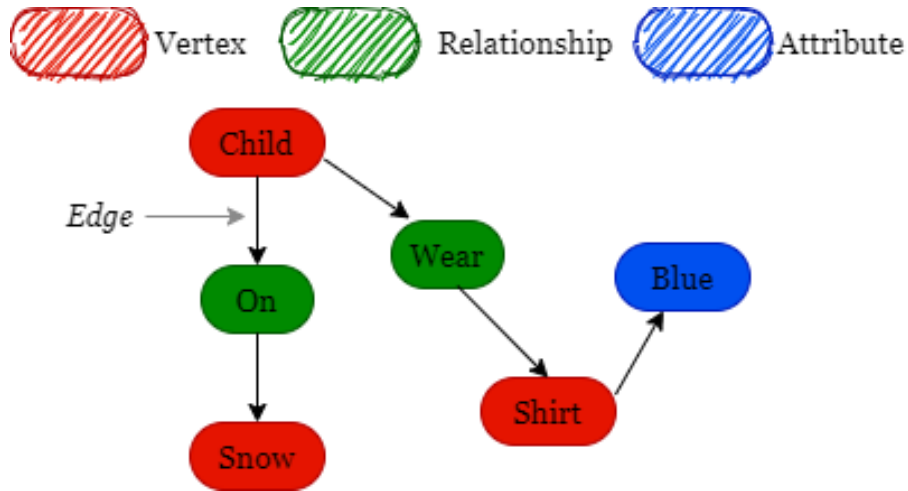


Figure 4.2: A scene graph example

ConceptNet

ConceptNet is large-scale commonsense knowledge base. It can be seen as a directed graph constituted by two main elements: nodes and edges. The nodes are concepts, expressed in natural language. They are connected one to each other with labeled edges. The labels specify the relationship that exists among two concepts.

Let $CN = (C, E)$ be a ConceptNet graph. We define c_i as a node in ConceptNet graph and e' as an edge's label. We ablate $N(c_i)$ the set of concepts having a direct link with c_i (i.e., it's neighbors). For instance, Figure 4.3 shows that given $c_i = Car$, we define $N(c_i)=[Drive, Vehicle, Travel]$.

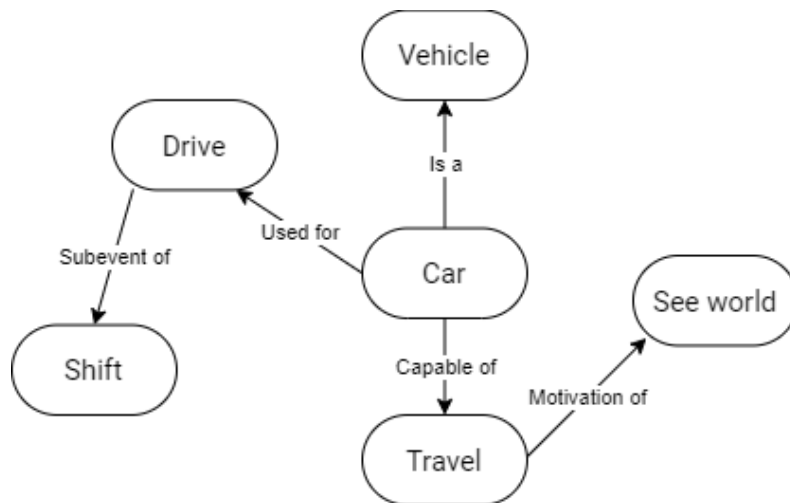


Figure 4.3: A frame of ConceptNet, related to ‘Car’

4.2 Scene graph embedding

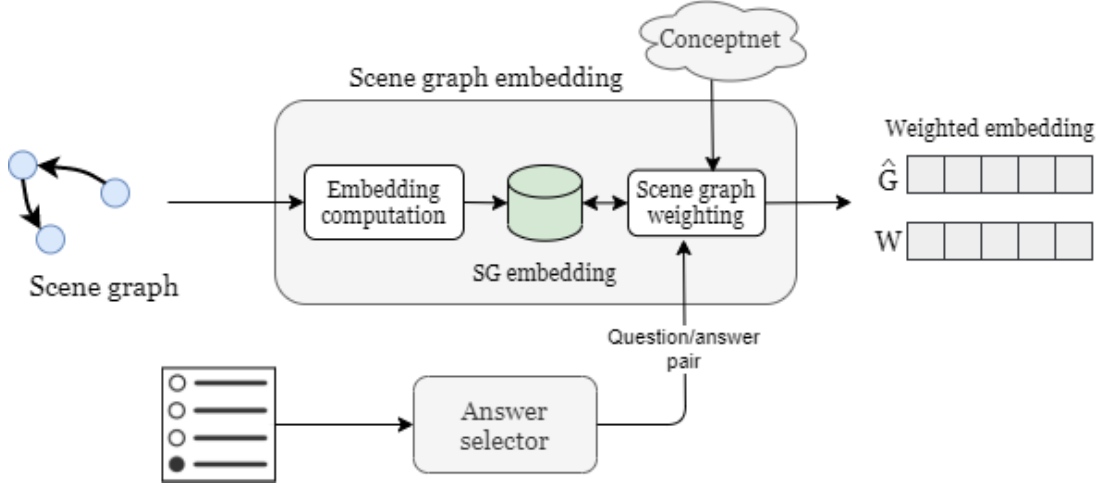


Figure 4.4: Scene graph embedding module

In our work, we created a part of the model with the aim of exploiting the scene graphs related to images. The module is composed of 2 blocks: (I) Embedding computation and (II) Scene graph weighting. The first is exploited for generating a scene graph embedding. The latter is used for assigning a weight to every element of the embedding. Weights indicate the importance of the elements with respect to the question/answer pair that the block receives as input. In the following paragraphs, we describe in depth the function of each block.

4.2.1 Embedding computation

In order to exploit scene graphs, we embedded them adopting two main approaches called (I) Element embedding, (II) convolutional embedding (CE). The first aims at extracting and embedding the relations inside the scene graph. The scene graph embedding will be composed of a set of embedded relations. The latter focuses on embedding every vertex individually. In this case, the scene graph

embedding will be composed of a set of embedded vertex.

Element embedding

From Visual Genome’s scene graphs, we created a set containing the 35 most common spatial relationship (e.g., ‘on’, ‘in front of’, ‘behind’, ‘in’, ‘next to’). Let SR be the set of 35 relationship labels extracted from Visual Genome. For every graph, we considered only these group of relationships. Let R be the set of relations of a scene graph $G = (V, E)$ and \hat{G} be the embedding of G . We define the subset $R_s \subseteq R$, where each relation $r = \langle v_i, r_1, v_j \rangle \in R_s$ satisfies $r_1 \in SR$. For every $r \in R_s$ we inspected two ways of embedding its content: (I) Element averaging embedding (EAE) and (II) Element concatenation embedding (ECE).

Given a relation, the first approach computes the embedding of the elements that constitute the triplet $\langle vertex, label, vertex \rangle$ using FastText. To create an embedding of the relation, EAE performs the average embedding of these triplets. The scene graph embedding will be a set composed of the embedding of spatial relations. In the following, we formalize this process. For every triplet $\langle v_i, r_1, v_j \rangle \in R_s$ EAE computes the embeddings $\hat{v}_i, \hat{r}_1, \hat{v}_j$. Let \hat{r} be the embedding of $r \in R_s$. We define:

$$\hat{r} = \frac{\hat{v}_i + \hat{r}_1 + \hat{v}_j}{3} \quad (4.1)$$

We finally define $\hat{G} = [\hat{r}_1, \dots, \hat{r}_i, \dots, \hat{r}_n]$ the scene graph embedding, which is the time series including the embedding of spatial relations. This approach is the simplest one but it does not preserve relation’s semantic. Indeed, switching the two vertices that constitute a relation, EAE computes the same result. For instance, Child→On→Snow provides the same embedding of Snow→On→Child.

Given a relation, ECE retrieves the embedding of the elements that constitute the triplet $\langle vertex, label, vertex \rangle$ using FastText. This approach separates vertices from the relationship label. It computes the average of the vertices embeddings, concatenating the obtained vector with the embedding of the relationship

label. Let \hat{v}_1 and \hat{v}_2 be two vertices embedding. We denote \parallel the concatenation operator of two embeddings. Let \hat{r} be the embedding of r . For every triplet $v_i, e, v_j \in R_s$ ECE computes $\hat{v}_i, \hat{e}, \hat{v}_j$. We define:

$$\hat{r} = \frac{\hat{v}_i + \hat{v}_j}{2} \parallel \hat{e} \quad (4.2)$$

We finally define $\hat{G}=[\hat{r}_1, \dots, \hat{r}_1, \dots, \hat{r}_n]$. ECE does not preserve graph semantic. On the other hand, it avoids performing a mean among vertices and relationships.

Convolutional embedding

Differently from element embedding, CE exploits the entire scene graph. This approach assigns a weight to every vertex employing the module *Scene graph weighting*, whose functionality is explained in Section 4.2.2. For every vertex CE extracts its neighbors, that are the vertices having a common relation with it (e.g., Child’s neighbors are Snowsuit, Helmet, Pole and Snow in Figure 4.5). Neighbors are embedded using FastText (see Section 2.7). The final vertex embedding is obtained averaging the neighbors that are multiplied for their corresponding weights. Finally, the scene graph embedding is constituted by a set of vertices s . In the following, we formalize this embedding process. Let $v_x \in G$ be a vertex node. Considering the subset $R_s \in R$, where each $r \in R_s$ has the form $\langle v_x, r_1, v_i \rangle$. We denote $N'(v_x)$ the vertices n_i belonging to each $r \in R_s$. Let $W = [w_1, \dots, w_i, \dots, w_n]$ be the set of weights assigned to $N'(v_x)$ elements. Let \hat{v}_x be the vertex embedding, defined as:

$$\hat{v}_x = \sum_{i=1}^{|N'(v_x)|} \frac{w_i \hat{n}_i}{\sum_i w_i} \quad (4.3)$$

Where $\hat{n}_i \in N'(v_x)$ is a neighbour of v_x embedded with FastText and $w''_i \in W$ is the weight assigned to the neighbour \hat{n}_i . The scene graph embedding \hat{G} is defined as $\hat{G}=[\hat{v}_1, \dots, \hat{v}_i, \dots, \hat{v}_n]$, which is the time series including the embedding of the vertices.

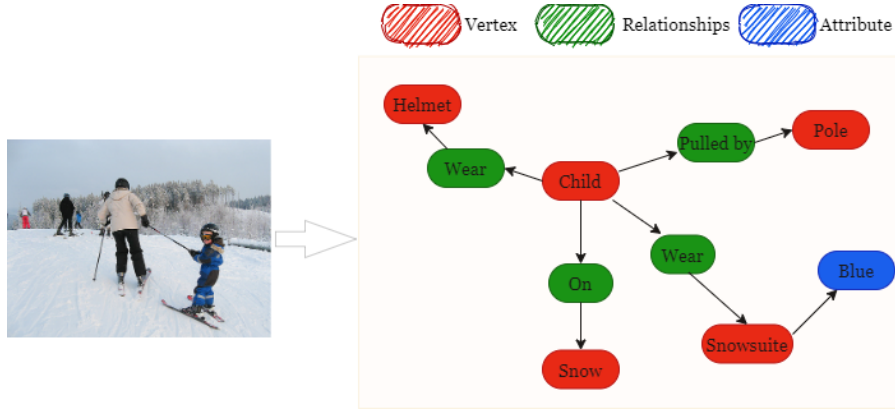


Figure 4.5: Scene graph lookup

4.2.2 Scene graph weighting

To enhance the importance of the concepts directly related with the Question/Answer couple, we created a set of weights. Leveraging on ConceptNet, this module assigns a weight to every relation or vertex with the aim of focusing the attention of the model on the relevant parts of the scene graph. This module receives as input the scene graph embedding, ConceptNet and the selected answer. ConceptNet connects words with respect to their semantic meaning (see Section 2.6).

We verified the existence of a direct link between every relation’s vertex in the scene graph and ones of the concepts related to both question and answer, assigning a higher weight in case this link exists. To increase the differences among the answers we assigned to their concepts a higher weight with respect to question’s one.

The weights can assume 3 different values. The smaller one is assigned in case a concept is related neither with question or answer ones. The second is used when the attribute is related to a question. The third one is exploited when a relation

with the answer’s concepts exists. We considered the latter the most important case, therefore we assigned to it the higher value. In the following, we formalize the weighting process.

Let $G=(V,E)$ be a scene graph. We refer to each relation r_i as the triplet $\langle v_i, r_i, v_j \rangle$. Let $W = [w_1, \dots, w_i, \dots, w_n]$ be the set of weights assigned to the graph relations (e.g., w_i assigned to relation r_i). Let $CN = (C, E)$ be ConceptNet graph. We define c as a concept in ConceptNet graph and $N(c)$ the set of concepts with 1 hop from c . Let Q be a question and AN one of its answers. Consider now the relationship r_i . If c is an Answer concept and v_i or v_j (in the triplets of r_i) belong to $N(c)$ set, we assign to w_i a value of 1. If c is a question concept and v_i or v_j belong to $N(c)$ set, we assign to w_i a value of 0.50. If v_i or v_j do not belong to $N(c)$ sets of any question/answer concepts, we assign to w_i a value of 0.25. This reasoning can be formalized with:

$$w_i = \begin{cases} 1 & \text{if } c \in AN \wedge (v_i \in N(c) \vee v_j \in N(c)) \\ 0.5 & \text{if } c \in Q \wedge (v_i \in N(c) \vee v_j \in N(c)) \\ 0.25 & \text{if } (c \in Q \vee c \in AN) \wedge (v_i \notin N(c) \vee v_j \notin N(c)) \end{cases}$$

After this step, the output of the scene graph embedding module is composed of \hat{G} (i.e., graph embedding) and \hat{W} (i.e., a weight for each relationship).

4.3 Attribute extraction

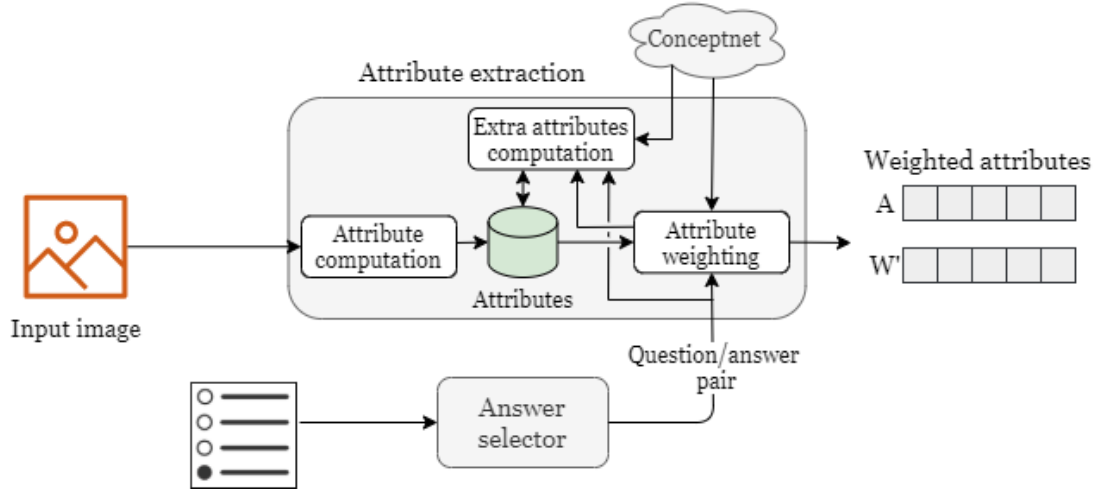


Figure 4.6: Attribute extraction module

We created this part of our model with the aim of extracting generic concepts related to an image. These data play a key role in our work, because they enhance the ability of the network to reason about the image’s context. Therefore, we exploited this knowledge in every attempt that we performed. This module is composed of 3 building blocks: (I) Attribute computation, (II) Attribute weighting and (III) Extra attributes computation. The Attribute computation block is exploited for extracting concepts from the image, called attributes. The Attribute weighting block is employed for assigning a weight to the extracted attributes. Weights indicate the importance of the elements with respect to the question/answer pair that the block receives as input. The Extra attributes computation block is exploited for mining further concepts from ConceptNet. In the following paragraphs, we describe in detail the function of these blocks.

4.3.1 Attribute computation

We employed an artificial neural network model called Clarifai [5], to obtain concepts present in each image. The model receives as input a picture and it has the ability of recognizing not only the objects (e.g., ‘boat’ in Figure 4.7), but also emotions (e.g., ‘composure’ in Figure 4.7), actions and abstract concepts. The network outputs a score for each of these terms which shows the likelihood that the concept is represented in the image. Among the outputs, we decided to exploit the ones characterized by a probability higher than 80%. We chose this threshold because the concepts characterized by a lower probability value are often too generic, not providing any useful knowledge.

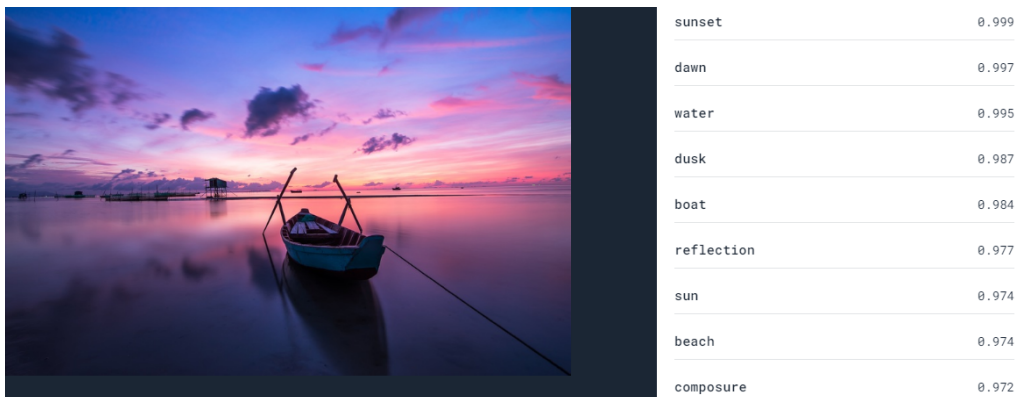


Figure 4.7: An example of attribute ensemble

4.3.2 Attribute weighting

This module aims at assigning a weight to the extracted attributes. These values stress the importance of the attributes inside the image context. The module

receives as input the attributes, ConceptNet and the selected answer. To assign the weights, we exploited an external ontological knowledge base, ConceptNet, where terms are connected with respect to their semantic meaning (see Section 2.6).

The adopted procedure is the same exploited in the module *Scene graph Weighting* (see Section 4.2.2). The weights can assume 3 different values. The smaller one is assigned in case a concept is not related neither with question or answer ones (e.g., the attribute “Powder”, in Figure 4.8). The second is used when the attribute is related with the question (e.g., the attribute “child” with the question concept “boy”, $\text{child} \xrightarrow{\text{synonym}} \text{boy}$, in Figure 4.8). The third one is exploited when a relation with the answer concepts (e.g., the attribute “skiing” with the answer concept “snow”, $\text{skiing} \xrightarrow{\text{requires}} \text{snow}$ in Figure 4.8). Also in this case, we considered the latter the most important case, therefore we assigned to it the higher value. In the following paragraph, we formalize the weighting procedure.

Let $A = [a_1, \dots, a_i, \dots, a_n]$ be the set of attributes extracted with Clarifai from an image and $W' = [w'_1, \dots, w'_i, \dots, w'_n]$ be the weight set related to this ensemble of concepts. Let $CN = (C, E)$ be ConceptNet graph. We define c as a concept in ConceptNet graph and $N(c)$ the set of concepts with 1 hop from c . Let Q be a question and AN an answer. If c is an answer concept and a_i belongs to $N(c)$, we assign to w'_i a value of 1. If c is a question’s concept and a_i belongs to $N(c)$, we assign to w'_i a value of 0.50. If a_i does not belong to $N(c)$ and it is not included in either question or answer concepts, we assign to w'_i a value of 0.25. This reasoning can be formalized with:

$$w'_i = \begin{cases} 1 & \text{if } c \in AN \wedge (a_i \in N(c)) \\ 0.5 & \text{if } c \in Q \wedge (a_i \in N(c)) \\ 0.25 & \text{if } (c \in Q \vee c \in AN) \wedge (a_i \notin N(c)) \end{cases}$$

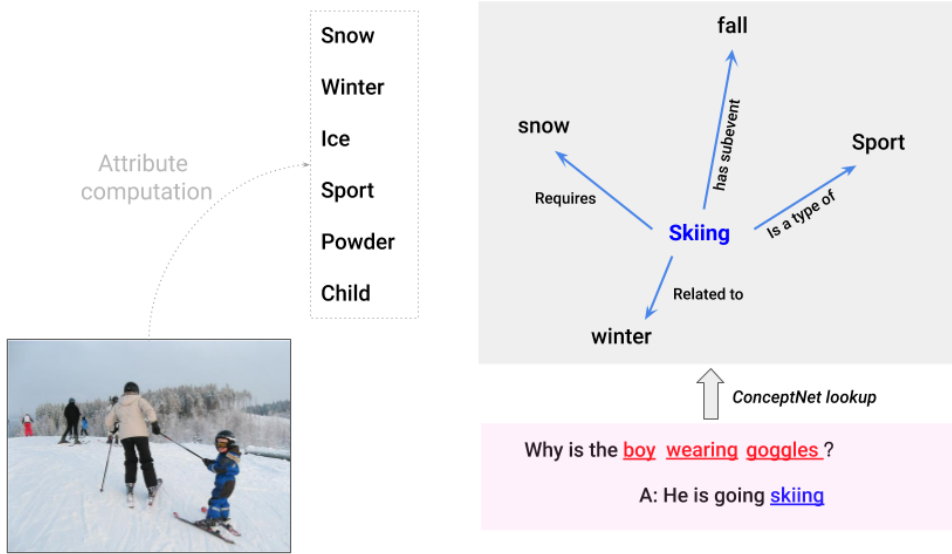


Figure 4.8: An example of attribute weighting procedure for an answer concept (i.e., skiing)

4.3.3 Extra attributes computation

This module aims at enriching the attribute set with further concepts extracted from ConceptNet. It receives as input the attributes, the weights, the selected answer and ConceptNet. The module extracts from ConceptNet the concepts correlated with the image attributes that are directly related with the answer concepts (e.g., Skiing $\xrightarrow{\text{requires}}$ snow in Figure 4.9). This module creates a set for every selected answer. Each set stores the concepts extracted from the external knowledge base by following a path starting from each attribute. The number of times every concept is reached by a path is also stored. The procedure enriches the attribute ensemble with the 10 concepts of the set that appeared to be more frequently related to the attributes. For instance, Figure 4.9 shows that the attributes Snow, Winter and Sport are directly related to the answer. Furthermore, it depicts that

the concept ‘Season’ is directly linked with these attributes. Hence, it will belong to the set created for that answer, paired with the number of paths from which it can be reached (i.e., 3 in this example).

Let Q be a question and AN be an answer. Let $A = [a_1, \dots, a_i, \dots, a_n]$ be the set of attributes extracted from an image and $W' = [w'_1, \dots, w'_i, \dots, w'_n]$ the attribute weights related to the answer. We define $A_s = [a_1, \dots, a_i, \dots, a_n] \subseteq A$ the attribute subset characterized by $w'_i = 1$. Let M_i be the set created for an answer. M_i is obtained by merging the neighbors of every $c_i \in A_s$. More formally, we define $M_i = \bigcup_{j=1}^{\|A_s\|} N(c_i)$ where $c_i \in A_s$ and $N(c_i)$ are its neighbors. Moreover, for each concept $m_{i,j} \in M_i$ we store in count $m_{i,j}$ the number of times it occurs in $N(c_i) \forall c_i$. For every Question/Answer couple, we refer to *extraattributes* as the 10 more frequent concepts in M_i .



Figure 4.9: The figure shows that the concept ‘Season’ is mined 3 times from ConceptNet.

4.4 Time series encoding

Input data have to be reshaped into a form that the network can understand. Therefore, this module aims at creating time series, a data structure composed of many pieces of information that we employ to train a LSTM network. Every piece can be obtained exploiting an embedding or by averaging an ensemble of embeddings. As previously described, an embedding is obtained by mapping a piece of information to a vector of a predefined size, in this way, concepts are projected into a different multidimensional space where they can be processed by our model. The module receives as input a question, the selected answer, the attributes and their weights, the scene graph embedding and its weights. Depending on the architecture modules in use, the time series encoding phase models 3 main types of Input time series: (I) Linear time series, (II) Contextual time series and (III) Vertex time series.

Linear time series

It is the simplest one, formed by the concatenation of attributes embedding, relations embedding, answer concepts embedding and question concepts embedding. The first time step is computed exploiting the data provided by the *Attribute computation* module: a set of attributes and their weights. Attributes are first multiplied by their weights (see Section 4.3). The first time step is then formed by averaging the embeddings of these attributes. The second time step is computed exploiting the data provided by ECE and EAE embedding approaches in *Scene graph embeddings* module: a set of relations and their weights (see Section 4.2.1). Relation embeddings are multiplied by their weights and the time step is formed by averaging the obtained values. The other time steps are computed embedding question and answer concepts. Every concept is exploited for a single time step. In the following paragraph, we formalize this process.

Let $A = [a_1, \dots, a_i, \dots, a_n]$ and $W' = [w'_1, \dots, w'_i, \dots, w'_n]$ be the output of the *Attribute computation* module. Let $R = [r_1, \dots, r_i, \dots, r_n]$ and $W = [w_1, \dots, w_i, \dots, w_n]$ be the output of *Scene graph embedding* module. Let AN be an answer and $[a\hat{n}_1, \dots, a\hat{n}_i, \dots, a\hat{n}_n]$ the embedding of its concepts. Let Q be a question and $[\hat{q}_1, \dots, \hat{q}_i, \dots, \hat{q}_n]$ the embedding of its concepts. Let S be the time series we want to generate and $s_i = | \cdot |$ a time step. We define:

$$S = | \frac{\sum_i w'_i a_i}{\sum_i w'_i} | \frac{\sum_i w_i r_i}{\sum_i w_i} | a\hat{n}_1 | \dots | a\hat{n}_i | \dots | a\hat{n}_n | \hat{q}_1 | \dots | \hat{q}_i | \dots | \hat{q}_n | \quad (4.4)$$

Contextual time series

Every time step is created by stacking question/answer concepts with both attributes and relations embeddings. In this way we enforce the scene graph information by presenting it multiple times in the time series. We refer to these data with the same notation exploited in *Linear time series*. Thanks to *FastText*, every embedding is composed of 300 values. Therefore, every time step has dimension 900.

$$S = | a\hat{n}_i | \parallel \frac{\sum_i w'_i a_i}{\sum_i w'_i} \parallel \frac{\sum_i w_i r_i}{\sum_i w_i} | \dots | \hat{q}_i | \parallel \frac{\sum_i w'_i a_i}{\sum_i w'_i} \parallel \frac{\sum_i w_i r_i}{\sum_i w_i} | \dots | \quad (4.5)$$

Vertex time series

It is created by leveraging on CE Scene graph embedding procedure (see Section 4.2.1). In this case, *Scene graph embedding* module computes a set of embedded vertices (each of them appearing in a distinct time step of the time series). Both Questions/Answer concepts and vertices embedding are stacked with the one related to attributes. Therefore, every time step has dimension 600. We refer to data with

the same notation exploited in *Linear time series*. Let $\hat{G}=[\hat{v}_1,\dots,\hat{v}_i,\dots,\hat{v}_n]$ be the scene graph embedding. The final time series will be:

$$S = | a\hat{n}_i | \left\| \frac{\sum_i w'_i a_i}{\sum_i w'_i} \right\| | \dots | \hat{q}_i | \left\| \frac{\sum_i w'_i a_i}{\sum_i w'_i} \right\| | \dots | \hat{v}_i | \left\| \frac{\sum_i w'_i a_i}{\sum_i w'_i} \right\| | \dots | \quad (4.6)$$

4.5 LSTM backbone

Our neural network architecture is composed of two parts. The first one was created stacking 2 LSTM layers, while the second is composed of two dense layers.

Figure 4.10 depicts our model, where the input (i.e., a time series) is characterized by the shape $((None, None, 300))$. The first dimension refers to the batch size (see Section 2.4). The ablation “None” indicates that this dimension does not have any size constraint. The second dimension indicates the number of time steps of every time series. Also in this case, the ablation “None” specifies the absence of any size constraint. The last one indicates that the input series are composed of embeddings of dimension 300. Figure 4.10 shows that both LSTM layers are characterized by 721’200 parameters, a set of values that are modified during the training phase of the network in order to minimize the error function of the model over the training data. The number of parameters depends on the number of input features that the layer handles. In this case, this value is 300. We estimated it as a compromise between the computational time and the task complexity.

The second part of the backbone is composed of 2 dense layers (see Section 2.4) that are employed to perform a prediction. As the problem is turned into a binary classification one (Correct or Wrong), the last dense layer has only one neuron that can output a value that sweeps between 0 and 1. If the value exceeds 0.5, the sample belongs to the class *Correct*.

| | Layer (type) | Output Shape | Param # |
|--------|-------------------------|---------------------|---------|
| Part 1 | lstm_input (InputLayer) | [(None, None, 300)] | 0 |
| | lstm (LSTM) | (None, None, 300) | 721200 |
| | lstm_1 (LSTM) | (None, 300) | 721200 |
| Part 2 | dense (Dense) | (None, 100) | 30100 |
| | dense_1 (Dense) | (None, 1) | 101 |

Figure 4.10: LSTM backbone

Chapter 5

Dataset, evaluation and implementation details

5.1 VQA dataset

Dataset’s choice is a fundamental step for building a VQA state-of-the-art. Indeed, as we explained in Section 2.1.1, every dataset is characterized by a series of characteristics that can easily bias the training phase of a model. In our case, our goal was employing different types of external knowledge to face a VQA-multiple choice task. Therefore, we tried to consider the datasets directly related to external knowledge bases that provide high quality labeled data. Moreover, our objective was improving reasoning ability of a VQA model. We found in visual7W the sweet spot for all these prerequisites. Indeed, this dataset contains images related to VisualGenome [30], a knowledge base where every picture is enriched with semantic annotations and scene graphs (see Section 2.5). We considered the latter a perfect source for mining useful information able to improve the reasoning capacity of our network.

Visual Genome’s scene graphs are biased toward the spatial description of a

picture. Indeed, the most common relationships are related to the object mutual position (see Figure 5.1). Moreover, Visual Genome’s pictures are often related to people involved in some kind of activity (see Figure 5.2), the perfect scenario for training the reasoning capacity of our model. To further focus on the reasoning task, we considered visual7W’s questions that started with pronoun *why*. Indeed, as Figure 5.3 shows, these are the questions that require the higher response time for humans and so the most complex reasoning. Furthermore, human performance without images on this subset is remarkably higher (44%) than the one obtained answering the other types of questions (35%). This score indicates that many *why* questions encode a fair amount of common sense that humans are able to infer without visual cues. In the following sections, we will describe the dataset and the knowledge base preparation steps.

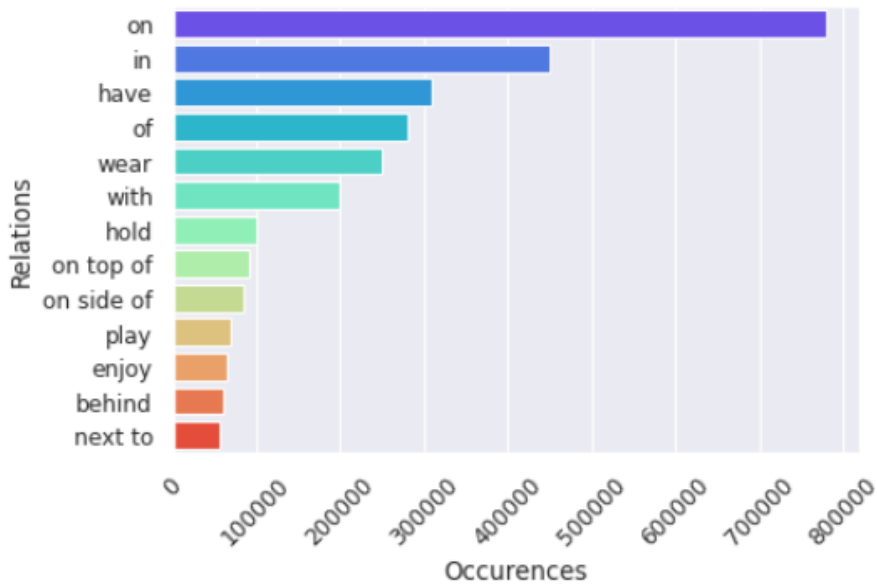


Figure 5.1: The most frequent scene graphs relations

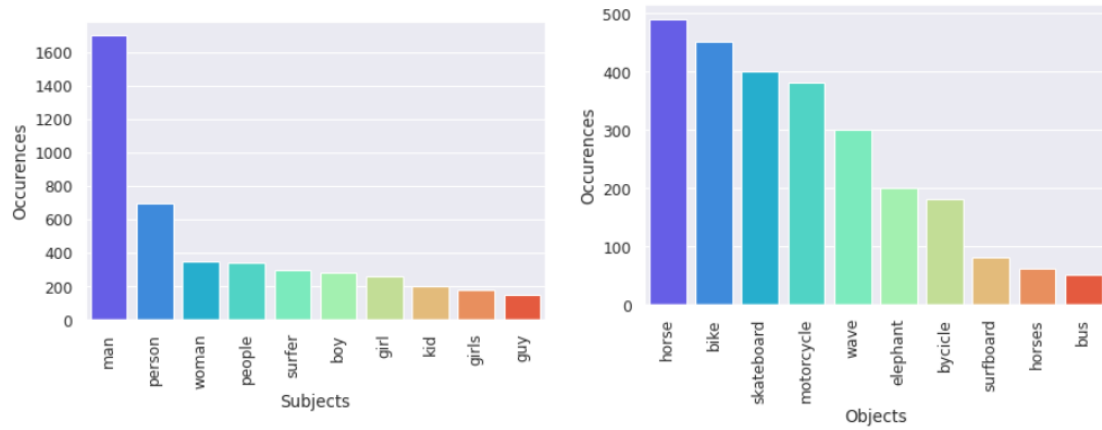


Figure 5.2: On the left, we show the most frequent subjects included in Visual Genome Scene Graphs. On the right, the most frequent objects.

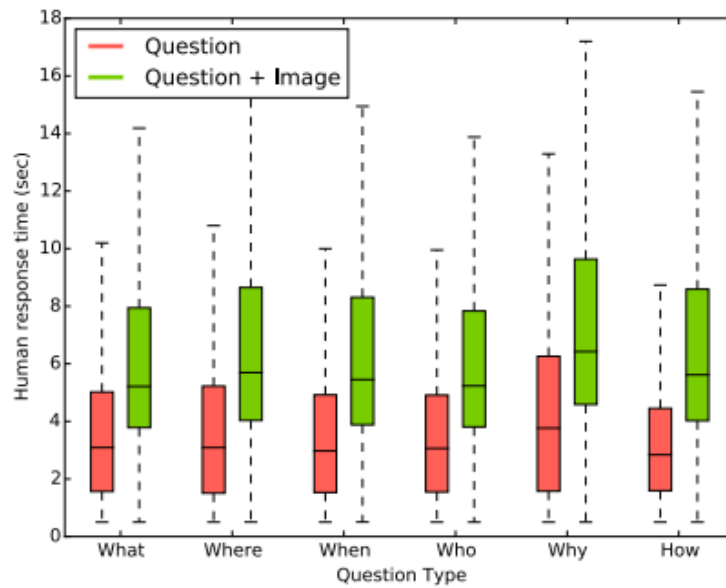


Figure 5.3: Response time required for answering visual7W questions with respect to their type. [36]

Dataset preparation

We started considering only the multiple choice questions related to the type *why*, obtaining a dataset of 8000 questions. To extract concepts, we exploited *Spacy*: a python library for Natural language processing. This library lets us obtain the main concepts of every sentence. The latters are embedded, with the procedure we introduced in Section 4.4. Finally, a binary label is assigned to every question/answer couple, ‘0’ for the wrong answers and ‘1’ for the right one. Therefore, every question is related to 4 samples.

Dataset extension

Dataset is biased towards ‘0’ class samples. To increase the number of samples characterized by label ‘1’, we picked further 5000 open-ended questions from *visual7W*, characterized by the same type *why*. Clearly, we used this extension only on training time.

5.1.1 ConceptNet

In order to use this Knowledge Base, we exploited its local version provided by *Bauer et al.* [2]. Moreover, we decided to consider only a subset of 15 relation types, composed of the most pertinent relations with this task. As Figure 5.4 shows, relation types can be grouped into various thematics. We exploited these groups and the frequency of ConceptNet relations for choosing the 15 relations included in our subset.

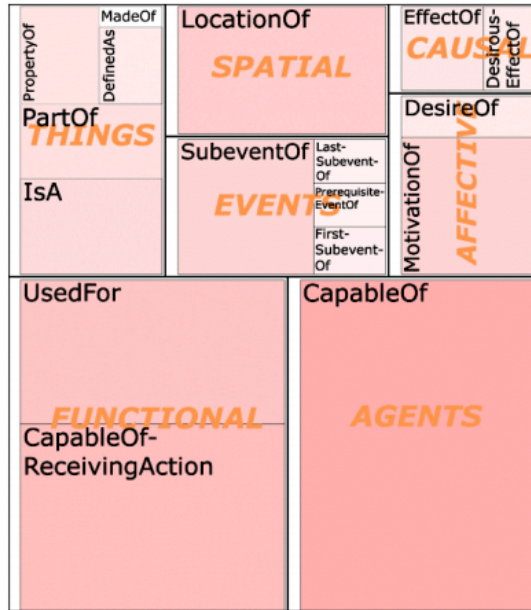


Figure 5.4: The sizes of the rectangles are proportional to the number of assertions belonging to each relation-type in ConceptNet. [4]

5.2 Evaluation and metric

In order to evaluate our model accuracy, we used the K-Fold-Cross Validation. This procedure is very common when the amount of available data is limited. During this approach, the dataset is divided into k subsets. At this point, k algorithm iterations are executed. Every iteration trains a model on $k - 1$ partitions and evaluates it on the remaining one. In the end, the final score will be achieved averaging the score obtained for the k test partitions. In this case, we chose k equal to 3. Multiple-choice questions can be evaluated similarly to multi-class classification tasks. Our model outputs a likelihood value for each answer. The one with the highest score is picked as the final output and compared with the ground-truth. We evaluate the result with the accuracy score (percentage of correct answers). Other metrics, such as precision, recall and F1 are not considered as the

4 possible answers cannot be interpreted as 4 meaningful class labels for which we need separate analysis.

5.3 Implementation Details

In this section, we analyze the setting that we adopted for training our network. As we faced a binary classification problem separately for each answer, we chose a *binary crossentropy* loss. As optimizer, we employed Adam technique (see Section 2.4). Our input time series have not a fixed length but LSTM batches require an input characterized by the same number of time steps (see Section 2.4). Instead of using padding (i.e., empty time steps to create fixed length batches), we trained the model with batches of dimension 1.

The hyperparameter setting varied with respect to the architecture we used. Nevertheless, we executed a series of crucial steps. Regarding the parameter called Learning rate, we performed an initial coarse tuning. Thanks to this starting phase, we were able to focus on a smaller interval of values for a grid search. Leveraging on the GPU provided by the Google’s cloud service *GoogleColab*, the training phase requires several hours and a cross validation attempt almost a day. Therefore, because of hardware limitations we tried to find a *LR* configuration that suited for different cases. Probably, results are slightly biased by a coarse tuning, but performing further tests we did not notice significant score variations.

Chapter 6

Results

In the following chapter, we will describe the results provided by our model. As introduced in Section 4.3, a cardinal step of our work consists in the extraction external concepts from images and from ConceptNet. Figure 6.1 describe the results provided by these two methods. It is composed by 4 histograms, the first 2 depict the most frequently extracted attributes and extra attributes. It's notable that these 2 methods provide a similar result, indeed the extracted terms are both the same or semantically close. The last 2 histograms describe the semantic of the extracted concepts. Indeed, we divided the concepts into 3 types: objects, object features and actions. Attributes tend to represent uniquely the visual elements of the image: its objects and their features. Indeed, the actions that can be involved in that context are less than 0.01%. Hence, the downside of this approach could be a scarce capacity of reasoning about the image. On the other hand, the second method (i.e., extra attributes extraction) slightly overcomes this trend mining also a small number of extra attributes related to actions. Therefore, these charts suggest that by enriching our set of attributes with the ones extracted from ConceptNet we can obtain a more comprehensive vision of the context.

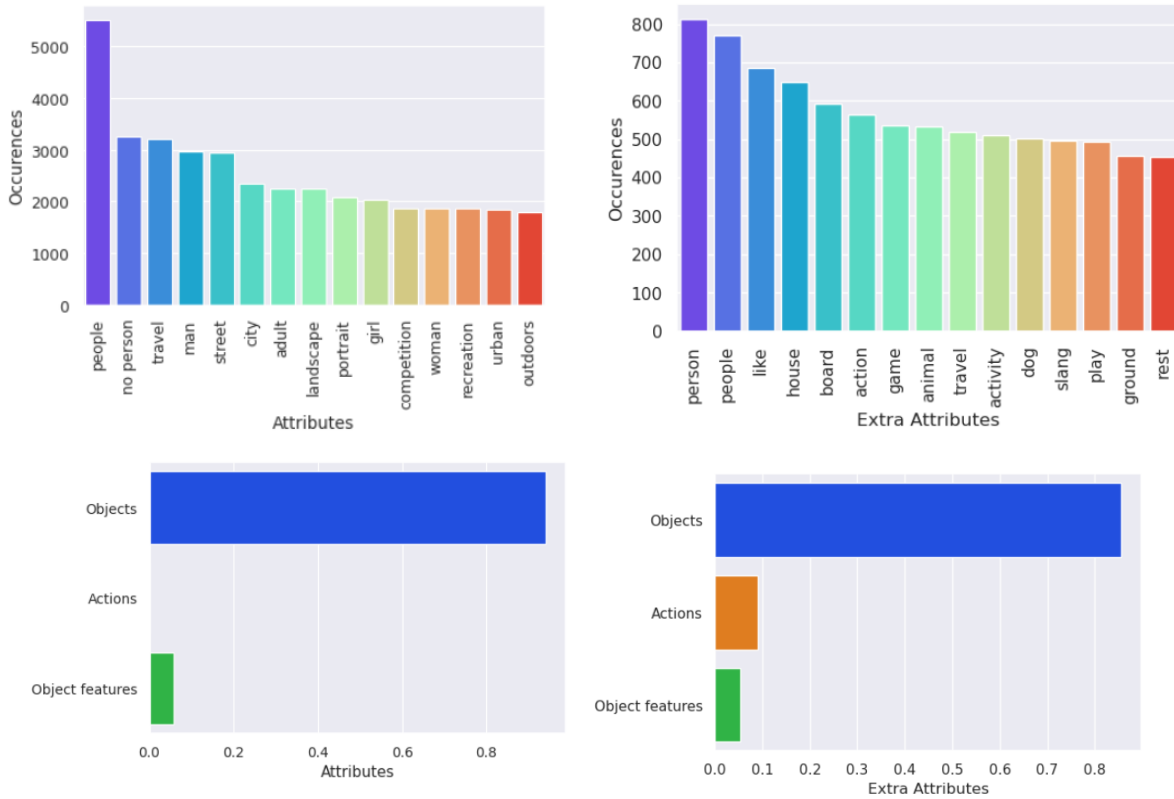


Figure 6.1: The first two histograms depict the most frequently extracted attributes and extra attributes. The last two show the semantic of the extracted concepts.

6.1 Comparison of the proposed methods

To analyze the contributions of each component of our model, we ablate our configurations as follows. *Baseline*, *Attributes*, *Attributes+weights* and *Attributes+Extra Attributes* only use question concepts, answer concepts and attributes. In addition, scene graphs are used for *Attributes+EAE*, *Attributes+EAE+Weights* and *Contextual time series*. Finally *ECE* and *CE* introduce a more structured way of exploiting scene graphs.

Baseline

This configuration considers only Question and Answer concepts. No information is used from the image scene graph. The concepts are encoded as a *Linear Time Series*, not considering the time steps related with Attributes and Scene graphs. We considered the result as a baseline to understand the improvements every module brings. For this configuration, we performed a coarse search on Learning Rate (*LR*) parameter with a range of $[1e-2, 1e-5]$. Then, we ran a finer tuning with a *LR* of $1e-3$, scaling it to a smaller value after 5 epochs. The baseline achieves an accuracy of 53 %. Exploiting this configuration, the model requires 2 hours to be trained.

Attributes

This configuration exploits the external concepts extracted from the images, the question and answer concepts. These ensemble of data is encoded as a *Linear Time Series* ignoring the time step related with scene graphs. During this case, we adopted the same learning rate of $1e-3$ for 6 epochs, reducing it to $1e-5$ for 5 epochs. As Figure 6.1 shows, we obtained a slight accuracy improvement.

Attributes+weights

To perform a weighted mean of concepts embedding, this configuration enhances the “attributes” one by using the attribute weights. In this case,

we maintained the same hyperparameters setting, obtaining an accuracy of 54.25%.

Attributes+Extra Attributes

During this attempt, in addition to question and answer concepts we employ both the attributes and the ones extracted from ConceptNet (i.e., extra attributes). In this case these external concept are not weighted. The ensemble of data is encoded as a *Linear Time Series* ignoring the time step related with scene graphs. Maintaining the same hyperparameters, we obtained an accuracy of 54.27%.

Attributes+EAE

This setting utilizes the question and answer concepts, attributes and relations. For these cases, we adopted the graph embedding procedure *EAE* (see Section 4.2.1). The ensemble of data is encoded as a *Linear Time Series*. The architecture required a finer hyperparameter tuning. We tried different configurations, noticing that varying learning rate in [1e-2, 1e-5] range did not bring any notable improvement. Therefore, for this case and the ones that require a bigger input structure we maintained the *LR*, varying the number of epochs. We trained the system for 8 epochs with a *LR* of 1e-3 and 1e-5 for 15 epochs. With respect to the baseline, we obtained a score improvement (55%). This configuration requires two hours and 30 minutes to be trained.

Attributes+EAE+Weights

For both Attributes and Scene graphs embedding, this trial utilizes the weights and performs a weighted mean as shown in Section 4.4. Weights brought a further improvement to accuracy that let us obtain our best score: 56%. Later in this section, Figure 6.2 depicts the reason why this setting achieves the higher accuracy. Indeed, it demonstrates how important is for the model to be *directed* in the “where to look” process. Picture context is full of noisy elements (e.g., kite, sandcastle, mountain), useless for reasoning about the question. The bold attributes and relations instead, are the ones characterized by a higher weight value. Indeed, they are directly related to the question and answer concepts.

Contextual time series

During this attempt, we exploit Contextual time series (see Section 4.4). We trained the system with the same parameters of the previous cases. Thanks to this structure, we were able to obtain one of the highest scores: 55.9%. We further tried to employ this structure by enriching the attributes with the ones extracted from ConceptNet. We refer to this configuration using the ablation *Contextual time series+Extra attributes*. Because of the input shape, the training phase requires 2 hours and 35 minutes.

ECE

This setting exploits the graph embedding approach *ECE* (see Section 4.2.1). We adopted this method using both *Spatial* and *All* scene graph relations. Maintaining the same two *LR* values, we trained the system for 20 epochs. This module achieves a score of 54.5%.

CE

This configuration exploits the CE approach, described in Section 4.2.1. Probably because of its input structure, this case required a higher number of training epochs. Maintaining the same two *LR* values, we trained the system respectively for 10 and 15 epochs. Table 6.1 shows that this module achieves a low accuracy value, very close to the baseline one (53.2%). In this case, the model requires 2 hours and 55 minutes to be trained.

| # | Model | Accuracy |
|----|--|----------|
| 0 | Baseline | 53.0 |
| 1 | Attributes | 54.1 |
| 2 | Attributes+Weights | 54.1 |
| 3 | Attributes+Extra Attributes | 54.3 |
| 4 | Attributes+EAE | 55.1 |
| 5 | Attributes+EAE+Weights | 56.0 |
| 6 | ECE - Spatial relations | 54.2 |
| 7 | ECE - All relations | 54.0 |
| 8 | CE | 53.2 |
| 9 | Contextual time series | 55.9 |
| 10 | Contextual time series Extra Attributes | 54.1 |

Table 6.1: Table shows accuracy with respect to different attempts.

The Table 6.2 shows that the configurations *Attributes+EAE+Weights* and *Contextual time series* obtain the best results. The first biases the model attention on the most important elements of the image. Concerning this attempt, Figure 6.2 depicts how important is for our model to exploit the weights that we assigned to attributes and scene graphs. Being directed in the “where to look” process avoid considering noisy elements, such as the object “Mountain” or “Kite”. On the other hand, *Contextual time series* enforces both scene graph and attributes information by presenting them multiple times in the time series. Probably, by exploiting both attributes and relations we provide a more comprehensive view of image context. Moreover, the tuning techniques that we adopted in these 2 attempts further enhance the reasoning capacity of our model. In proof of this, the configuration that exploits uniquely scene graphs and attributes (i.e., *attributes+EAE*) achieves a lower accuracy score. The table shows also that the method *CE* performs poorly on this dataset. It is possible that in this case, the scene graph embedding and encoding procedures are too preponderant and they do not allow a generic comprehension

of the context.

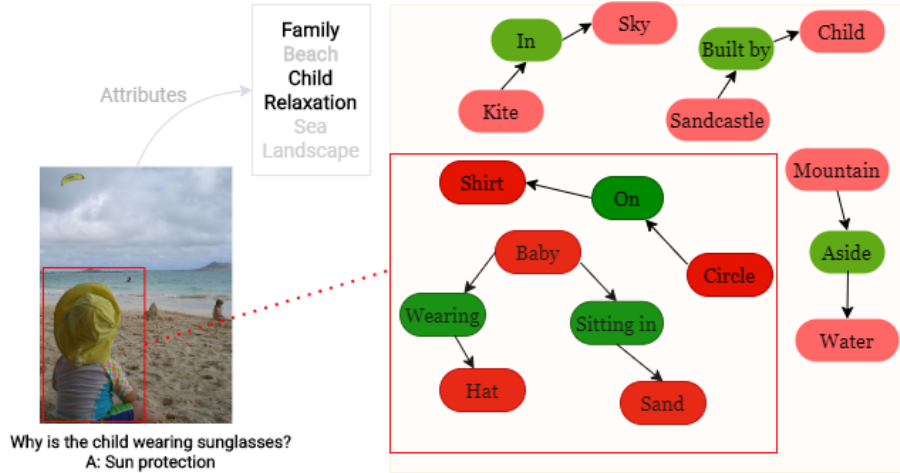


Figure 6.2: Picture shows how we faced the “where to look” problem. The highlighted concepts are the ones characterized by a higher weight value. Indeed, they are directly linked with the question/answer ‘Why is the kid wearing a hat? Sun protection’.

6.2 Qualitative results

For a better understanding of the impact of each architecture module, we quantitatively analyzed the results obtained on the different attempts. A group of answers was always correctly predicted (e.g., Figure 6.3a), while another subset has always been answered wrongly by all models (e.g., Figure 6.3b). Performing a coarse analysis on these two sets in search of a possible structural cue of Question/Answer couples, we did not notice any significant difference related to tokens, answer lengths and concepts. Moreover, assigning a “degree of difficulty” to questions or inferring the reason why the model makes a mistake it’s not a trivial task. Therefore, when comparing the results of different attempts we focused on which type of improvement every component brings. Initially, we expected a progressive enhancement on questions predicted correctly, we supposed that questions answered

correctly using the base model were included among the ones related to a higher accuracy structure. Nevertheless, we noticed a difference among answers related to different trials. In particular, the ones involving scene graphs embeddings remarkably differ from the others. Supposing that different modules improve the reasoning capacity in different ways, we tried to divide questions with respect to the type of reasoning required. In the following sections, we analyze the performance of the architecture exploiting 2 different groups of questions.



Why is the fork on the food?

- A. To be used to hold down for cutting.
- B. To be used to launch food in a food fight.
- C. Used to serve food to dinner guests.
- D. To be used for eating



Why are the little ones following the big one?

- A. To know where to go. ✗
- B. That's the leader.
- C. It's the teacher.
- D. It's their mother.

Figure 6.3: *The first question has always been predicted correctly, in the second case our attempts never provided the right answer. It's easy inferring that the first sample requires trivial reasoning to be answered. On the contrary, the second one can be misleading. Indeed, the model chooses always the wrong answer (A).*

Spatial Reasoning

In addition to accuracy, we wanted to observe how the modules improved the capacity of reasoning of the system. Most of scene graphs edges, involve spatial

relations. Therefore, by exploiting this external knowledge we expected an improvement on questions requiring spatial reasoning. For this experiment, among the questions of our validation sets we picked the subset involving spatial relations (*behind, on, in front of, ...*), analyzing the score variations on this subset. As Table 6.2 shows, by exploiting these questions the baseline model achieves an accuracy of about 50%. As expected, modules involving scene graphs embeddings (highlighted in bold in Table 6.2) are the ones that obtain the higher accuracy and the most pertinent answers (see Figure 6.4). It is interesting to note a slight worsening related to modules that use only external attributes (i.e., 1-3). On the contrary, modules related to the two more elaborated scene graph embeddings approaches (*ECE* and *CE*) brought a sensible improvement on this subset. It is also notable that *CE* and *ECE* do not achieve good performances on the whole dataset, but in this case they show a higher capacity of improving spatial reasoning, higher or equal to the one obtained on the whole dataset.

| # | Model | Accuracy |
|----|---|----------|
| 0 | Baseline | 51.4 |
| 1 | Attributes | 51.7 |
| 2 | Attributes+Weights | 51.2 |
| 3 | Attributes+Extra Attributes | 51.1 |
| 4 | Attributes+EAE | 51.8 |
| 5 | Attributes+EAE+Weights | 53.4 |
| 6 | ECE - Spatial relations | 55.7 |
| 7 | ECE - All relations | 51.9 |
| 8 | CE | 53.0 |
| 9 | Contextual time series | 51.1 |
| 10 | Contextual time series+ Extra Attributes | 52.6 |

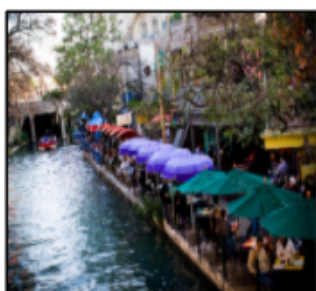
Table 6.2: Accuracy on questions involving spatial reasoning



Q: Why are people walking around town?

- A. Sightseeing
- B. Exercise
- C. Play
- D. Shopping

| # | Model | Answer |
|----|--|--------|
| 0 | Baseline | C |
| 1 | Attributes | A |
| 2 | Attributes+Weights | A |
| 3 | Attributes+Extra attributes | A |
| 4 | Attributes+EAE | A |
| 5 | Attributes+EAE+Weights | D |
| 6 | ECE - Spatial relations | C |
| 7 | ECE - All relations | D |
| 8 | CE | A |
| 9 | Contextual time series | C |
| 10 | Contextual time series+ Extra Attributes | A |



Q: Why are people sitting at table?

- A. Drinking
- B. Playing games
- C. Watching a band play
- D. Eating

| # | Model | Answer |
|----|--|--------|
| 0 | Baseline | B |
| 1 | Attributes | C |
| 2 | Attributes+Weights | A |
| 3 | Attributes+Extra Attributes | C |
| 4 | Attributes+EAE | D |
| 5 | Attributes+EAE+Weights | D |
| 6 | ECE - Spatial relations | D |
| 7 | ECE - All relations | D |
| 8 | CE | D |
| 9 | Contextual time series | D |
| 10 | Contextual time series+ Extra Attributes | D |


Figure 6.4: The figure shows two examples of Spatial reasoning questions. They stress how the relation’s embeddings improves spatial reason capacity of the system. Indeed, in this case the Baseline and “attribute” architectures (i.e., 0-3) tend to answer wrongly.

Context reasoning

In general, *why* questions are the ones that require a higher capacity of reasoning. Nevertheless, we considered a further subset, different from the spatial reasoning one, to analyze which module tended to improve the most the context reasoning capacity. To do so, we considered the questions related to a subject doing an action (i.e., presence of at least a sentence of type: *subject + verb + '-ing' + object*) as the most appropriate for this task. This subset composes the 30% of the dataset and it has only a small amount (15%) of common samples with the previous one. As we exploited external concepts, we expected that they would produce the best results on this task. In general, every module brought an improvement on these questions. In contrast to the previous case, *scene graph embeddings* did not bring any significant enhancement. As expected, modules related to external concepts achieve a higher score with respect to the previous case (see Table 6.3). As Figure 6.6 shows, also in this case the *Attributes+EAE+weights* and *Contextual time series* seems the best configuration for enhancing the capacity of the system to reason about the image context. Also in this case, it is interesting to note a discrepancy between the dataset score and the subset one. In conclusion, different modules bring an improvement on different types of questions. Therefore, involving both attributes and scene graphs and trying to find a sweet spot between these two approaches could be the best way of improving the reasoning capacity. In proof of this, the configuration related to their concurrent usage is the one that achieves the higher accuracy.

| # | Model | Accuracy |
|----|--|----------|
| 0 | Baseline | 52.3 |
| 1 | Attributes | 55.8 |
| 2 | Attributes+Weights | 55.1 |
| 3 | Attributes+Extra Attributes | 55.5 |
| 4 | Attributes+EAE | 55.9 |
| 5 | Attributes+EAE+Weights | 56.6 |
| 6 | ECE - Spatial relations | 53.8 |
| 7 | ECE - All relations | 53.9 |
| 8 | CE | 54.3 |
| 9 | Contextual time series | 56.7 |
| 10 | Contextual time series Extra Attributes | 55.5 |

Table 6.3: Accuracy on questions involving context reasoning




Q: Why is the flowers and vase casting a shadow?

A. The sun
B. Their position
C. They are in the way
D. Lights

| # | Model | Answer |
|----|---|--------|
| 0 | Baseline | C |
| 1 | Attributes | D |
| 2 | Attributes+Weights | D |
| 3 | Attributes+Extra Attributes | D |
| 4 | Attributes+EAE | C |
| 5 | Attributes+EAE+Weights | A |
| 6 | ECE - Spatial relations | A |
| 7 | EAE - All relations | D |
| 8 | CE | B |
| 9 | Contextual time series | A |
| 10 | Contextual time series+ Extra Attributes | D |

Figure 6.5: The figure shows that the configurations that exploit only the attributes provide always the right answer.



Q: Why is the player tilting backward?

A. To catch the ball
 B. To propel himself forward
 C. To see the crowd
 D. To better hit the ball with the racket

| # | Model | Answer |
|----|---|--------|
| 0 | Baseline | B |
| 1 | Attributes | D |
| 2 | Attributes+Weights | D |
| 3 | Attributes+Extra Attributes | D |
| 4 | Attributes+EAE | B |
| 5 | Attributes+EAE+Weights | C |
| 6 | ECE - Spatial relations | B |
| 7 | ECE - All relations | C |
| 8 | CE | B |
| 9 | Contextual time series | A |
| 10 | Contextual time series+ Extra Attributes | B |

Figure 6.6: *The examples are related to Context reasoning. What we noticed is that most of the times Answer is given correctly (D) ✓ by “attribute” modules. On the contrary, in some cases relations tend to bias the prediction (e.g., Figure (b) (B) ✗).*

Comparison to State-of-the-Art

Table 6.4 shows that our results are in line with the ones obtained with similar approaches [23, 25]. On the other hand, we note that recent state-of-the-art exploit different methods that outperform the ones based on LSTM networks. Even if our model achieves an acceptable score on visual7W panorama, it must be said that in contrast to the listed state-of-arts our architecture is trained uniquely on ‘why’ question. Therefore, further tests should be carried out in order to verify that the model does not simply learn and exploit biases in the distribution of answers.

| Method | Accuracy |
|----------------------------------|----------|
| Semantic aware VQA | 56.0 |
| LSTM (Question+Images) [25] | 50.3 |
| LSTM - Att (question+Image) [23] | 55.5 |
| MLP (A+Q+I) [14] | 70.0 |
| Ensemble model [29] | 67.0 |

Table 6.4: Table shows the accuracy achieved on ‘why’ questions by most recent state-of-arts

Chapter 7

Conclusions and future works

In this work, we investigated a way of integrating external knowledge in a VQA-system. In particular, we demonstrated how the concurrent usage of Scene graphs and ConceptNet can improve the reasoning capacities of an architecture. Our experimental results show that scene graphs can definitively benefit the Visual QA task, especially if the reasoning requires the object spatial relationships. At the same time, it shows how small tuning techniques with ConceptNet can “direct” and improve the reasoning capabilities of our system.

Clearly, this work has not the aim of outclassing the recent VQA-models. We neither exploited complex modules nor we disposed of the hardware required for an exhaustive analysis. We tried instead to focus our efforts toward new computationally light mining approaches, capable of improving the way with which external knowledge sources are employed.

In a recent conference [11], the authors of the VQA challenge affirmed that future works must shift their focus on extracting more information from the images. Nevertheless, we believe that a finer usage of external knowledge sources could still benefit this research area.

Indeed, in the future our intuitions could be studied in depth and integrated into

more sophisticated systems. For example, models that employ attention mechanisms could sharpen the weighting methods that we adopted, paving the way to new approaches able to exploit common knowledge bases for this purpose. Further attempts could also improve the usage of encoders, combining them with the extraction of contextually relevant concepts from ConceptNet.

Bibliography

- [1] Stanislaw Antol et al. “Vqa: Visual question answering”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.
- [2] Lisa Bauer, Yicheng Wang, and Mohit Bansal. “Commonsense for generative multi-hop question answering tasks”. In: *arXiv preprint arXiv:1809.06309* (2018).
- [3] Jeffrey P Bigham et al. “VizWiz: nearly real-time answers to visual questions”. In: *Proceedings of the 23rd annual ACM symposium on User interface software and technology*. 2010, pp. 333–342.
- [4] Stephan Bloehdorn et al. “Organization Workshop Organizers”. In: (Jan. 2008).
- [5] *Clarifai*. <https://www.clarifai.com/>.
- [6] *Convolutional Neural Networks*. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [7] Abhishek Das et al. “Human attention in visual question answering: Do humans and deep networks look at the same regions?” In: *Computer Vision and Image Understanding* 163 (2017), pp. 90–100.
- [8] Abhishek Das et al. “Visual dialog”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 326–335.

- [9] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).
- [10] Haoyuan Gao et al. “Are you talking to a machine? dataset and methods for multilingual image question”. In: *Advances in neural information processing systems*. 2015, pp. 2296–2304.
- [11] Yash Goyal et al. “Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6904–6913.
- [12] Marcel Hildebrandt et al. “Scene Graph Reasoning for Visual Question Answering”. In: *arXiv preprint arXiv:2007.01072* (2020).
- [13] Kushal Kafle and Christopher Kanan. “Visual question answering: Datasets, algorithms, and future challenges”. In: *Computer Vision and Image Understanding* 163 (2017), pp. 3–20.
- [14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [15] Ranjay Krishna et al. “Visual genome: Connecting language and vision using crowdsourced dense image annotations”. In: *International journal of computer vision* 123.1 (2017), pp. 32–73.
- [16] Guohao Li, Hang Su, and Wenwu Zhu. “Incorporating external knowledge to answer open-domain visual questions with dynamic memory networks”. In: *arXiv preprint arXiv:1712.00733* (2017).
- [17] Bill Yuchen Lin et al. “Kagnet: Knowledge-aware graph networks for commonsense reasoning”. In: *arXiv preprint arXiv:1909.02151* (2019).

- [18] Hugo Liu and Push Singh. “ConceptNet—a practical commonsense reasoning tool-kit”. In: *BT technology journal* 22.4 (2004), pp. 211–226.
- [19] *LSTM input*. https://github.com/MohammadFneish7/Keras_LSTM_Diagram. May 2013.
- [20] *LSTM Neural Networks*. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. May 2013.
- [21] Chih-Yao Ma et al. “Attend and interact: Higher-order object interactions for video understanding”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6790–6800.
- [22] Mateusz Malinowski and Mario Fritz. “A multi-world approach to question answering about real-world scenes based on uncertain input”. In: *Advances in neural information processing systems*. 2014, pp. 1682–1690.
- [23] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. “Ask your neurons: A neural-based approach to answering questions about images”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1–9.
- [24] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. “Listen, attend, and walk: Neural mapping of navigational instructions to action sequences”. In: *arXiv preprint arXiv:1506.04089* (2015).
- [25] Alejandro Newell and Jia Deng. “Pixels to graphs by associative embedding”. In: *Advances in neural information processing systems*. 2017, pp. 2171–2180.
- [26] *Recurrent Neural Networks*. <https://davideliu.com/2020/02/29/rnn-recurrent-neural-networks/>. May 2013.
- [27] Mengye Ren, Ryan Kiros, and Richard Zemel. “Exploring models and data for image question answering”. In: *Advances in neural information processing systems*. 2015, pp. 2953–2961.

- [28] Jiaxin Shi, Hanwang Zhang, and Juanzi Li. “Explainable and explicit visual reasoning over scene graphs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8376–8384.
- [29] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).
- [30] *Visualgenome*. <https://visualgenome.org/>. May 2013.
- [31] Qi Wu et al. “Image captioning and visual question answering based on attributes and external knowledge”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1367–1381.
- [32] Zhibiao Wu and Martha Palmer. “Verb semantics and lexical selection”. In: *arXiv preprint cmp-lg/9406033* (1994).
- [33] Dongfei Yu et al. “Multi-level attention networks for visual question answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4709–4717.
- [34] Matthew D Zeiler. “Adadelta: an adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701* (2012).
- [35] Wanjun Zhong et al. “Improving question answering by commonsense-based pre-training”. In: *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer. 2019, pp. 16–28.
- [36] Yuke Zhu et al. “Visual7w: Grounded question answering in images”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4995–5004.