AUTONOMOUS UNIVERSITY OF MADRID

HIGHER POLYTECHNIC SCHOOL

MASTER'S FINAL PROJECT

# Application of Multimodal Machine Learning to Visual Question Answering

Master's Degree in ICT Research and Innovation (i$^2$-ICT)

Author: Carlos Galve Mateo
Tutor: Julián Fiérrez Aguilar
Electronics Technology and Communications Department

DATE: SEPTEMBER 2021

# APPLICATION OF MULTIMODAL MACHINE LEARNING TO VISUAL QUESTION ANSWERING

Author: Carlos Galve Mateo

Tutor: Julián Fiérrez Aguilar

Biometrics and Data Pattern Analytics Lab

Electronics Technology and Communications Department

SEPTEMBER 2021

# Abstract

## Abstract

Due to the great advances in Natural Language Processing and Computer Vision in recent years with neural networks and attention mechanisms, a great interest in VQA has been awakened, starting to be considered as the "Visual Turing Test" for modern AI systems, since it is about answering a question from an image, where the system has to learn to understand and reason about the image and question shown. One of the main reasons for this great interest is the large number of potential applications that these systems allow, such as medical applications for diagnosis through an image, assistants for blind people, e-learning applications, etc.

In this Master's thesis, a study of the state of the art of VQA is proposed, investigating both techniques and existing datasets. Finally, a development is carried out in order to try to reproduce the results of the art with the latest VQA models with the aim of being able to apply them and experiment on new datasets.

Therefore, in this work, experiments are carried out with a first VQA model, MoViE+MCAN [1] [2] (winner of the 2020 VQA Challenge), which after observing its non-viability due to resource issues, we switched to the LXMERT Model [3], which consists of a pre-trained model in 5 subtasks, which allows us to perform fine-tunnig on several tasks, which in this specific case is the VQA task on the VQA v2.0 [4] dataset.

As the main result of this Thesis we experimentally show that LXMERT provides similar results to MoViE-MCAN (the best known method for VQA) in the most recent and demanding benchmarks with less resources starting from the pre-trained model provided by the GitHub repository [5].

## Key words

Visual question answering, Computer vision, Natural language processing, Multimodal machine learning, Image featurization, Question featurization, Fusion techniques, Attention mechanisms

# Acknowledgements

I would like to thank my tutor, Julián Fiérrez Aguilar, for all the help he has given me, advising and guiding me throughout the work. Also to professors Aythami Morales Moreno and Roberto Daza García for all the help they are giving me in the creation of a new dataset for VQA based on edBB [6].

To my company Accenture, especially to Jesús Figueres Cañadas, who was the person who encouraged me to try to enter the world of research by introducing me to the option of doing an industrial doctorate.

And finally, I would also like to thank my family and friends for all the support they have given me.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation

With a view to the completion of an industrial doctorate in collaboration with Accenture and the Autonomous University of Madrid, we first decided to investigate Machine Learning techniques on multimodal data streams (audio, video, tabulated information, graphs, etc.) and their application to the problem of generating automatic responses to visual information (Visual Question Answering).

The motivation for this line of research arises mainly from the vision of Dr. Andrew Fitzgibbon [22], who recently gave a visionary talk at an international research event held at the EPS of the UAM [23]. From his vision it is clear that there is a compelling need for research, and great opportunities for both scientific and industrial impact, in the study and advancement of machine learning methods on heterogeneous data beyond temporal sequences, images, videos, or structured data; what Dr. Andrew called "all-data AI" [22].

So, we decided that the line of research and innovation would be multimodal machine learning on heterogeneous data, trying to simultaneously exploit the flow of information from different data modalities. After deciding on the field of research, we started by reading the paper [24], which summarises well the state of the art in this field. The reading of this work allowed us to further refine the research focus, with the current idea of focusing on an experimental level on multimodal models for the generation of automatic responses based on visual information (Visual Question Answering).

Given that beginning and the kind of research projects currently being developed at the BiDA Lab (UAM), which include AI tools for improving e-learning platforms, we fixed the following **objectives** for this Master's Final Project: **study of the state of the art in Visual Question Answering (VQA) and its application to e-learning platforms that include audiovisual data of the students**. New technologies in VQA may improve the quality of teaching in online platforms by facilitating the detection of different events, e.g.: detection of loss of attention, detection of the number of people or the main activity in the scene, etc.

Therefore, machine learning with neural networks will be studied, especially focused on text and image processing.

## 1.2 Objectives

The objectives of this Master's Final Project are aimed at laying the foundations for the subsequent completion of an industrial doctorate, which are as follows:

- **Study of the state of the art in Visual Question Answering (VQA)**: a study of the state of the art in VQA will be carried out, where the different techniques and models will be analysed.
- **VQA Challenge**: the different novel techniques with respect to VQA that have been published at the different workshops in CVPR conferences will be analysed and studied with the aim of continuing with the line of research that obtains the best results.
- **Experimental approach to the state of the art in VQA**: after observing the best results obtained from the VQA competition organised at the CVPR conferences (i.e., the best known methods for VQA), we will try to replicate results by implementing an environment with sufficient capabilities to be able to run the codes training neural networks, subsequently trying to find some improvement or application in other data.

## 1.3 Methodology and work plan

In order to achieve the objectives set, a work plan has been drawn up consisting of the following milestones:

- To study the state of the art in VQA with the aim of finding important workshops and papers with public data and results.
- To choose the VQA model as a starting point.
- To become familiar with the format of the usual data processed in VQA multimodal models (e.g., the COCO format).
- To become familiar with the used frameworks in VQA models.
- To replicate key state-of-the-art results.
- To try fine-tuning or introduce small modification to get improved results on the same dataset (VQA-v2.0) or other.
- Analysis of the results obtained.
- Drafting of this document and preparation of the defence.

## 1.4 Structure of the Dissertation

- *Chapter 1*: It describes the motivation, the objectives that were set, and how this work is structured.
- *Chapter 2*: It summarises the state of the art of VQA.
- *Chapter 3*: It describes the datasets used in the experiment of this Master's Final Project.
- *Chapter 4*: It describes the VQA model used, explaining in detail how it works.
- *Chapter 5*: It presents the results obtained by carrying out the experiments described in this chapter.
- *Chapter 6*: It sets out the conclusions reached and the possible future challenges for further progress.

# 2 State of the art

## 2.1 Introduction

The rise of deep learning in recent years with the great advances in complex tasks of Computer Vision (CV) [25][26][27][28][29] and Natural Language Processing (NLP) [30][31][32][33][28][15] has generated a great interest in the field of Visual Question Answering (VQA), which is considered an AI-complete task [8] where the system generates a textual answer from an image and a question, which requires multi-modal knowledge. Therefore, a good simple and schematic representation of the scope and definition of VQA would be the figure 2.1 (taken from [12]):

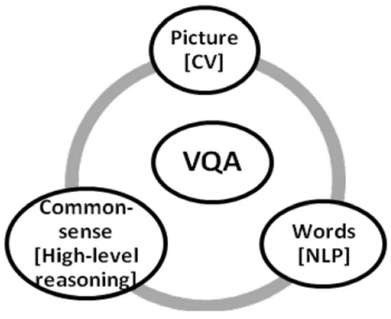


**Figure 2.1:** Definition of VQA, taken from [12]

Another aspect for which the field of VQA has taken great interest is the immense amount of potential applications, among which the following stand out: AI-based medical image understanding and related medical questions-answering (med-VQA), Assistance to blind people, video-surveillance scenarios, education, etc. [13].

To understand the challenges and problems in VQA, it is necessary to know the generic structure of VQA algorithms, which consists of a first process of extracting features from the images and from the questions, carried out independently. This is followed by a process of interaction between the features of the two modalities that facilitates the identification of the important features, in order to generate the answer to the initial question. The following figure 2.2 (taken from [13]) shows the explained flow, as well as the different subtasks addressed in VQA.

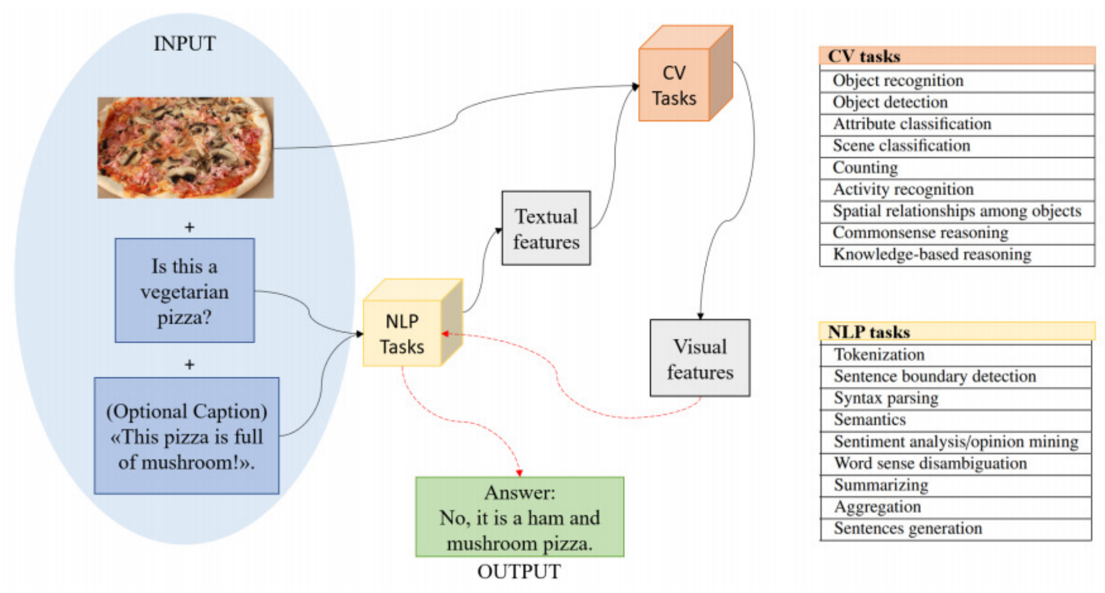


**Figure 2.2:** Flow of VQA and sub-tasks, taken from [13]

After this brief introduction to VQA, each of the phases of the models will be addressed in more detail, followed by a review of the most relevant datasets and metrics currently available. Finally, we will take a look at the VQA Challenge competition that has been held every year since 2016, and whose most relevant results are presented in the VQA workshop at the CVPR conference, ending with an explanation of the VQA Challenge.

## 2.2 Image featurization

One of the most important tasks in VQA is image featurization, which consists in translating an image as a numerical tensor where different mathematical operations can be applied. There are many techniques to carry out image featurization, from the most classical ones such as RGB vector representation to the most recent ones with neural networks. However, since the rise of deep learning, convolutional neural networks (CNN) have dominated, as they allow image featurization to be performed implicitly.

Among the pre-trained convolutional networks applied in VQA models for image featurization, VGGNet [25] and ResNet [26] stand out above all, where VGGNet dominated at the beginning due to its greater simplicity and speed of convergence in fine-tunnig, but ResNet has finally been the predominant one due to the great advances in hardware [12]. In fact, it can be observed that the last 4 winners of the VQA challenge [10] have used ResNet as Baseline for the image featurization [34][35][11][1].

As can be seen, all of these latter winning models actually use the image features (regions [34][35][11] or grid [1]) obtained from the bottom-up attention mechanism [36] of a Faster R-CNN [37] (object detector) pre-trained on the Visual Genome dataset [9], whose baseline is a CNN, specifically a ResNet as shown in figure 2.3 taken from [14].

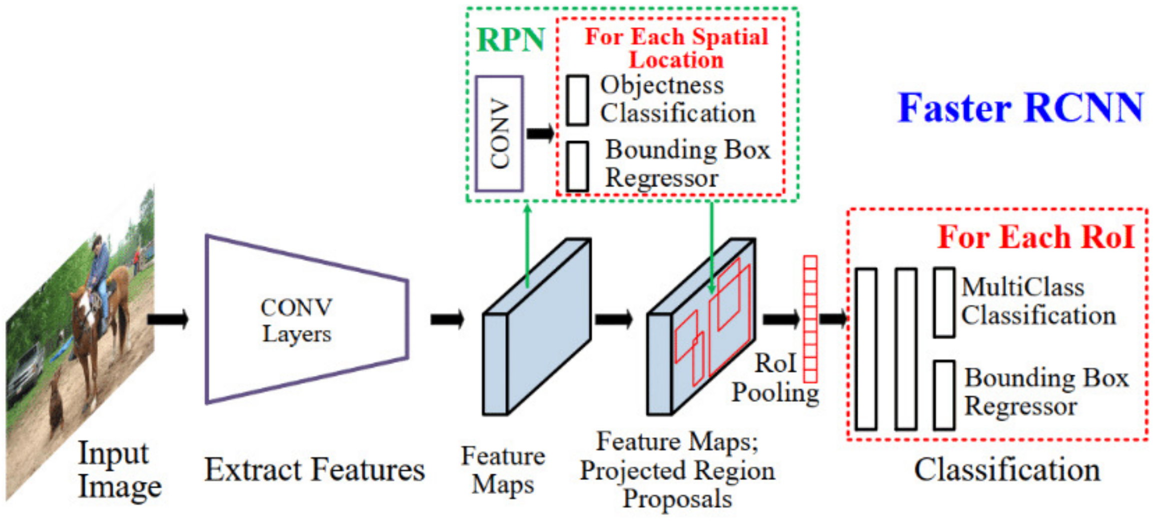


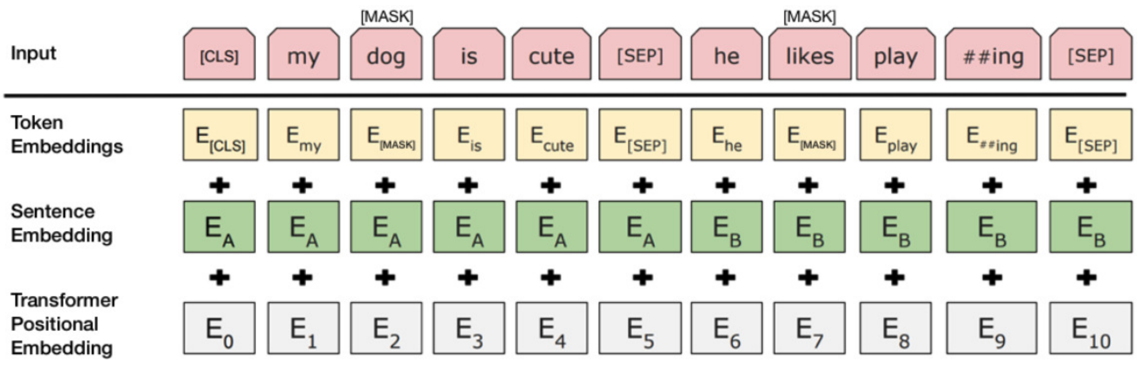
**Figure 2.3:** High-level diagram of Faster R-CNN, taken from [14]

However, the fact of using image features (region features) of an object detection model as a Faster RCNN has some limitations among which the fact of being restricted to the limited number of categories present in the dataset used in the training, which implies a loss of generalization as mentioned in the works of Pixel-Bert [38] or in E2E-VLP [39]. Therefore, it is believed that future VQA models will follow the research lines of the latter works, where end-to-end models are used that apply transformers directly from image (pixels) and text (tokens) inputs. Another great advantage of these models is the computational efficiency in inferring new predictions, as they are one single stage models [39].

## 2.3 Question featurization

Other key task in VQA is question featurization: translating string or text plain to numerical tensors where different mathematical operations can be applied. There are many techniques to carry out question featurization, from the most classical ones such one-hot encoding, matrix concurrence + SVD [40] until most modern as word2vec embedding [31][32], GloVe embedding [33], fastText embedding [41], LSTM embedding [30] or transformer embedding such as BERT [15]. The main advantages of these latter question feature techniques are that they capture more semantic, morphological and contextual information.

As in image featurization, in the studies of the last few years there is a predominance of deep learning in question featurization where the VQA challenge winning models from 2017 have used an embedding+GRU [34] [35] or embedding+LSTM [11] techniques. However, after reading the work of [39], it is considered that future models will go in the direction of end-to-end models where the embedding of tokens will follow the logic of BERT by assigning each token three embeddings (token, segment, position) as shown in figure 2.4 taken from [15].



Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ $E_{my}$ $E_{[MASK]}$ $E_{is}$ $E_{cute}$ $E_{[SEP]}$ $E_{he}$ $E_{[MASK]}$ $E_{play}$ $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
$E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_B$ $E_B$ $E_B$ $E_B$ $E_B$
Transformer Positional Embedding
$E_0$ $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ $E_6$ $E_7$ $E_8$ $E_9$ $E_{10}$

**Figure 2.4:** BERT embedding [15]

## 2.4 Joint comprehension of image and text

One of the fundamental aspects that characterise the VQA as multimodal and highly complex task is that it requires an understanding of the relationships between two modalities (question features and image features). Therefore, it will be necessary to include this knowledge in the model, where among all possible methodologies, the following stand out built on [16]:

### 2.4.1 Fusion based on simple vector operation

These methods can be summarised for simplicity in three types, vector concatenation, element-wise addition and element-wise multiplication, where the latter two require compatibility between dimensions, i.e. they need to be of the same dimension as they are element-wise operations. In case they were not of the same dimension, it would be necessary to apply a linear projection ($v_I = W_i v_I$ and $v_Q = W_q v_Q$).

Summarising, the three methods of simple vector operations for the fusion ($v_F$) for the joint comprehension of image and text would be:

- vector concatenation: $v_F = v_I \parallel v_Q$, that is used in [42] [43] [44] [45].
- element-wise addition: $v_F = v_I \oplus v_Q$, that is used in [46] [47] [48] [49] [50] [51] [52].
- element-wise multiplication: $v_F = v_I \quad v_Q$, that is used in [8] [53] [36] [54] [55] [56] [57].

However, using these simple vector operations to fusion two channels is usually not very effective, so it is more common to apply them on features where attention methods have been applied or to use other fusion techniques.

### 2.4.2 Fusion based on bi-linear models

In order to obtain more information and more complex interactions between the two channels (Image and Text), the use of bilinear pooling was proposed, which consists of the outer product of the feature vectors ($v_I$ and $v_Q$), providing a multiplicative interaction between all the elements of both vectors. However, directly applying bilinear pooling would be very costly, since if for example you have an image vector ($v_I$) of dimension 2048, a text vector ($v_Q$) of dimension 2048, you would obtain a 2048x2048 matrix, which if connected to the 3000 classes by a matrix of learnable parameters, you would obtain approximately 12.5 billion learnable parameters.

For this reason, different dimensionality reduction techniques are sought in order to be able to apply bilinear pooling, highlighting the work of:

- [58] where multimodal compact bilinear pooling (MCB) is proposed, which applies dimensionality reduction by projecting the image and text features randomly in a common space by count sketch function, allowing the outer product not to be applied explicitly.
- [59] where multimodal low-rank bilinear pooling method (MLB) is proposed which achieves further dimensionality reduction (as the authors considered that MCB still required high spatial dimensionality) by rewriting the weight matrix as the multiplication of two smaller matrices.
- [60] where multimodal factorized bilinear model (MFB) is proposed which is practically an improvement of MCB with better stability in training.
- [58] where multimodal tensor-based Tucker decomposition (MUTAN) which decomposes the weight tensor in the bilinear model into three factor matrices and a central tensor.

### 2.4.3 Fusion based on neural networks

Another fusion technique that would allow the recovery of more complex and non-linear interactions between the two channels, image and text, would be neural networks, where LSTMs and CNNs stand out for carrying out the fusion of features from both modalities.

Within fusion methods based on LSTM, there are different techniques such as embedding the image feature $v_I$ as if it were the first word within the word embedding of the question, leaving the input as $(v_I, v_{Q_1}, v_{Q_2}, v_{Q_3}, \ldots, v_{Q_m})$ [61], concatenate the image feature $v_I$ with each of the word embeddings, increasing the input to $([v_I, v_{Q_1}], [v_I, v_{Q_2}], [v_I, v_{Q_3}], \ldots, [v_I, v_{Q_m}])$ [62], project the image feature $v_I$ as the first and last word within word embedding space leaving the input as $(v_I, v_{Q_1}, v_{Q_2}, v_{Q_3}, \ldots, v_{Q_m}, v_I)$ [63], etc.

In order to counteract the fading of the image effect at each LSTM step in the relationship between the projected image as a word and the other semantic representations, a CNN [64] was proposed as an alternative. Some outstanding examples where this type of fusion based on multimodal convolutional neural networks is used are [65] where they use a VGG-16 structure [25] removing the last layer to obtain the image features and add 3 new fully-connected (FC) layers where the second of them consists of dynamic parameters that come from a GRU that extracts the textual features, [66] where they use a hybrid convolution that consists of performing a convolution on the image features whose kernels are guided by the textual features, etc.

### 2.4.4 Attention mechanisms

Attention mechanisms are widely used in VQA tasks because they allow to efficiently extract meaningful regions from images and important terms from questions, improving the interaction between images and text by reducing noise and allowing to answer fine-grained questions [60] [67] [68]. Within the mechanisms of attention in the VQA task, three types of attention are distinguished: visual attention, textual attention and co-attention, where in [50] the use of stacked attention network was proposed to allow learning attention iteratively.

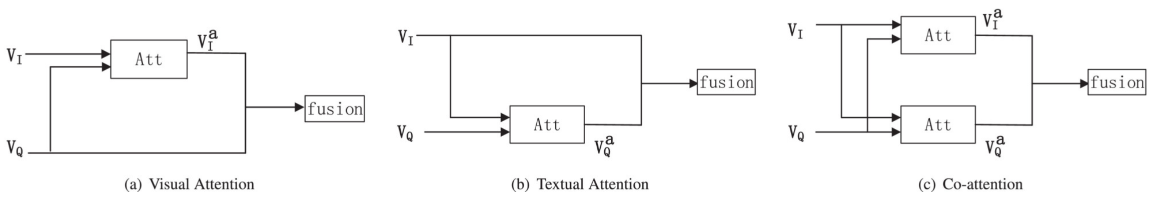


**Figure 2.5:** Single-layer attention strategies taken from [16]

Within the three types of attention models, co-attention models stand out as showing the most accurate results in recent work [67] [68]. In fact, the latest VQA Challenge winning models of 2019 [11] and 2020 [2] [1] are based on Deep Modular Co-Attention Networks models where inspired by Transformers models [28] use cascading layers of self-attention (SA) units (intra-modal interactions) and guide-attention (GA) units (inter-modal interactions) based on the scaled dot-product attention [28]:

$$\text{Attention}(q, K, V) = \text{softmax}(\frac{qK^T}{\sqrt{d}})V$$

Where to improve the representation, multi-head attention is employed which consists of paralleled 'heads'.

## 2.5 Datasets

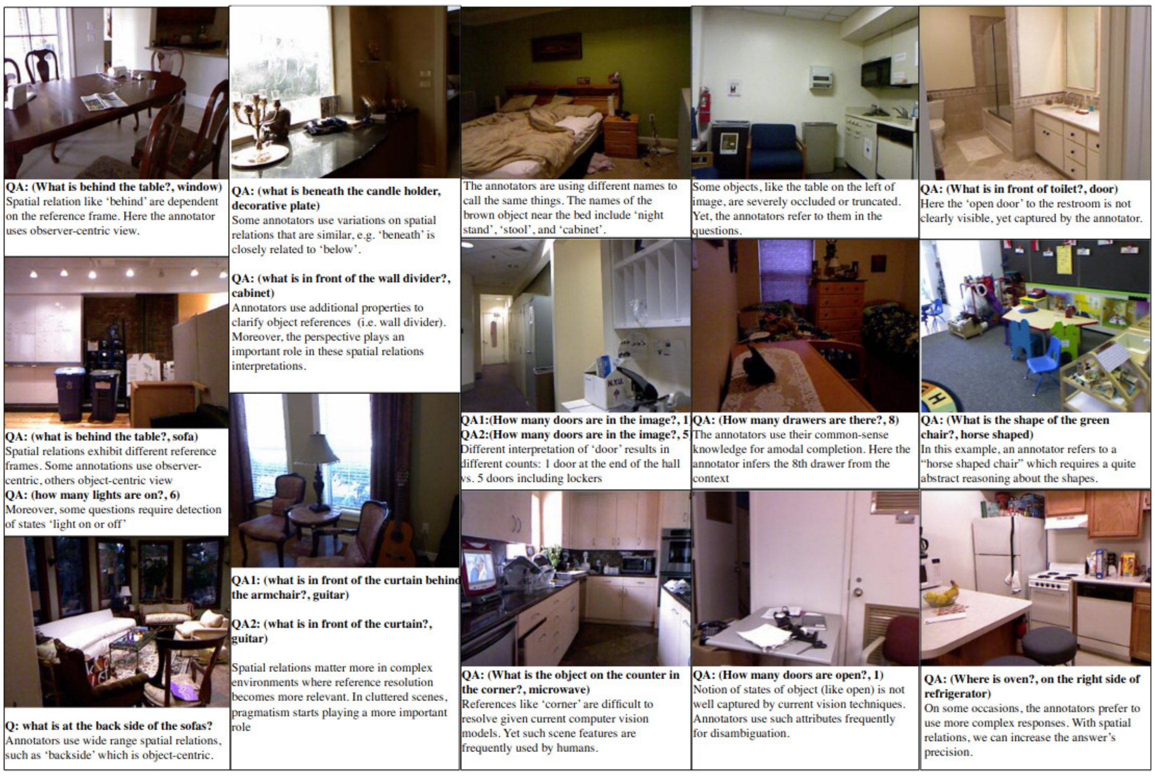
In this section we are going to list several VQA datasets that are considered remarkable, where some of the common characteristics between all of them are:

- Must contain a large amount of data (images + text).
- There must be a large variety of images and questions.
- It must support a fair evaluation form to validate the different VQA models.
- It must be minimally biased.

Among the large list of currently existing VQA datasets, the following datatsets are considered necessary to highlight:

### 2.5.1 DAQUAR [7]

The DAQUAR dataset was the first dataset and benchmark released for VQA. It was built with real-word images taken from NYU-Depth V2 dataset [69] and synthetic question-answer pairs more human question-answer pairs, where these latter had an introduced bias by people. Therefore, it is a dataset containing 1449 real-word images and 12468 question-answer pairs. Examples taken from [7] is shown in figure 2.6 below.



**QA: (What is behind the table?, window)**
Spatial relation like 'behind' are dependent on the reference frame. Here the annotator uses observer-centric view.

**QA: (what is behind the table?, sofa)**
Spatial relations exhibit different reference frames. Some annotations use observer-centric, others object-centric view
**QA: (how many lights are on?, 6)**
Moreover, some questions require detection of states 'light on or off'

**Q: what is at the back side of the sofas?**
Annotators use wide range spatial relations, such as 'backside' which is object-centric.

**QA: (what is beneath the candle holder, decorative plate)**
Some annotators use variations on spatial relations that are similar, e.g. 'beneath' is closely related to 'below'.

**QA: (what is in front of the wall divider?, cabinet)**
Annotators use additional properties to clarify object references (i.e. wall divider). Moreover, the perspective plays an important role in these spatial relations interpretations.

**QA1: (what is in front of the curtain behind the armchair?, guitar)**

**QA2: (what is in front of the curtain?, guitar)**

Spatial relations matter more in complex environments where reference resolution becomes more relevant. In cluttered scenes, pragmatism starts playing a more important role

The annotators are using different names to call the same things. The names of the brown object near the bed include 'night stand', 'stool', and 'cabinet'.

**QA1:(How many doors are in the image?, 1)**
**QA2:(How many doors are in the image?, 5)**
Different interpretation of 'door' results in different counts: 1 door at the end of the hall vs. 5 doors including lockers

**QA: (What is the object on the counter in the corner?, microwave)**
References like 'corner' are difficult to resolve given current computer vision models. Yet such scene features are frequently used by humans.

Some objects, like the table on the left of image, are severely occluded or truncated. Yet, the annotators refer to them in the questions.

**QA: (How many drawers are there?, 8)**
The annotators use their common-sense knowledge for amodal completion. Here the annotator infers the 8th drawer from the context

**QA: (How many doors are open?, 1)**
Notion of states of object (like open) is not well captured by current vision techniques. Annotators use such attributes frequently for disambiguation.

**QA: (What is in front of toilet?, door)**
Here the 'open door' to the restroom is not clearly visible, yet captured by the annotator.

**QA: (What is the shape of the green chair?, horse shaped)**
In this example, an annotator refers to a "horse shaped chair" which requires a quite abstract reasoning about the shapes.

**QA: (Where is oven?, on the right side of refrigerator)**
On some occasions, the annotators prefer to use more complex responses. With spatial relations, we can increase the answer's precision.

**Figure 2.6:** Examples of DAQUAR taken from [7]

### 2.5.2 VQA [8][4]

The VQA dataset is currently the most widely used dataset for VQA, becoming a standard reference in VQA. This dataset has two versions, a first version consisting of 204721 world-real images (MS-COCO [70]) and 50000 abstract scenes ([71]) with a 614163 and 150000 questions respectively where each question has 10 associated answers. One of the problems detected in this first version was the statistical bias and the language priors that allowed learning to answer the model without the need to understand the image ([4]). Therefore, to minimise these problems, a new version (VQA-v2.0) [4] was created, being a dataset with approximately 1.1 million (image, question) pairs with approximately 13 million associated answers in the 265016 MS-COCO and abstract images, where the model is forced to understand the information contained in the image by identifying similar images where the same question changes depending on the image (see figure 2.7 extracted from [4]).

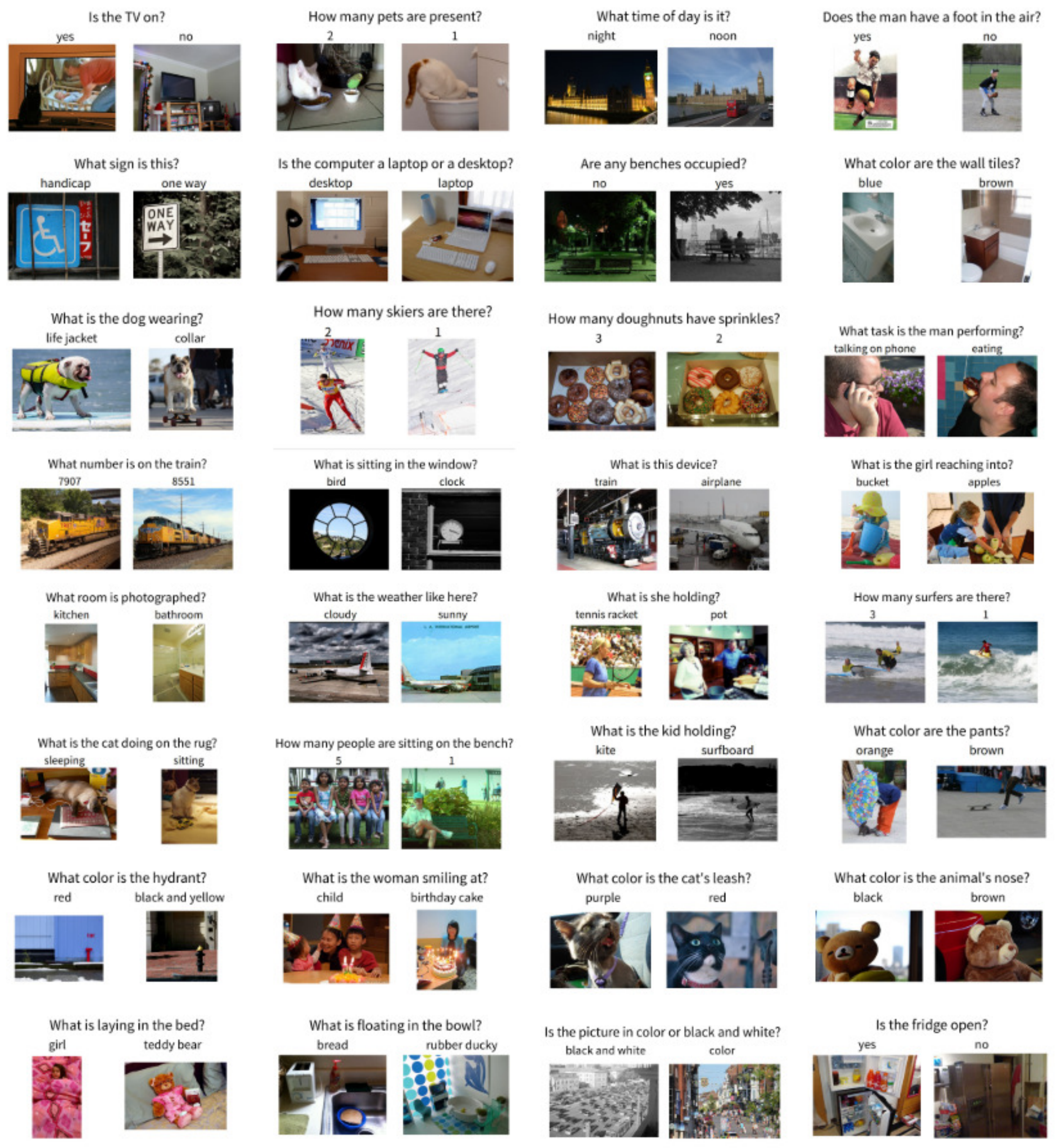

Is the TV on?
yes
no
How many pets are present?
2
1
What time of day is it?
night
noon
Does the man have a foot in the air?
yes
no
What sign is this?
handicap
one way
Is the computer a laptop or a desktop?
desktop
laptop
Are any benches occupied?
no
yes
What color are the wall tiles?
blue
brown
What is the dog wearing?
life jacket
collar
How many skiers are there?
2
1
How many doughnuts have sprinkles?
3
2
What task is the man performing?
talking on phone
eating
What number is on the train?
7907
8551
What is sitting in the window?
bird
clock
What is this device?
train
airplane
What is the girl reaching into?
bucket
apples
What room is photographed?
kitchen
bathroom
What is the weather like here?
cloudy
sunny
What is she holding?
tennis racket
pot
How many surfers are there?
3
1
What is the cat doing on the rug?
sleeping
sitting
How many people are sitting on the bench?
5
1
What is the kid holding?
kite
surfboard
What color are the pants?
orange
brown
What color is the hydrant?
red
black and yellow
What is the woman smiling at?
child
birthday cake
What color is the cat's leash?
purple
red
What color is the animal's nose?
black
brown
What is laying in the bed?
girl
teddy bear
What is floating in the bowl?
bread
rubber ducky
Is the picture in color or black and white?
black and white
color
Is the fridge open?
yes
no

**Figure 2.7:** Examples of VQA v2.0 taken from [4]

### 2.5.3 Visual Genome [9]

Visual Genome dataset is a dataset generated in 2017 containing more than 108077 images (obtenidas de YFCC100M ([72]) and COCO images [70]) in which each image has an average of 35 objects, 26 attributes and 21 pairwise relationships between objects, whose main components are: descriptions of regions, objects, attributes, relationships, region graphs, scene graphs and question-answer pairs. This makes it a dataset to investigate and improve multi-perspective understanding of images, from pixel-level information, such as objects, to relationships that require further inference, and even deeper cognitive tasks, such as question answering. Examples of this dataset extracted from [9] are shown in the figure 2.8 below.

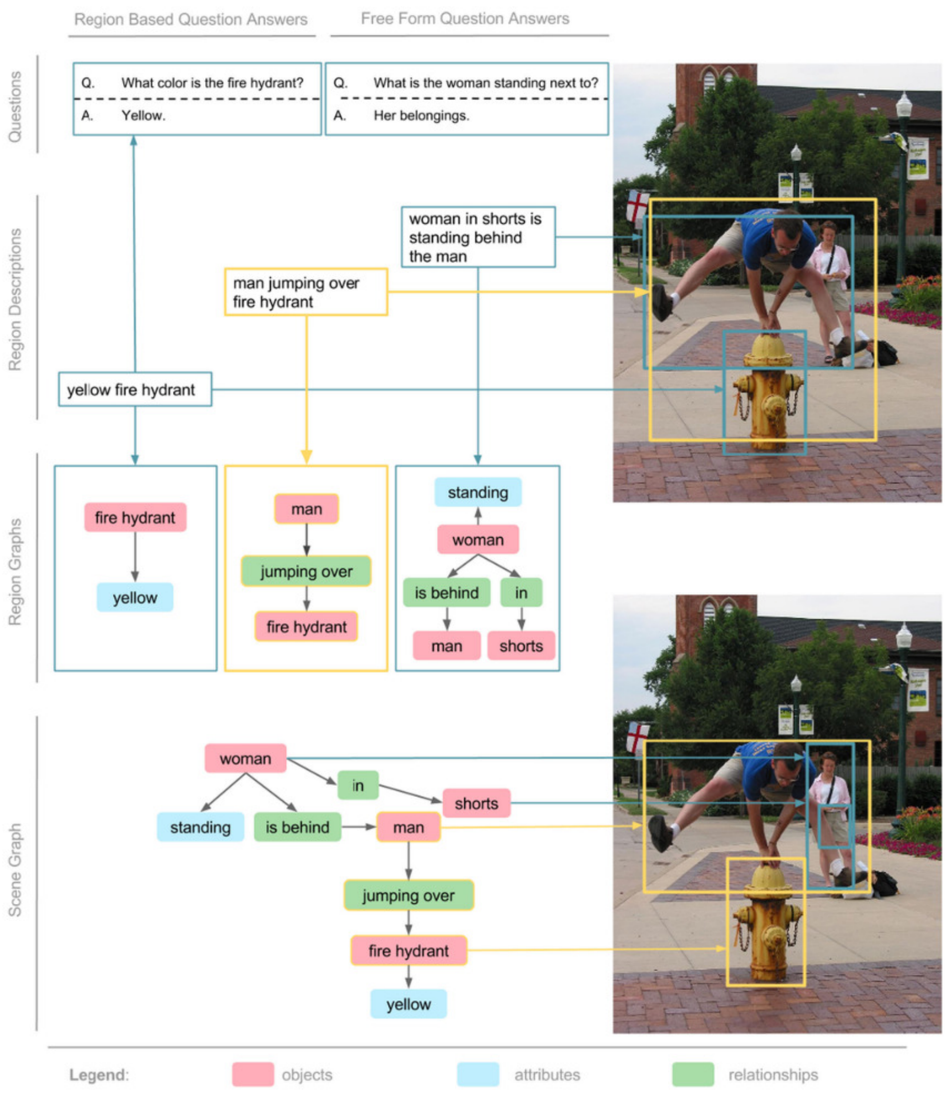


**Figure 2.8:** Examples of Visual Genome taken from [9]

### 2.5.4 Other datasets

After briefly explaining the main VQA datasets of real images, we proceed to list other datasets that are considered noteworthy and that will allow further progress in the research of multimodal models related to VQA: **Flickr30k Entities** [73] (dataset of region-to-phrase correspondences for image description), **Visual7W** [53] (subset of the Visual Genome that contains additional annotations), **SHAPE** [74] (synthetic dataset that contains of complex questions about simple arrangements of col-ored shapes), **CLEVR** [75] (diagnostic dataset to study the ability of VQA systems to perform visual reasoning), **TextVQA** [76] (dataset that require reasoning about the text in images to answer question about them), **TextCaps** [77] (dataset that require read and reason about text in images to generate captions about them), **DocVQA** [78] (dataset with the aim to inspire a "purpose-driven" approach in document image analysis and recognition research.), **OK-VQA** [79] (dataset that requires methods which can draw upon outside knowledge to answer questions), **VQA-Med** [80] (dataset for learning to answer medical questions based on the visual content of radiological images), etc.

## 2.6 Performance evaluation

Given the two types of questions in VQA, multiple-choice (one correct answer per question) and open-ended (multiple correct answers per question), evaluation is not trivial in the latter case. One of the solutions usually adopted in VQA for the latter case is to restrict the answers to a few words or to select an answer from a closed set of (more frequent) answers.

Thus, accuracy ($\frac{\#correctly\ answered}{\#total\ question}$) may be valid as a metric for the multiple-choice setting, but it becomes a too restrictive metric for the open-ended setting, so alternative metrics have been proposed:

**WUPS** [81]: a smoothed accuracy measure ranging from 0 to 1 that was proposed in [7] where to avoid semantically distant words having a high WUPS a threshold was proposed where if the WUPS measure was less than this threshold, it will be scaled down by a factor. However, high scores are still produced between lexically related responses with different semantic meaning, and it is a metric that only works with single word responses, i.e. it does not work with sentence anwers. Its equation (2.1) is shown below:

$$WUPS = \frac{1}{N}\sum_{i=1}^{N} \min\left\{\prod_{a\epsilon A'}\left(\max_{t\epsilon T'} WUP(a,t), \prod_{t\epsilon T'} \max_{a\epsilon A'} WUP(a,t)\right\} \cdot 100 \tag{2.1}$$

where $N$ is total number of questions, $A$ is set of predicted anwers, $T$ is set of ground truth anwers and $WUP(a,b)$ returns the positions of words $a$ and $b$ in the taxonomy relative to the position of Least Common Subsumer $(a,b)$.

**Consensus measure**: based on having multiple true answers obtained independently by the scorers for each question. There are two types of consensus measure:

- Medium consensus, the final score is weighted to prefer the most popular answer provided by the scorers.
- Minimum consensus, the answer must be in agreement with at least one annotator.

The latter measure was used in the VQA dataset [8] and is currently used in the VQA Challenge [10], which is shown in the equation 2.2:

$$Accuracy_{vqa} = \min\left(\frac{\#humans\ that\ said\ ans}{3}, 1\right)\left( \tag{2.2}$$

The problems with this type of measure are that it allows several correct answers for the same question and the difficulty in getting the large number of answers needed.

**MPT** [82]: Metric proposed with the objective of minimising the problem of skewed distribution of question types, which consists of the calculated arithmetic or harmonic mean accuracy per question type. Due to the bias that is also found in the distribution of answers within each question type, exist these normalized metrics also. If there is big differences between unnormalized and normalized scores indicates that the VQA model don't generalize well for rarer answers. The algebraic expressions (2.3 and 2.4) for the MPT calculation is shown below:

$$MPT_{arithmetic} = \frac{\sum_{t=1}^{T} A_t}{T} \tag{2.3}$$

$$MPT_{harmonic} = \frac{T}{\sum_{t=1}^{T} A_t^{-1}} \tag{2.4}$$

where $T$ is total number of question types and $A_t$ is accuracy over question type $t$.

**BLEU** [83]: metric for automatic evaluation of machine translation that ranges from 0 to 1, that build on matches of n-grams between the predicted answer and ground truth label. Therefore is computed as the geometric mean geometric mean of the test corpus' modified precision scores and then multiply the result by an exponential brevity penalty factor [83] such as shown in the equation (2.5) below:

$$BLEU = BP \cdot \left( \sum_{n=1}^{N} W_n \; logP_n \right) \tag{2.5}$$

where $BP$ is Brevity Penalty [83], $W_n$ are positive weights that summing to one and $P_n$ is Precision score of entire corpus [83].

**METEOR** [84]: metric for automatic evaluation of machine translation that compute the harmonic mean of the precision (fraction of the hypothesis which matches the reference) and recall (the fraction of the reference which is contained in the hypothesis) calculated based on exact, stemmed, synonyms and paraphrase matches. The algebraic expression for the METEOR calculation is shown below:

$$METEOR = (1 - Pen) * F_{mean} \tag{2.6}$$

where $Pen$ is a fragmentation penalty and $F_{mean}$ is the parameterized harmonic mean (read [84]).

**Manual evaluation**: based on people's subjective judgement where it works well but is very costly due to the large amount of resources and time it requires.

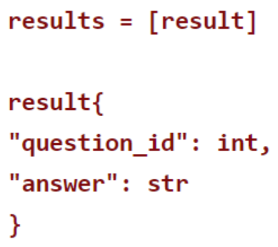
## 2.7 VQA Challenge [10]

VQA challenge is a competition held every year since 2016, which consists of a task of correctly answering question-image pairs, where the results are displayed at the VQA Challenge Workshop at The Conference on Computer Vision and Pattern Recognition (CVPR).

The data used by the competition to evaluate the results is the VQA v2.0 test data, which is divided into test-dev, test-standard, test-challenge and test-reserve, to limit overfitting and give researchers more flexibility to test their system:

- **Test-dev split**: used for debugging and validation experiments (allows a maximum of 10 submissions per day). These results are not public
- **Test-dev split**: this is the default test data for the VQA competition. The results shown in the articles must be on the test-standard. This data will be made public each time it is submitted, updating the public leaderboard, allowing to know the progress in the VQA challenge.
- **Test-reserve split**: is used to protect against possible overfitting. If there are substantial differences between a method's scores in the test-standard and the test-reserve, an alarm signal will be raised and further investigation will be requested. These results are not public.
- **Test-challenge split**: is used to determine the winners of the challenge.

To participate in the VQA challenge it is necessary to create an EvalAI account [18], where the VQA challenge is available, from where it is allowed to submit a json file in the correct format with the results obtained in the split test of the VQA dataset. The format of this results file is shown below in figure 2.9:

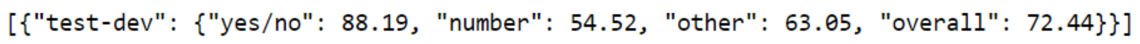


```
results = [result]

result{
"question_id": int,
"answer": str
}
```

**Figure 2.9:** Correct json format taken from [17]

Once the json file has been submited in the correct format, after waiting a few minutes and if everything has worked correctly, the submission will appear in 'Finished' status, where the results will be available for viewing with the different accuracies for the different types of answers ('yes/no', 'number', 'other'), as well as an overall accuracy, where this metric is calculated as $Accuracy_{vqa} = \min\left(\frac{\#humans\ that\ said\ ans}{3}, 1\right)$. Below, the figure 2.10 is an example of what the output results would look like:



```
[{"test-dev": {"yes/no": 88.19, "number": 54.52, "other": 63.05, "overall": 72.44}}]
```

**Figure 2.10:** Evaluation json format taken from [18]

Thanks to this competition, it is possible to visualise the evolution and progress of the VQA Artifial Intelligence task in recent years, where every year at the VQA workshop held at the CVPR an analysis of the results obtained in the VQA challenge is shown, explaining the improvements obtained with respect to previous years and possible deficiencies of the models presented. One of the graphs presented in CVPR 2020 that we wanted to highlight in this work is the progress in VQA, which is shown below in figure 2.11, where it can be seen that from 2015 (Accuracy $\sim$ 55%) to 2020 (Accuracy $\sim$ 76%) there has been an improvement of $\sim$ 21% in the Acurracy in the VQA task.

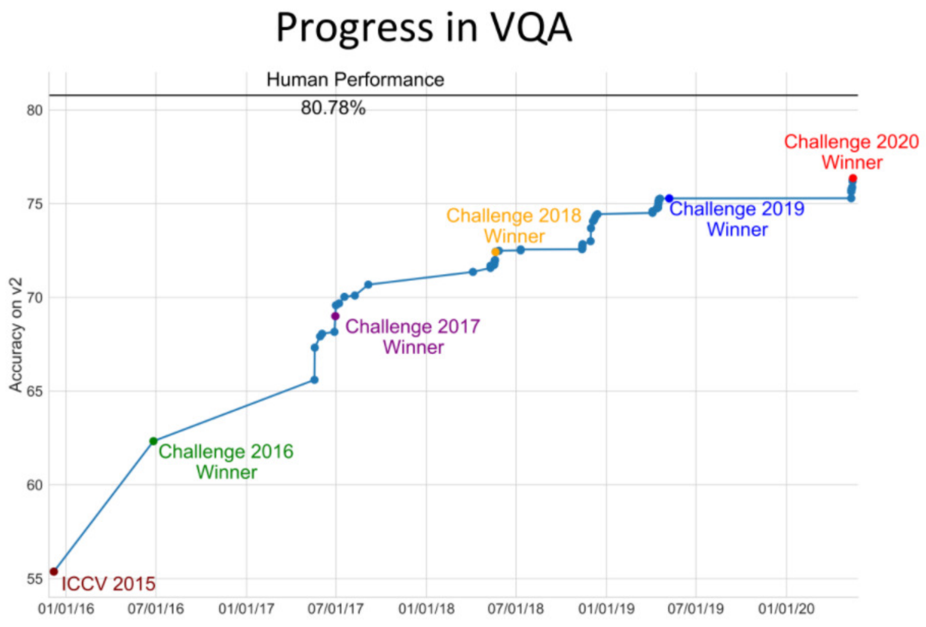


**Figure 2.11:** VQA progress taken from [19]

# 3 Datasets

This chapter explains the main dataset of VQA [4] used in this work, and then briefly explains the dataset that is currently being worked on in order to try to build a valid VQA dataset with respect to e-learning based on the existing edBB dataset [6].

## 3.1 VQA v2.0 [4]

For the first experiments of this work we use the most widely used dataset [4] for the task of Visual Question Answering (VQA), which will allow us to try to replicate the state of the art of the latest state-of-the-art models with the best results obtained in recent years in the VQA challenge [10].

It is a second version of the VQA v1.0 dataset [8], where it has been evolved in an effort to minimize the linguistic bias by forcing the model to focus on visual information in order to improve the understanding of the image. For this purpose, "complementary" images were incorporated, which consist of images similar to those already existing in the VQA v1.0 dataset with the peculiarity that for the same question they have different answers. For instance, given an triplet (Image (I), Question (Q), Answer (A)) from the VQA v1.0 version, it is added a similar image (I') of Image (I), where the answer for the same Question (Q) in both images it has different answers (A for I and A' for I'). This example is illustrated in the next figure 3.1 with real cases:

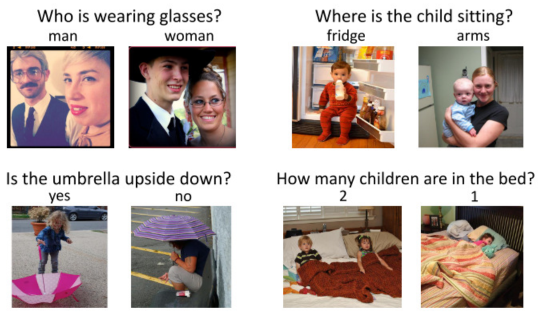

Who is wearing glasses?
man
woman
Where is the child sitting?
fridge
arms
Is the umbrella upside down?
yes
no
How many children are in the bed?
2
1

**Figure 3.1:** Examples of complementary images taken from [4]

The process of building this new version of the VQA dataset is composed of several stages [4], where the first one consisted in retrieving complementary images with respect to the images already existing in the first version of the VQA dataset. To do so, they created an interface to collect these complementary images on Amazon Mechanical Turk (AMT). In this interface, AMT workers were shown a list of the 24 most similar images of image I, where they had to choose an image I' from the list in which the question Q (corresponding to the triplet (I, Q, A) of image I) made sense and whose answer A' was different from A. And the second stage consisted of data annotation, where 10 new answers were collected for each question in the complementary images.

Lastly, after finishing with the collection process all the new triplets (I', Q', A') of the complementary images, left a new version of the VQA dataset where approximately 195K complementary images had been collected for the training set, 93K complementary images for the val set and 191K complementary images for the test set, leaving a balanced dataset with approximately more than 443K train, 214K val and 453K test question-image pairs, with an average of 10 answers per question. The large number of test data is due to the fact that the test is divided into 4 splits (test-dev, test-standard, test-challenge and test-reserve) as explained in detail in section 2.7.

Therefore, the final dataset consists of MSCOCO images of different sizes in jpg format, input questions about the images stored in json format following the structure shown in figure 3.2 and the labelling of the data (annotations) in json format with the structure shown in figure 3.3.

```
{
"info" : info,
"task_type" : str,
"data_type": str,
"data_subtype": str,
"questions" : [question],
"license" : license
}

info {
"year" : int,
"version" : str,
"description" : str,
"contributor" : str,
"url" : str,
"date_created" : datetime
}

license{
"name" : str,
"url" : str
}

question{
"question_id" : int,
"image_id" : int,
"question" : str
}
```

**Figure 3.2:** Input questions format taken from [20]

```
{
"info" : info,
"data_type": str,
"data_subtype": str,
"annotations" : [annotation],
"license" : license
}

info {
"year" : int,
"version" : str,
"description" : str,
"contributor" : str,
"url" : str,
"date_created" : datetime
}

license{
"name" : str,
"url" : str
}

annotation{
"question_id" : int,
"image_id" : int,
"question_type" : str,
"answer_type" : str,
"answers" : [answer],
"multiple_choice_answer" : str
}

answer{
"answer_id" : int,
"answer" : str,
"answer_confidence": str
}
```

**Figure 3.3:** Annotations format taken from [20]

## 3.2 edBB Dataset [6]

With the idea of investigating the application of the VQA models of this work in a new database, and given the great expansion of e-learning platforms (virtual education) in recent years driven by the situation of non-presence that has meant the COVID-19 , it has been decided to study VQA on a dataset with images of students recorded during e-learning sessions. To this end, work has begun on the construction of a new database with the help of the BiDA Lab of the Universidad Autonoma de Madrid, since they already have the edBB database [6], which consists of recordings of 20 students in controlled laboratory conditions during a session where several measurements are taken as shown in the following figure 3.4:

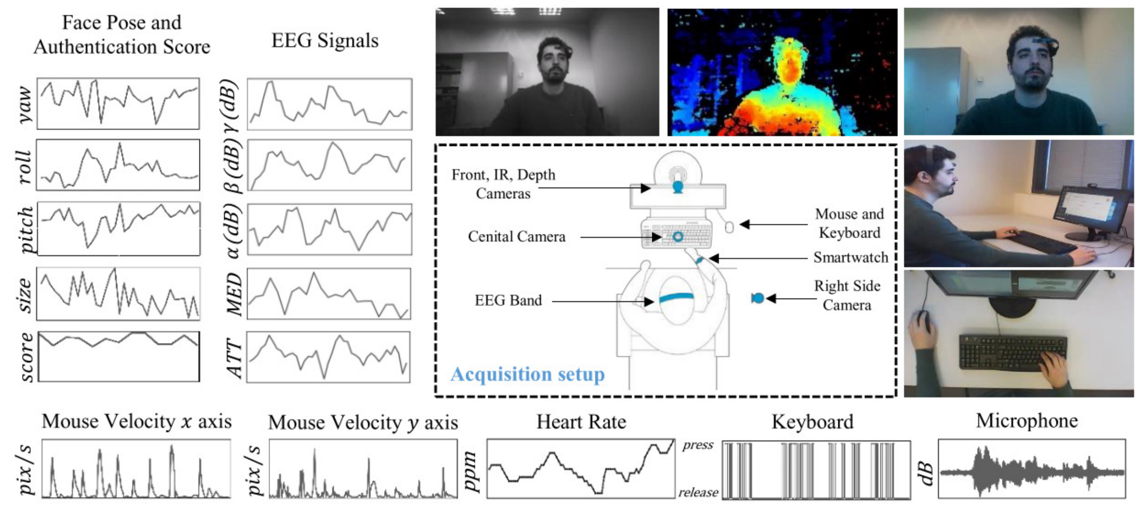


**Figure 3.4:** Example of the information captured with the edBB platform taken from [6]

Since the available data are in video format, and the models used in this work are based on images and text, it is necessary to transform the videos into images on which questions will be asked. Therefore, for the construction of the new e-learning database, we are currently working on the extraction of images from the videos as diverse as possible, trying to minimise the possible biases that may occur when asking the questions. That is to say, we are trying to choose images in such a way that the same question has different answers depending on the image, since otherwise the system could model only with the text without taking into account the image.

With regard to the questions to be generated, with the aim of trying to obtain as many formulations of the same question as possible, it has been established that each person involved in the creation of the new database will make a list of simple questions so that in the end a final list with all the unique questions will be created in the correct format.

# 4 VQA models

This chapter explains the two VQA models used in this work, where the first one (MoViE+MCAN) was chosen because it was the winning model of the VQA Challenge 2020 and the second one (LXMERT) was chosen as an alternative to the first one due to the problems that arose during the experimentation of the first one when trying to replicate results.

## 4.1 MoViE+MCAN [1] [2]

This model is based on the MCAN model [11] (winner of the VQA challenge 2019) with two modifications: image grid features instead of region features [2] and incorporation of the MoVie method [1].

### 4.1.1 MCAN model [11]

Inspired by the Transformers models [28], this model consists of Modular Co-attention (MCA) layers cascaded in depth, where each MCA layer is a modular composition of two basic attention units, the self-attention (SA) unit to capture the dense intra-modal interactions (word2words or region2region) and the guided-attention (GA) unit to capture the dense inter-modal interactions (word2region).

These units are based on the scaled dot-product attention [28], whose inputs consist of queries and keys and values where the attention weights ($\alpha$) are obtained from the query (q) and keys (K), which allow to compute the attended feature (f) by weighted summation over all values (V):

$$f = A(q, K, V) = \text{softmax}(\frac{qK^T}{\sqrt{d}})V = \alpha(q, K)V$$

Where to improve the representability of the attended features, multi-head attention (in parallel) is introduced, where each head corresponds to a scaled dot-product attention:

$$f = MA(q, K, V) = [head_1, head_2, ..., head_h]W^o$$
$$head_j = A(qW_j^Q, KW_j^K, VW_j^V)$$

Therefore, the self-attention unit (see figure 4.1) is composed of a multi-head attention layer and a point-wise feed-forward layer (2 fully connected layers with ReLU activation and dropout (FC-ReLU-Dropout-FC)) that take only one group of inputs corresponding to those of a modality. In addition a residual connection and normalization layer is applied to the two outputs. And the guide-attention unit (see figure 4.2) is composed of the same structure with the difference that it takes two groups of inputs corresponding to two modalities for one modality to guide the attention learning of the other modality.

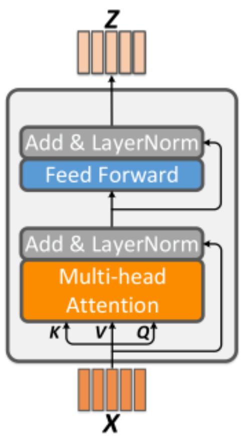


**Figure 4.1:** Self-Attention (SA) unit taken from [11]

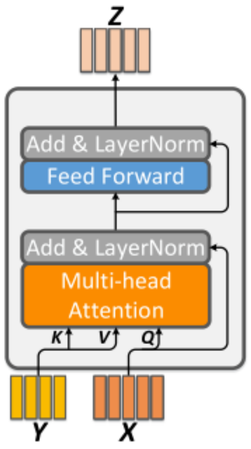


**Figure 4.2:** Guided-Attention (GA) unit taken from [11]

Among the three variants presented in [11], the model used in this work uses the one that obtained the best results in the [11] experiments, i.e. SA(Y)-SGA(X,Y) (see figure 4.3) with an encoder-decoder strategy (see figure 4.4), where the output and input of each MCA layer has the same number of features allowing to perform a Deep Co-Attention Learnimg.

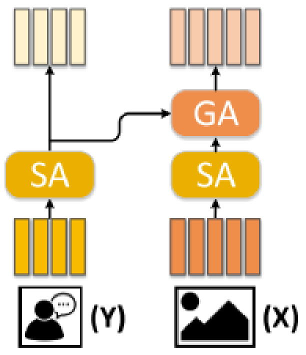


**Figure 4.3:** MCA variant of SA-SGA(X,Y) taken from [11].(Y) and (X) denote the question and image features respectively.

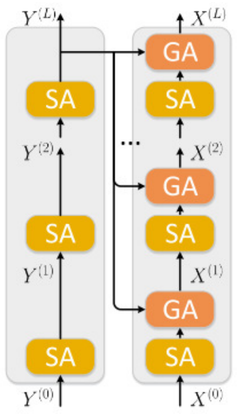


**Figure 4.4:** taken from [11]

With that in mind, the architecture of the 2019 winning MCAN model (see figure 4.5), which is then modified by adding the MoViE module and changing the region of interest (RoI) features by grid features to achieve the winning model of the 2020 VQA Challenge, would consist of:

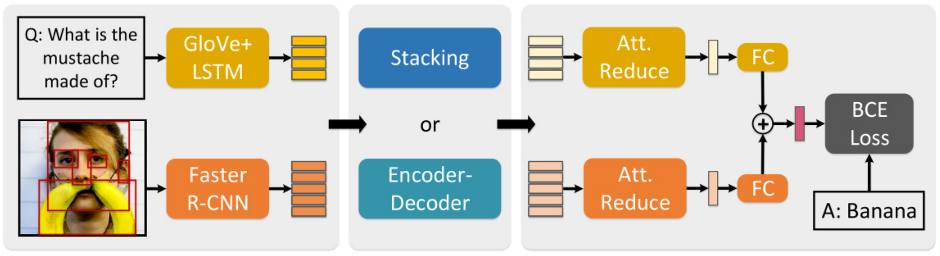


**Figure 4.5:** Modular Co-Attetion Networks architecture taken from [11]

- Inputs: Within the inputs two groups are distinguished, the input image which in the MCAN model of 2019 correspond to a set of region visual features extracted by bottom-up attention [36] (Faster R-CNN) corresponding to a dynamic number of detected objects ($m\epsilon[10, 100]$). However, these region features were transformed to grid features [2] in the winning 2020 MoViE+MCAN model (model used in this work), explaining in the following section the advantages and modifications made in the Faster R-CNN. The other group of features are the question features, which are first tokenised and trimmed to a maximum of 14 words, where each word is transformed to a vector using 300-D GloVe word embedding. The word embeddings are then passed through an LSTM layer.
- Deep Co-Attention Learning: With the inputs in the format explained in the above section, deep co-attetntion learning is carried out by passing the inputs features through a deep co-attention model consisting of 6 MCA layers ($MCA^{(1)}, MCA^{(2)}, \cdot, MCA^{(6)}$) of the type SA(Y)-SGA(X,Y), where the inputs feature ($X, Y$) of each layer are the outputs of the previous layer, i.e. they are fed in a recursive way:

$$[X^{(l)}, Y^{(l)}] = MCA^{(l)}([X^{(l-1)}, Y^{(l-1)}])$$

  Each SA and GA layer, consisting of 8-head multi-heads with a hidden dimensionality of 1024, is staked following an encoder-decoder strategy, where the input feature $Y^{(l)}$ of each GA unit is the question feature of the last MCA layer ($Y^{(6)}$) as shown in figure 4.4.
- Multi-modal Fusion and Output Classifier: Once Deep Co-attention Learning has been carried out, the output of the image features ($X^{(6)}$) and of the question features $Y^{(6)}$ already contain relevant information about the attention of question words and image features, Therefore, an attentional reduction model is carried out built with two multi-layer perceptron (MLP) formed by two Fully Connected layers with ReLU activation and dropout (FC-ReLU-Dropout-FC) for both $X^{(6)}$ and $Y^{(6)}$, obtaining their attended features ($\tilde{x}$ and $\tilde{y}$). After obtaining the attended features, a linear fusion function is carried out:

$$z = \text{LayerNorm}(W_x^T \tilde{x} + W_y^T \tilde{y})$$

  Where the fused feature ($z$) is projected to a vector s of dimension 3129 (most frequent answers) to which a sigmoid function is applied.

The modifications carried out on the MCAN model to transform it into the MoViE+MCAN model, winner of the 2020 VQA Challenge, are explained below:

### 4.1.2 In defense of Grid Features [2]

Among the modifications carried out in the winning model of the 2019 VQA Challenge (MCAN) to transform it into the winning model was to change the inputs with respect to the images, changing from region of interest features to grid features where it was empirically demonstrated in [2] that similar results were obtained.

The fact of changing to grid features in spite of achieving similar results was due to the advantages provided by these features such as the simplicity of the VQA model (avoiding region-related computations), speed increase without loss of Accuracy, high recall when dealing with the whole image instead of sparse region of interest, giving the possibility of implementing an end2end model and earlier fusion for reasoning tasks such as counting.

To take the step from region features to grid features, it was necessary to change the structure of the Faster R-CNN (see figure 4.6), going from a 14x14 RoIPool to a 1x1 RoIPool where it forced to move the $C_5$ block to the backbone of the ResNet to optimize the performance of $C_5$, where the stride-2 layers are replace with stride-1 layers and a factor 2 dilation was carried out to avoid loss of resolution during training. Therefore, as shown in figure 4.6 the entire ResNet backbone up to $C_5$ is used for the shared feature computation and for the region-level computation 2 fully connected layers are placed on the top, which accept input vectors.

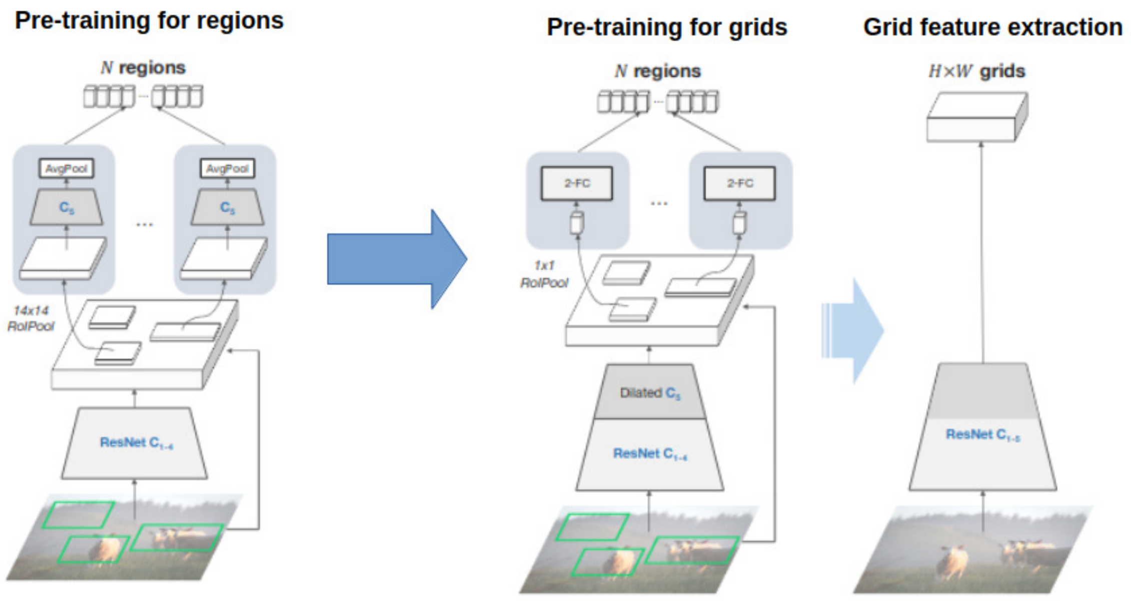


**Figure 4.6:** From regions back to grids taken from [2]

Finally, as seen on the right of the 4.6 image, the computations with respect to regions are removed during feature extraction, i.e., the grid feature extractor is kept untouched during inference.

### 4.1.3 MoVie: Modulated conVolutional bottlenecks [1]

The other modification carried out on the MCAN model to achieve the MoViE+MCAN model, winner of the 2020 VQA Challenge, was the incorporation of the MoViE module (shorter for Modulated conVolutional bottlenecks) to improve the counting task, based on the fact that the local fusion scheme of modulated convolutions is preferable to other fusion schemes given the translation-equivariant nature of the counting task, and that the sparsely image region starts at a disadvantage with respect to the convolutional features that cover all image locations. The idea behind this module is to densely apply the modulation of the query representation over all locations.

The architecture of this new module is shown in figure 4.7, where it is shown:

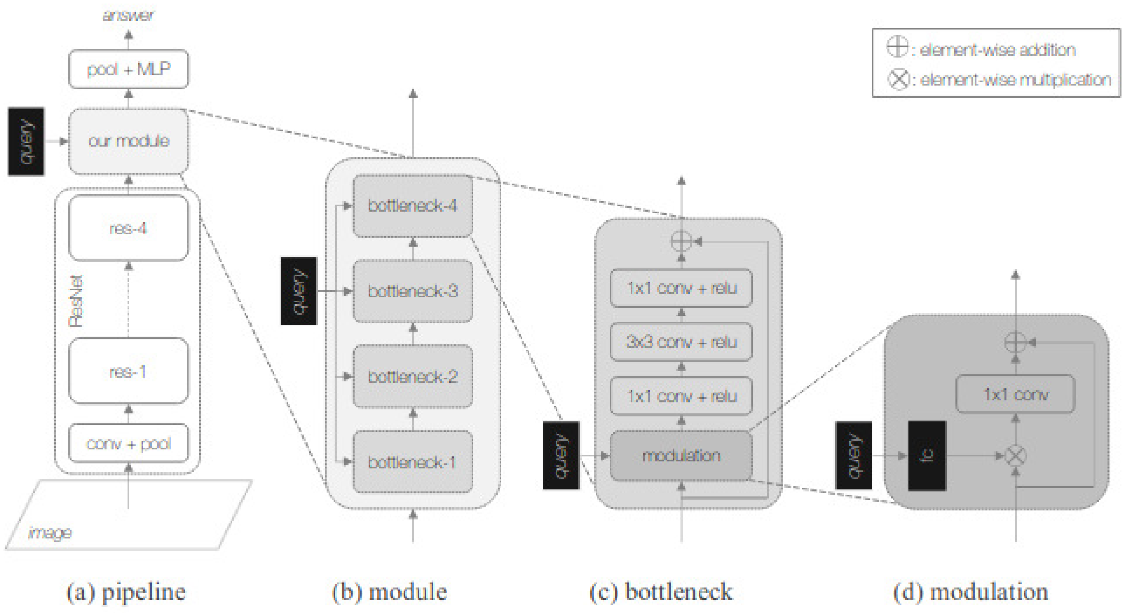


**Figure 4.7:** Overview of MoViE Architecture taken from [1]

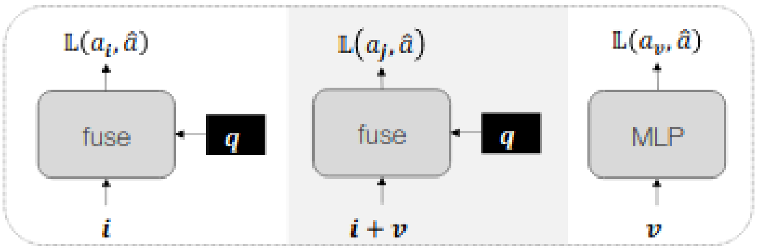
- Overall pipeline: it is observed that the MoViE module is applied to the output convolutional features of a CNN (e.g. ResNet), where finally an average-pooling and a two-layer MLP matcher are applied to predict the response.
- Module: consists of four modulated convolutional bottlenecks, where each bottleneck receives the extra input from the query to modulate the feature map.
- Bottleneck: very similar to the ResNet bottleneck [26], with the difference that before the first 1x1 convolutional layer, a modulation block is inserted.
- Modulation block: this modulation block is defined as:

$$\bar{v}_{MoViE} = v \oplus W^T(v \otimes \Delta\gamma)$$

  where $v$ is a feature vector, $W$ is a learneable weigth matrix and $\Delta\gamma$ serves to scale the residual connection of the feature vector, which is conditioned on the query representation $(q)$.

Note that the fusion is not performed between query and global pooled vector, i.e., all interactions between query and image occur in modulated bottlenecks locally.

Finally, it is worth mentioning that the incorporation of this module in VQA models (e.g. MCAN) has the peculiarity that 3 branches are used for training (see figure 4.8), where each branch is assigned the same loss weight:



$\mathbb{L}(a_i, \hat{a})$
fuse
$q$
$i$
$\mathbb{L}(a_j, \hat{a})$
fuse
$q$
$i + v$
$\mathbb{L}(a_v, \hat{a})$
MLP
$v$

**Figure 4.8:** MoViE as counting module for VQA taken from [1]

- Original branch consisting of the source VQA model, i.e., from an image representation ($i$) and a question representation to which a fusion scheme is applied to produce the final answer.
- Joint branch consisting of adding the pooled feature ($v$) obtained from MoViE to $i$ (image representation) and applying fusion scheme to predict the response.
- Branch that consists of training a Movie nomral with pooled feature ($v$) and MLP.

However, at the time of inference only the joint branch is used, which avoids losing inference speed.

## 4.2 LXMERT: Learning Cross-Modality Encoder Representations from Transformers [3]

This pre-trained cross-modality language and vision framework consists of a large-scale transformation model consisting of three encoders (object relationship encoder, language encoder and cross-modality encoder) pre-trained on five tasks with a large number of image-sentence pairs to try to learn intra-modality and inter-modality interactions, allowing the model the ability to infer masked features from the modalities themselves (as unimodal models) or from other modalities. The five pre-training tasks (see figure 4.9) that incorporate generalisation into the model after which fine-tunnig of the pre-trained parameters will be used for the VQA task are:

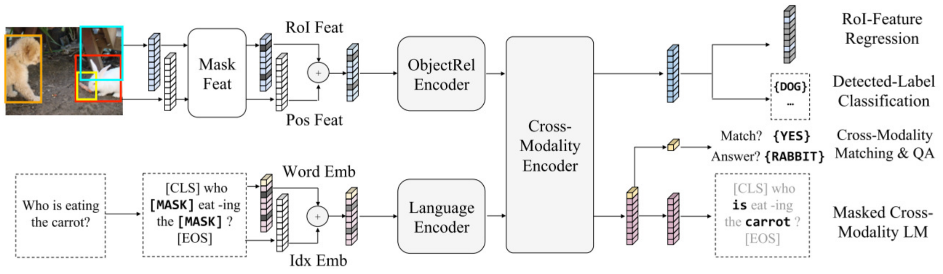


**Figure 4.9:** Pre-training in LXMERT taken from [3]

- Language Task of **Masked Cross-Modality Language Model**: this task consists of randomly masking words with a probability of 15% for the model to predict these masked words. Since the model has inputs from two modalities, vision and language, it can infer the masked words from the rest of the unmasked words or from the features coming from the vision modality, allowing to clarify the prediction of the words. For instance, if the sentence is "How many [mask] are there on the table?" it would be difficult to predict the masked word "books" from the context of the other words, but looking at the image and seeing two books on the table would make it easier to predict the masked word "books".
- Vision Task of **Masked Object Prediction via RoI-Feature Regression**: this task consists of randomly masking objects (i.e., masking RoI Features with zeros) with a probability of 15% so that the model predicts the object RoI feature $f_j$. Since the model has inputs from two modalities, vision and language, it can infer objects from both modalities, allowing it to learn intra-modal and inter-modal relationships. This task regresses object RoI feature fj with L2 loss.
- Vision Task of **Masked Object Prediction via Detected-Label Classification**: this task consists of randomly masking objects (i.e., masking RoI Features with zeros) with a probability of 15% so that the model predicts the labels of the masked objects. Since the model has inputs from two modalities, vision and language, it can infer objects from both modalities, allowing it to learn intra-modal and inter-modal relationships. This task allows learning to infer the labels of masked objects with cross-entropy loss, where the labels from the output of the Faster RCNN are used.
- Cross-Modality Task of **Cross-Modality Matching**: This task consists of randomly replacing with a 50% probability a sentence corresponding to an image with another missmatched sentence so that the model can learn to predict whether an image and a sentence match each other (similar to the 'Next Sentence Prediction' task of BERT [15]).
- Cross-Modality Task of **Image Question Answering**: This task consists of predicting the answer to a question when the image and the question are correctly matched, i.e. the Cross-Modality Matching task replacement has not been applied.

Regarding the architecture of the model (see figure 4.10), it is built with self-attention and cross-attention layers based on the recent work of transformers [28], where the inputs of this model are an image and a question, which are pre-processed to be projected in the right space so that the model can learn correctly:

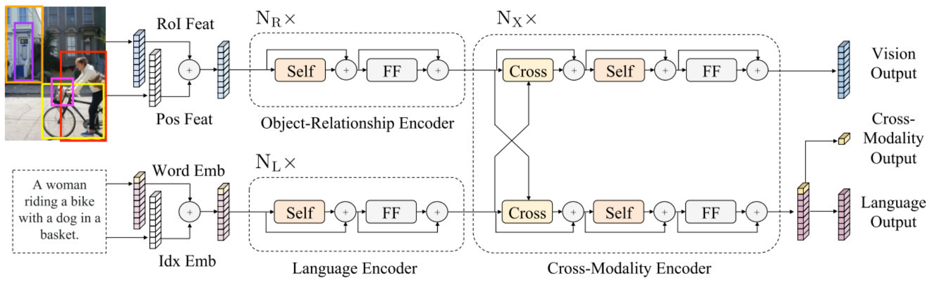


**Figure 4.10:** Architecture of LXMERT taken from [3]

- **Input Representations**: a distinction is made between those corresponding to the visual modality and those corresponding to the textual modality.
  - Within the textual modality, Word-Level Sentence Embeddings are used, which consists of representing each sentence as a sequence of words with length $n$ that have been obtained by means of the same WordPiece tokenizer used in BERT [15]. Where once the words $w_i$ and their absolute position indices in the $i$ sentences are obtained, these are projected by embedding sub-layers, allowing to learn the index-aware word embedding $h_i$ as shown in figure 4.10:

$$\begin{aligned}\hat{w}_i &= \text{WordEmbed}(w_i)\\ \hat{u}_i &= \text{IdxEmbed}(i)\\ h_i &= \text{LayerNorm}(\hat{w}_i + \hat{u}_i)\end{aligned}$$

  - Within the visual modality, Object-Level Image Embeddings is used, which consists of representing each image as $m$ objects (obtained by Faster RCNNN from [36]) where each of them is represented by its position feature (bounding box coordinates) $p_j$ and its 2048-dimensional region-of-interest (RoI) feature $f_j$, using the sum of the outputs of the two fully-connected layers ($\hat{p}_j$ and $\hat{f}_j$) to get the model to learn a position-aware embedding $v_j$ as shown in figure 4.10:

$$\begin{aligned}\hat{f}_j &= \text{LayerNorm}(W_F f_j + b_F)\\ \hat{p}_j &= \text{LayerNorm}(W_P p_j + b_P)\\ v_j &= (\hat{f}_j + \hat{p}_j)/2\end{aligned}$$

- **Encoders**:The LXMERT model is built by three encoders, the language encoder, the object-relationship encoder and the cross-modality encoder, which are based on self-attention layers and cross-attention layers, so before going into detail about the encoders of the model, we will briefly explain how the attention layers work.
  Attention layers aim to retrieve information from a set of context vectors ($y_j$) related to a query vector ($X$). The attention mechanism is based on the scaled dot product, where first the calculation of the matching score ($a_i$) between $x$ and each $y_j$ is carried out by normalising it by softmax. Where finally the output of an attention layer would be the weighted sum of the context vectors and the normalised score:

$$\begin{aligned} a_j &= \text{score}(x, y_j) \\ \alpha_j &= \text{Softmax}(a_j) \\ \text{Att}_{X \to Y}(x, \{y_j\}) &= \sum_j \alpha_j y_j \end{aligned}$$

  A layer is called self-attention when the query vector ($x$) and the set of context vectors ($\{y_i\}$) are the same. As with Transformer models [28] in LXMERT multi-head attention is used in order to obtain better representations.
  Having explained how the attention mechanisms work, we proceed to explain the types of encoders implemented in LXMERT:

  – Single-Modality Encoders: As shown in figure 4.10, after performing the embeddings (Word-Level Sentence Embeddings and Object-Level Image Embeddings), two encoders are applied, where each of them is oriented for textual modality (language encoder) and visual modality (object-relationship encoder) respectively. The model used in this work is formed by $N_L = 9$ stacked language encoders and $N_R = 5$ object-relationship encoders, where each of these encoders is composed of a self-attention sub-layer and a feed-forward sub-layer composed of two fully-connected sub-layers. As seen in figure 4.10, a residual connection and normalization layer is added after each sub-layer (represented by $\oplus$).
  – Cross-Modality Encoder: As shown in figure 4.10, after of $N_L$ language encoders and $N_R$ object-relationship encoders, one additional encoder is applied, where it is oriented to learn joint cross-modality representations. The model used in this work is formed by $N_X = 5$ staked cross-modality encoders, where each of these encoders is composed of two self-attention sub-layers, one bi-directional cross-attention sub-layer and two feed-forward sub-layers. As can be seen in the figure 4.10 the order of the sub-layers in each of the cross-modality encoders is, first the cross-attention sub-layers are applied (one from language to vision and one from vision to language), where the query and context vectors correspond to the outputs of the (k-1)-th layer:

$$\begin{aligned} \hat{h}_i^k &= \text{CrossAtt}_{L \to R}(h_i^{k-1}, \{v_1^{k-1}, \ldots, v_m^{k-1}\}) \\ \hat{v}_j^k &= \text{CrossAtt}_{R \to L}(v_j^{k-1}, \{h_1^{k-1}, \ldots, h_n^{k-1}\}) \end{aligned}$$

    After applying the cross-attention sub-layers, a self-attention layer is applied to continue building internal connections to finally produce the outputs by applying the feed-forward sub-layers:

$$\begin{aligned} \tilde{h}_i^k &= \text{SelfAtt}_{L \to L}(\hat{h}_i^k, \{\hat{h}_1^k, \ldots, \hat{v}_n^k\}) \\ \tilde{v}_j^k &= \text{SelfAtt}_{R \to R}(\hat{v}_j^{k-1}, \{\hat{v}_1^k, \ldots, \hat{v}_m^k\}) \end{aligned}$$

    As in the previous encoders, a residual connection and normalization layer is also applied after each sub-layer (represented by $\oplus$).

- **Output Representations**: LXMERT model returns three outputs for vision, language and cross-modality respectively, where the vision and language outputs are two feature sequences, and the cross-modality output is a special token [CLS] corresponding to the first element of the sequence returned in language feature sequence (similar functionality that in [15]) as shown in figure 4.10.

Therefore, the LXMERT model used in this work consists of three stack of encoders, where the first two are oriented for each of the individual modalities (textual-modality and visual-modality) separately followed by the third encoder focused on exchanging information and aligning the entities between the two modalities:

- Stack of $N_L$ = 9 Language Encoders.
- Stack of $N_R$ = 5 Object-Relationship Encoders.
- Stack of $N_X$ = 5 Cross-Modality Encoders.

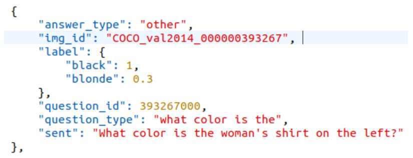
Where the model used in this work has been inherited from [3], which was pre-trained with data from several vision-and-language datasets, where all images come from MS-COCOCO or Visual Genome, and whose captions and questions come from the original MS-COCO and Visual Genome datasets (for captions), and VQA v2. 0, GQA balanced version and VG-QA (for questions), where none of the test data from any dataset is used for pre-training. A summary table (table 4.1) with the data used for the pre-training of the model is shown below:

| Image Split | Images | *Sentences (or Questions)* | | | | | |
|---|---|---|---|---|---|---|---|
| | | *COCO-Cap* | *VG-Cap* | *VQA* | *GQA* | *VG-QA* | *All* |
| *MS COCO - VG* | *72K* | *361K* | - | *387K* | - | - | *0.75M* |
| *MS COCO ∩ VG* | *51K* | *256K* | *2.54M* | *271K* | *515K* | *724K* | *4.30M* |
| *VG - MS COCO* | *57K* | - | *2.85M* | - | *556K* | *718K* | *4.13M* |
| *All* | *180K* | *617K* | *5.39M* | *658K* | *1.07M* | *1.44M* | *9.18M* |

**Table 4.1:** Data used for pre-training in LXMERT taken from [3].

Once the pre-trained LXMERT model was obtained and loaded, it only remained to carry out the fine-tunnig for the VQA task of Multiclass classification (3129 most frequent answers), where the data used corresponded to those of the VQA v2.0 dataset [4], where it is necessary to pre-process the original data:

- To obtain the pre-trained LXMERT model-compatible representations with respect to the Visual modality, a pre-trained Faster R-CNN in Visual Genome Dataset was used to detect 36 objects representing each of them as their 2048-dimensional region of interest and their four bounding box coordinates.
- To obtain the representations compatible with the pre-trained LXMERT model with respect to the Textual modality, BERT's WordPiece tokenizer was used, where the format of the original annotations has been modified to the format shown in figure 4.11:



```
{
    "answer_type": "other",
    "img_id": "COCO_val2014_000000393267",
    "label": {
        "black": 1,
        "blonde": 0.3
    },
    "question_id": 393267000,
    "question_type": "what color is the",
    "sent": "What color is the woman's shirt on the left?"
},
```

**Figure 4.11:** Annotation format for LXMERT

# 5 Experiments and results

## 5.1 Experimental Protocol

To carry out the experiments we used an OMEN 30L Desktop GT13-0043ns (figure 5.1), whose main features are:

- Intel(R) Core(TM) i9-10850K CPU @ 3.60GHz CPU
- 64GB RAM
- GeForce RTX 3090 24GB GPU
- 1TB SSD + 2TB HDD of storage space

The operating system used has been Ubuntu 20.04 LTS, where it was necessary to install Python 3.8.10 along with the libraries used, and to apply Pytorch it was necessary to install and configure the drivers and libraries compatible with the RTX 3090 GPU. For the configuration and installation of the Python work environment, a virtual environment was used, where the code editor used was Visual Studio Code, which allowed debugging and experimenting with the model repositories.

**Figure 5.1:** Hardware and software used in this work

Once the equipment was configured, we moved on to the experimental phase, where we tried to replicate the results of the state of the art. To do this, we started experimenting with the MMF framework [85] (whose code is available in the github repository [86]) where the winner of the VQA Challenge 2020 is available. When we experimented with this framework trying to replicate results, it gave problems both in terms of resources and image features retrieval using [21]. Therefore, after encountering these problems, we opted for a model that did not require so many resources and that could obtain the image features with a view to being able to apply the model to other different datasets, finally choosing LXMERT as the model, whose code both for obtaining the image features and for executing the fine-tunnig of the model is available in github repositories ([87] and [5] respectively).

The protocol for carrying out the experiments on each model is explained below:

### 5.1.1 MoViE+MCAN model

First, different VQA models were investigated until the winning VQA Challenge 2020 (MoVie+MCAN) was found, which is implemented in the MMF framework [85], whose code is available in the GitHub project [86].

Once this model was chosen, the MMF repository [86] was cloned, which allowed experimenting and understanding the structure and functionality of the framework, and thus creating the necessary environment variables in the equipment, for example, to indicate the path where the data will be stored when downloaded by the framework.

Once the environment was configured, the following experiments were carried out:

- Replicating results from the MoVie+MCAN model with the grid features provided by the MMF framework [86].
- Replicating results of the MoVie+MCAN model with the grid features extracted with the GitHub project [21] and attempt to train the MoViE+MCAN model with these last extracted image grid features.

After these experiments, we came to the conclusion that it was not feasible to use the MoVie+MCAN model, so we returned to investigate VQA models that required less time and resources, which led us to the LXMERT model.

### 5.1.2 LXMERT model

As discussed in Chapter 4 (VQA models), this is a pre-trained model in 5 tasks that allows to subsequently perform a fine-tunnig in the VQA task obtaining similar results in less time starting from the pre-trained model in the 5 subtasks.

To carry out the experiments with this last model, it was necessary to use the LXMERT repository [5] to reproduce the results, as well as to use the GitHub repository [87] to obtain the RoI features as well as the bounding box coordinates. Therefore, the experiments carried out with this model were:

- Replicate results of the LXMERT model in VQA task with the RoI features and bounding box coordinates provided by the same repository of LXMERT [5].
- Replicate results of the LXMERT model in VQA task with the RoI features and bounding box coordinates obtained with the GitHub repository [87].
- Fine-tunnig of the LXMERT model in VQA task with the RoI features and bounding box coordinates obtained with the GitHub repository [87].

## 5.2 Results

### 5.2.1 MoViE+MCAN model

#### 5.2.1.1 Experiment 1 - Execute trained MoVie+MCAN model provided by MMF with data provided by own MMF

Once the MMF repository was installed and the environment variables were configured in the system, we started with the process of trying to replicate the results of the MoVie+MCAN model directly with the data that the MMF framework automatically downloads.

To do this, we proceeded to run the "predict.py" script (located in the repository path "/mmf/mmf_cli/predict.py") from Visual Studio Code (VS Code) adding the necessary configuration parameters (see table 5.1), where a storage space problem appeared when decompressing the data due to the large amount that is automatically downloaded (more than 1 TB).

| Argument | Value |
|---|---|
| *config* | *projects/movie_mcan/configs/vqa2/defaults.yaml* |
| *model* | *movie_mcan* |
| *dataset* | *vqa2* |
| *run_type* | *test* |
| *checkpoint.resume_zoo* | *movie_mcan.grid.vqa2_vg* |

**Table 5.1:** Configuration arguments of MMF execution for testing.

Since the MMF framework creates and stores a json file as the data is first downloaded (compressed in ".tar. gz" format), which indicates that the data have been downloaded correctly so that the next time the model is run the download and decompression of the data is not performed again, we proceeded with the automatic download of the compressed data together with the creation of the json files, but the decompression of the data was performed manually using two hard disks, one of 1 TB where the compressed data were stored, and another of 2 TB where the decompressed data were stored.

Once the data was correctly decompressed, the evaluation of the MoVie+MCAN model was carried out on the test data, taking approximately 2 hours and obtaining a json with the results in the appropriate format to be evaluated by uploading it to evalAI [18]. Where the following results were obtained and shown in the table 5.2

| VQA Model | Data | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *MoViE+MCAN* | *MMF Repository* | *test-dev* | *89.13%* | *58.31%* | *64.49%* | *73.94%* |

**Table 5.2:** Results obtained by running MoViE+MCAN model with data from the MMF repository.

Observing that the results of the MoVie+MCAN model winner of the 2020 VQA Challenge are reproduced.

#### 5.2.1.2 Experiment 2 - Execute trained MoVie+MCAN model provided by MMF with data extract with GitHub repository [21]

With a view to being able to test the same model on other datasets (e.g.: the edBB dataset), the experiment of evaluating the MoViE+MCAN model with the image grid features extracted with the help of the GitHub repository [21] (referenced from MMF as the repository that was used to obtain the image grid features used by the 2020 winning MoViE+MCAN model) was carried out.

For this, it was necessary to clone and install this repository, which needed to be used from a linux operating system (e.g.: Ubuntu), since this repository uses the Detectron2 library [88]. Once the working environment was configured, the script "extract_grid_feature.py" located in the path "/grid-feats-vqa/extract_grid_feature.py" was executed with the configuration of "/grid-feats-vqa/configs/X-152-challenge.yaml" to obtain the image grid features from the test image data.

To validate that the execution had worked correctly, the image grid features of some randomly chosen images were compared with the image grid features that could be downloaded directly from the GitHub repository [21], observing that the dimensions of the image grid features were the same.

Therefore, once the image grid features were obtained, we proceeded to evaluate the model of MoViE+MCAN from MMF with the new data, obtaining the following results shown in the table 5.3:

| VQA Model | Data | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *MoViE+MCAN* | *Extrated with [21]* | *test-dev* | *83.06%* | *34.09%* | *56.86%* | *65.07%* |

**Table 5.3:** Results obtained by running MoViE+MCAN model with extracted data from GitHub repository [21].

It is observed that the results obtained are worse than with the image grid features that were automatically downloaded from MMF. So we compared the image grid feature from MMF with those extracted from the GitHub repository [21], observing that both the values and the dimensions of the image grid features were different.

Due to the poor results obtained because the extracted image grid features were different, it was decided to try to re-train the model with the features extracted with the code from the GitHub repository [21], to see what results could be achieved, and if it was possible to obtain similar results. Therefore, we executed the script "run.py" located in "/mmf/mmf_cli/run.py" from Visual Studio Code setting the parameters shown in the table 5.4, appearing a problem due to lack of memory in GPU due to the calculations with the gradient, which was solved using the accumulated gradient with a reduction of the batch size. However, once the training started, it was observed that it took more than 100 days to train, so it was beyond the scope of the work, both in terms of time and resources.

| Argument | Value |
|---|---|
| *config* | *projects/movie_mcan/configs/vqa2/defaults.yaml* |
| *model* | *movie_mcan* |
| *dataset* | *vqa2* |
| *run_type* | *train_val* |

**Table 5.4:** Configuration arguments of MMF execution for training.

Therefore, a new search was started for a VQA model that had available both the implementation of the model and the obtaining of the image features, which could be executed in reasonable times with the available resources. The model that was finally chosen was LXMERT [3].

### 5.2.2 LXMERT model

#### 5.2.2.1 Experiment 3 - Fine-tuning in the VQA task of pre-trained LXMERT model with the data that provide LXMERT repository

Once the LXMERT repository is installed and configured on the system, we begin the process of trying to replicate the results of the LXMERT model directly with the data that the LXMERT repository provides for download, skipping the step of applying the bottom-up attention model (Faster R-CNN) [36] on raw images.

To do this, we proceeded to run a custom script from Visual Studio Code (VS Code) that simulate execute "vqa_finetune.bash" script (located in the repository path "/run/vqa_finetune.bash") with the configuration parameters shown in table 5.5 where it was necessary download the pre-trained model that is provided by LXMERT repository for fine-tunning in the VQA task.

| Argument | Value |
|---|---|
| *train* | *train,nominival* |
| *valid* | *minival* |
| *llayers* | *9* |
| *xlayers* | *5* |
| *rlayers* | *5* |
| *loadLXMERTQA* | *<path where model pre-trained was saved >* |
| *batchSize* | *32* |
| *optim* | *bert* |
| *lr* | *5e-5* |
| *epochs* | *4* |
| *tqdm* | *True* |
| *output* | *<path where results are saved >* |

**Table 5.5:** Configuration arguments of LXMERT fine-tunnig for VQA task.

Performed fine-tunning on the VQA task, we proceeded to evaluate the fine-tuned model on the VQA test data, managing to replicate the results of the paper [3] shown in the table 5.6 below:

| VQA Model | Data | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | *LXMERT Repository* | *test-dev* | *88.34%* | *54.55%* | *63.29%* | *72.62%* |

**Table 5.6:** Results obtained by running LXMERT model with data from the LXMERT repository.

Following the same strategy as with the previous model, MoViE+MCAN, of autonomously extracting the image features (in this case RoI features and bounding box coordinates) in order to test the same model, LXMERT, on other datasets (for example: the edBB dataset), two GitHub repositories were tested to carry out the extraction of the image features.

#### 5.2.2.2 Experiment 4 - Fine-tuning in the VQA task of pre-trained LXMERT model with the data extracted by the bottom-up attention model implemented in Pytorch [87]

The repositories tested for image feature extraction were:

- Repository [89]: consists of a Pytorch implementation of the bottom-up attention model (original bottom-up attetion model is implemented in Caffe [36]) using Detectron2 [88]. This repository was chosen as the first choice because the author is the same as the LXMERT repository [3] and the output format of the image features is compatible with the LXMERT model. However, when extracting the features and comparing them with those that can be downloaded directly from the LXMERT repository corresponding to those obtained by the original bottom-up attetion model on Caffe, it was observed that they were different, so it was decided to use the second repository [87].

- Repository [87]: consists of a Pytorch implementation of the bottom-up attention model using Detectron2 based on the original bottom-up attetion model implemented on Caffe. When carrying out the extraction of image features, it was observed that the same objects were detected with image features practically the same as those obtained with the original Caffe implementation (deviation less than 0.01), where it was necessary to modify the output format so that the image features were stored in the format compatible with LXMERT. Note that having the implementation in Detectron2 (Pytorch) will make it easier to test image feature extractions with new Faster R-CNN structures for future experiments.

Therefore, once the image features were extracted with the [87] repository, we proceeded to run the fine-tunnig of the pre-trained LXMERT model in the VQA task with the last extracted features and the same parameters used as in the previous experiment (see table 5.5), as well as its evaluation with the test data, obtaining the results shown in the table 5.7:

| VQA Model | Data | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | *Extrated with [87]* | *test-dev* | *88.19%* | *54.52%* | *63.05%* | *72.44%* |

**Table 5.7:** Results obtained by running LXMERT model with data extracted with bottom-up attention model.

Looking at the results in the table, it is concluded that the objective of reproducing the state of the art results of the LXMERT model with the image features extracted autonomously has been achieved, since the results with both image features have very similar accuracies, where it is assumed that the difference is due to that deviation of less than 0.01 in the image features. Therefore, once the new e-learning database based on edBB is built, it will be possible to test running the LXMERT model on it to analyze how it behaves in this new database.

Finally, a fine-tunnig was carried out in the VQA task of the pre-trained LXMERT model by varying hyper-parameters, with the intention of finding a better configuration than the one provided by the LXMERT repository [5].

#### 5.2.2.3 Experiment 5 - Fine-tuning in the VQA task of pre-trained LXMERT model with the data extracted by the bottom-up attention model implemented in Pytorch [87] testing different hyper-parameters

For this purpose, the following hyper-parameter variations were carried out, where all hyper-parameters were left the same as in the LXMERT repository ($epochs = 4$, $learning\ rate = 5 \cdot 10^{-5}$ and $batch\ size = 32$), except for one of them which was varied. Where finally the results were compared by applying the best fine-tuned models (according to minival dataset, a small subset of the valid dataset that is not used for training) on the test data:

- **Varying hyper-parameter batch size**: To perform this experiment, we kept the hyper-parameters of "*epochs*" at "4" and "*learning rate*" at "$5 \cdot 10^{-5}$" fixed, and tested with 3 different "*batch size*" (32, 64 and 128), where the results obtained in train and test are shown in figure 5.2.

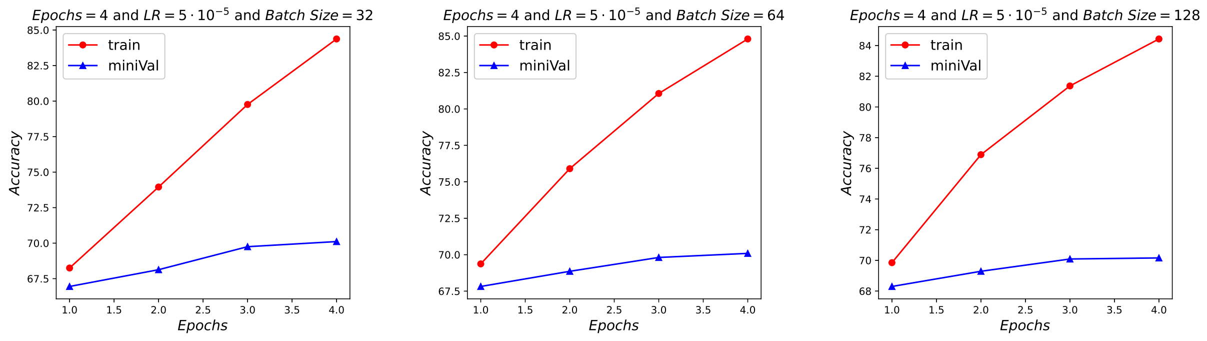


**Figure 5.2:** Training and mini-validation results with different batch sizes

Observing that the best results in the minival split in all the batch size variations are obtained in the last epoch (4 in all), it is decided to evaluate them in the test split in evalAI [18] obtaining the following results shown in table 5.8:

| VQA Model | Hyper-parameters | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | $epochs = 4$ <br> $lr = 5 \cdot 10^{-5}$ <br> $batchSize = 32$ | *test-dev* | *88.19%* | *54.52%* | *63.05%* | *72.44%* |
| *LXMERT* | $epochs = 4$ <br> $lr = 5 \cdot 10^{-5}$ <br> $batchSize = 64$ | *test-dev* | *88.23%* | *55.00%* | *63.16%* | *72.56%* |
| *LXMERT* | $epochs = 4$ <br> $lr = 5 \cdot 10^{-5}$ <br> $batchSize = 128$ | *test-dev* | *88.02%* | *54.81,%* | *63.19%* | *72.47%* |

**Table 5.8:** Results obtained on test data with fine-tuned LXMERT model with data extracted with bottom-up attention model.

- **Varying hyper-parameter epochs**: To carry out this experiment, the hyper-parameters of ”*learning rate*” at ”$5 \cdot 10^{-5}$” and ”*batch size*” at ”32” were kept fixed, and tested with 3 different number of ”*epochs*” (4, 6 and 8), where the results obtained in train and test are shown in Figure 5.3.

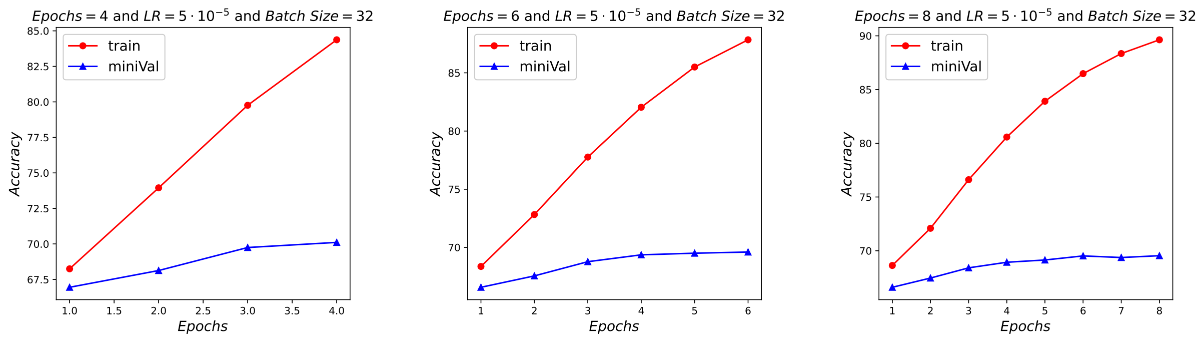


**Figure 5.3:** Training and mini-validation results with different epochs

Observing that the best results in the minival split in all the epoch variations are obtained in the last epoch (4, 6 and 8 respectively), it is decided to evaluate them in the test split in evalAI [18] obtaining the following results shown in table 5.9:

| VQA Model | Hyper-parameters | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | $epochs = 4$ $lr = 5 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *88.19%* | *54.52%* | *63.05%* | *72.44%* |
| *LXMERT* | $epochs = 6$ $lr = 5 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *88.36%* | *54.25%* | *62.74%* | *72.33%* |
| *LXMERT* | $epochs = 8$ $lr = 5 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *88.21%* | *54.22%* | *62.45%* | *72.13%* |

**Table 5.9:** Results obtained on test data with fine-tuned LXMERT model with data extracted with bottom-up attention model.

- **Varying hyper-parameter learning rate**: To carry out this experiment, the hyper-parameters of "*epochs*" at "4" and "*batch size*" at "32" were kept fixed, and 3 different "*learning rate*" ($1 \cdot 10^{-5}$, $2.5 \cdot 10^{-5}$ and $5 \cdot 10^{-5}$) were tested, where the results obtained in train and test are shown in figure 5.4.

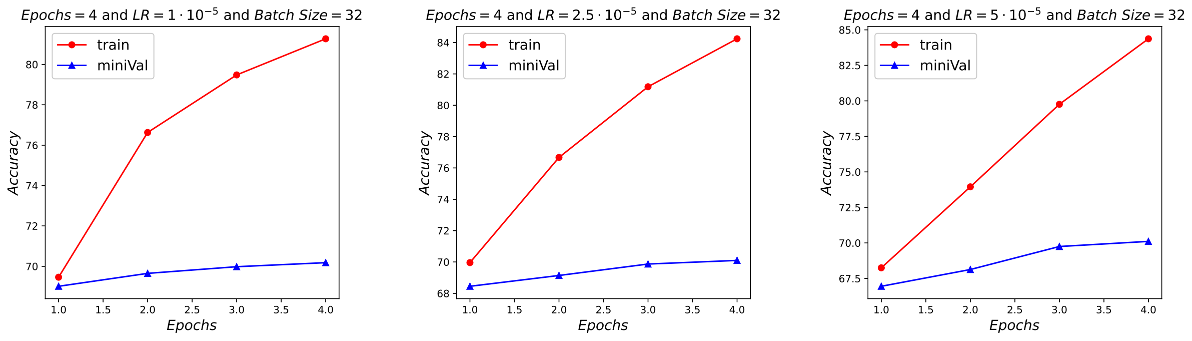


**Figure 5.4:** Training and mini-validation results with different learning rates

Observing that the best results in the minival split in all the learning rate variations are obtained in the last epoch (4 in all), it is decided to evaluate them in the test split in evalAI [18] obtaining the following results shown in table 5.10:

| VQA Model | Hyper-parameters | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | $epochs = 4$ $lr = 5 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *88.19%* | *54.52%* | *63.05%* | *72.44%* |
| *LXMERT* | $epochs = 4$ $lr = 2.5 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *87.94%* | *54.52%* | *63.06%* | *72.34%* |
| *LXMERT* | $epochs = 4$ $lr = 1 \cdot 10^{-5}$ $batchSize = 32$ | *test-dev* | *87.54%* | *54.64%* | *62.93%* | *72.13%* |

**Table 5.10:** Results obtained on test data with fine-tuned LXMERT model with data extracted with bottom-up attention model.

To conclude the experiments, observing the best results in the minival data set, a combination of the best performing hyper-parameters by "separated" ($epochs = 4$, $lr = 1 \cdot 10^{-5}$ and $batchSize = 128$) is tested, which a priori would not necessarily provide better results, since the hyper-parameters are dependent on each other. Therefore, testing this combination yielded the results shown in Figure 5.5, being in the fourth epoch where better results are obtained in the minival split, which are worse than those obtained with other previously tested combinations (e.g. $epochs = 4$, $learning\ rate = 1 \cdot 10^{-5}$ and $batch\ size = 32$).

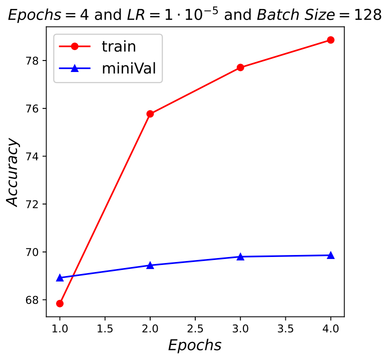


**Figure 5.5:** Training and mini-validation results with a combination hyper-parameters

And testing to evaluate the data in the test split evalAI [18] we get the results shown in table 5.11, which are still worse than with other combinations.

| VQA Model | Hyper-parameters | Eval. Phase | Results ($Accuracy_{VQA}$) | | | |
|---|---|---|---|---|---|---|
| | | | Yes/No | Number | Other | Overall |
| *LXMERT* | $epochs = 4$<br>$lr = 1 \cdot 10^{-5}$<br>$batchSize = 128$ | *test-dev* | *87.26%* | *54.46%* | *62.92%* | *71.98%* |

**Table 5.11:** Results obtained on test data with fine-tuned LXMERT model with data extracted with bottom-up attention model.

# 6 Conclusions and future challenges

## 6.1 Conclusions

After the completion of this master's thesis, we can conclude that currently the multimodal task of VQA is of great interest due largely to the great advances in deep learning. One of the key points in recent years has been the attention mechanisms to obtain better results, as well as the VQA v2.0 dataset that has allowed in recent years to hold the VQA Challenge whose winners present their solution at the VQA Workshop of CVPR, where the winner of 2020 (MoViE+MCAN) obtained a 76.19% of Accuracy on test-dev.

Regarding the models used in this work, we can conclude that the winning model of the 2020 VQA Challenge, MoViE+MCAN, is beyond the scope of the work due to lack of resources needed to properly train and run it. We were anyway able to replicate its results based on the image grid features provided by MMF. However, it was not possible to replicate the full results of MoViE+MCAN (grid features) with the repository that was supposed to have been used as cited by MMF. Therefore, we switched to the LXMERT model, which allowed us to replicate the results with a slight difference due to the solution used to extract the feature image (region of interest and bounding box coordinates), where in our case we used the bottom-up attention model using the detectron2 library (implemented in Pytorch) instead of the original model (same model but implemented in Caffe).

Regarding the results obtained with LXMERT, especially those that reproduce the results of the state of the art, it is observed that in the VQA task the questions with "Yes/No" answers are those with the highest Accuracy ($\sim 88 - 89\%$) followed by the questions with answers called "Other" ($\sim 63 - 64\%$), i.e., different answers of numbers ("Number") and binary ("Yes/No"), and in last place would be the questions with answers of a "Number" ($\sim 54 - 58\%$). So it seems that there is still a great evolution in answering "Number" and "Other" questions, which currently may be affected by the fact that in the end the open-response VQA models tend to transform into Multi-class Classification problems, restricting the answers to the most frequent ones.

Finally, when testing LXMERT with several combinations of hyper-parameters (*epochs*, *learing rate* and *batch size*), the best results with respect to overall accuracy in dev-test are obtained with $epochs = 4$, $learing\ rate = 5 \cdot 10^{-5}$ and $batch\ size = 64$ obtaining an Overall Accuracy of 72.56%.

Therefore, after obtaining these results, we can consider that we have reached the state of the art, which will allow us to apply the LXMERT model to other datasets, such as the one we are currently working on focused in e-learning based on the edBB dataset.

## 6.2 Future challenges

As future lines that arise as a result of this work with a view to continuing research in the field of VQA we identify the following ones:

- The main line of work would consist of continuing and finalising the creation of the new dataset for studying the application of VQA methods to improving e-learning audiovisual sessions based on the edBB database with the help of the BiDA Lab at UAM. The final purpose there will be applying the LXMERT model developed in this thesis to observe how it works in e-learning, and fine-tune it to that particular application scenario. Within this line of research, at first the LXMERT model trained with the VQA data would be applied directly on the new database, and then continue with the fine-tunning of the LXMERT model using the newly created e-learning dataset.
- Try to use other structures of object detection models to obtain the Region of Interest features, observing how it affects the results.
- Try to improve the LXMERT model by applying modifications to use more input data (e.g.: add grid features), add new pre-training sub-tasks, etc.

# Bibliography

[1] Duy-Kien Nguyen, Vedanuj Goswami, and Xinlei Chen. Movie: Revisiting modulated convolutions for visual counting and beyond. *arXiv preprint arXiv:2004.11883*, 2020. [`https://arxiv.org/abs/2004.11883`].

[2] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. [`https://arxiv.org/abs/2001.03615`].

[3] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2019. [`https://arxiv.org/abs/1908.07490`].

[4] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision*, 2016. [`https://arxiv.org/abs/1612.00837`].

[5] Lxmert: Learning cross-modality encoder representations from transformers. [`https://github.com/airsplay/lxmert`].

[6] Javier Hernandez-Ortega, Roberto Daza, Aythami Morales, Julian Fierrez, and Javier Ortega-Garcia. edbb: Biometrics and behavior for assessing remote education. *arXiv preprint arXiv:1912.04786*, 2019. [`https://arxiv.org/abs/1912.04786v1`].

[7] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. *Advances in Neural Information Processing Systems*, 2014. [`https://arxiv.org/abs/1410.0210`].

[8] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering. *Proceedings of the IEEE International Conference on Computer Vision*, 2015. [`https://arxiv.org/abs/1505.00468`].

[9] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 2017. [`https://arxiv.org/abs/1602.07332`].

[10] Vqa: Visual question answering. [`https://visualqa.org/challenge.html`].

[11] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [`https://arxiv.org/abs/1906.10770`].

[12] Sruthy Manmadhan, · Binsu, and C Kovoor. Visual question answering: a state-of-the-art review. *Artificial Intelligence Review*, 2020. [https://doi.org/10.1007/s10462-020-09832-7].

[13] Silvio Barra, Carmen Bisogni, Maria De Marsico, and Stefano Ricciardi. Visual question answering: which investigated applications? *arXiv preprint arXiv:2103.02937*, 2021. [https://arxiv.org/abs/2103.02937].

[14] Cemil Sungur and Akif Durdu. Real-time diseases detection of grape and grape leaves using faster r-cnn and ssd mobilenet architectures. 2019. [url-https://www.researchgate.net/publication/334987612].

[15] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019. [https://arxiv.org/abs/1810.04805].

[16] Dongxiang Zhang, Rui Cao, and Sai Wu. Information fusion in visual question answering: A survey. *Information Fusion*, 2019. [https://www.sciencedirect.com/science/article/pii/S1566253518308893].

[17] Vqa: Visual question answering. [https://visualqa.org/evaluation.html].

[18] Overview - evalai. [https://eval.ai/web/challenges/challenge-page/830/overview].

[19] Vqa: Visual question answering. [https://visualqa.org/workshop_2020.html].

[20] Vqa: Visual question answering. [https://visualqa.org/download.html].

[21] Grid features pre-training code for visual question answering. [https://github.com/facebookresearch/grid-feats-vqa].

[22] Andrew Fitzgibbon. All data ai with dr. andrew fitzgibbon - microsoft research, 2019. [https://www.microsoft.com/en-us/research/podcast/all-data-ai-with-dr-andrew-fitzgibbon/].

[23] Iberoamerican congress on pattern recognition — springerlink, 2018. [https://link.springer.com/conference/ciarp].

[24] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. [https://arxiv.org/abs/1705.09406].

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. [https://arxiv.org/abs/1409.1556].

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [https://arxiv.org/abs/1512.03385].

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. [https://arxiv.org/abs/1703.06870].

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. [https://arxiv.org/abs/1706.03762].

[29] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020. [https://arxiv.org/abs/2005.12872].

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. [https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf].

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013. [https://arxiv.org/abs/1301.3781].

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 2013. [https://arxiv.org/abs/1310.4546].

[33] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014. [https://www.aclweb.org/anthology/D14-1162.pdf].

[34] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2017. [https://arxiv.org/abs/1708.02711].

[35] Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia v0.1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*, 2018. [https://arxiv.org/abs/1807.09956].

[36] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [https://arxiv.org/abs/1707.07998].

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. [https://arxiv.org/abs/1506.01497].

[38] Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. *arXiv preprint arXiv:2004.00849*, 2020. [https://arxiv.org/abs/2004.00849].

[39] Haiyang Xu, Ming Yan, Chenliang Li, Bin Bi, Songfang Huang, Wenming Xiao, and Fei Huang. E2e-vlp: End-to-end vision-language pre-training enhanced by visual learning. *arXiv preprint arXiv:2106.01804*, 2021. [https://arxiv.org/abs/2106.01804].

[40] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936. [http://www.mathweb.zju.edu.cn:8080/wjd/TN/T3.pdf].

[41] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 2016. [https://arxiv.org/abs/1607.04606].

[42] Yang Shi, Tommaso Furlanello, Sheng Zha, and Animashree Anandkumar. Question type guided attention in visual question answering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. [https://arxiv.org/abs/1804.02088v2].

[43] Chao Ma, Chunhua Shen, Anthony Dick, Qi Wu, Peng Wang, Anton van den Hengel, and Ian Reid. Visual question answering with memory-augmented networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2017. [https://arxiv.org/abs/1707.04968v2].

[44] Junwei Liang, Lu Jiang, Liangliang Cao, Li-Jia Li, and Alexander Hauptmann. Focal visual-text attention for visual question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. [https://arxiv.org/abs/1806.01873v2].

[45] Duy-Kien Nguyen and Takayuki Okatani. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [https://arxiv.org/abs/1804.00775v2].

[46] Lianli Gao, Pengpeng Zeng, Jingkuan Song, Xianglong Liu, and Heng Tao Shen. Examine before you answer: Multi-task learning with adaptive-attentions for multiple-choice vqa. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, 2018. [https://doi.org/10.1145/3240508.3240687].

[47] Khyathi Raghavi Chandu, Mary Arpita Pyreddy, Matthieu Felix, and Narendra Nath Joshi. Textually enriched neural module networks for visual question answering. *arXiv preprint arXiv:1809.08697*, 2018. [https://arxiv.org/abs/1809.08697].

[48] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015. [https://arxiv.org/abs/1511.05960].

[49] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering. *Advances in Neural Information Processing Systems*, 2015. [https://arxiv.org/abs/1505.05612].

[50] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [https://openaccess.thecvf.com/content_cvpr_2016/papers/Yang_Stacked_Attention_Networks_CVPR_2016_paper.pdf].

[51] Jingkuan Song, Pengpeng Zeng, Lianli Gao, and Heng Tao Shen. From pixels to objects: Cubic visual attention for visual question answering. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 2018. [https://doi.org/10.24963/ijcai.2018/126].

[52] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016. [https://link.springer.com/chapter/10.1007/978-3-319-46478-7_28].

[53] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. [https://arxiv.org/abs/1511.03416].

[54] Jasdeep Singh, Vincent Ying, and Alex Nutkiewicz. Attention on Attention: Architectures for Visual Question Answering (VQA). *arXiv preprint arXiv:1803.07724*, 2018. [https://arxiv.org/abs/1803.07724].

[55] Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. A focused dynamic attention model for visual question answering. *arXiv preprint arXiv:1604.01485*, 2016. [https://arxiv.org/abs/1604.01485].

[56] Ilija Ilievski and Jiashi Feng. Generative attention model with adversarial self-learning for visual question answering. *Proceedings of the on Thematic Workshops of ACM Multimedia 2017 - Thematic Workshops '17*, 2017. [https://doi.org/10.1145/3126686.3126695].

[57] Zhou Su, Chen Zhu, Yinpeng Dong, Dongqi Cai, Yurong Chen, and Jianguo Li. Learning Visual Knowledge Memory Networks for Visual Question Answering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[58] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2016. [https://arxiv.org/abs/1606.01847].

[59] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2016. [https://arxiv.org/abs/1610.04325].

[60] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. *Proceedings of the IEEE International Conference on Computer Vision*, 2017. [https://arxiv.org/abs/1708.01471].

[61] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. *Advances in Neural Information Processing Systems*, 2015. [https://proceedings.neurips.cc/paper/2015/hash/831c2f88a604a07ca94314b56a4921b8-Abstract.html].

[62] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. [https://openaccess.thecvf.com/content_iccv_2015/html/Malinowski_Ask_Your_Neurons_ICCV_2015_paper.html].

[63] Mengye Ren, Ryan Kiros, and Richard Zemel. Image question answering: A visual semantic embedding model and a new dataset. *Advances in Neural Information Processing Systems*, 2015. [https://arxiv.org/abs/1505.02074v1].

[64] Lin Ma, Zhengdong Lu, and Hang Li. Learning to answer questions from image using convolutional neural network. *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. [https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/11745].

[65] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [https://openaccess.thecvf.com/content_cvpr_2016/html/Noh_Image_Question_Answering_CVPR_2016_paper.html].

[66] Peng Gao, Hongsheng Li, Shuang Li, Pan Lu, Yikang Li, Steven C.H. Hoi, and Xiaogang Wang. Question-guided hybrid convolution for visual question answering. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. [https://openaccess.thecvf.com/content_ECCV_2018/html/gao_peng_Question-Guided_Hybrid_Convolution_ECCV_2018_paper.html].

[67] Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. [https://ieeexplore.ieee.org/document/8334194].

[68] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Advances in Neural Information Processing Systems*, 2016. [https://arxiv.org/abs/1606.00061].

[69] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012. [https://link.springer.com/chapter/10.1007/978-3-642-33715-4_54].

[70] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. [https://arxiv.org/abs/1405.0312].

[71] C Lawrence Zitnick, Devi Parikh, and Virginia Tech. Bringing semantics into focus using visual abstraction. *openaccess.thecvf.com*, 2013. [https://openaccess.thecvf.com/content_cvpr_2013/html/Zitnick_Bringing_Semantics_into_2013_CVPR_paper.html].

[72] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2015. [https://arxiv.org/abs/1503.01817].

[73] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *International Journal of Computer Vision*, 2015. [https://arxiv.org/abs/1505.04870].

[74] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [https://openaccess.thecvf.com/content_cvpr_2016/html/Andreas_Neural_Module_Networks_CVPR_2016_paper.html].

[75] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2016.

[76] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [https://arxiv.org/abs/1904.08920].

[77] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020. [`https://arxiv.org/abs/2003.12462`].

[78] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images. *arXiv preprint arXiv:2007.00398*, 2020. [`https://arxiv.org/abs/2007.00398`].

[79] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [`https://arxiv.org/abs/1906.00067`].

[80] Asma Ben Abacha, Sadid A Hasan, Vivek V Datla, Joey Liu, Dina Demner-Fushman, and Henning Müller. Vqa-med: Overview of the medical visual question answering task at imageclef 2019. *ceur-ws.org*, 2019. [`http://ceur-ws.org/Vol-2380/paper_272.pdf`].

[81] Zhibiao Wu and Martha Palmer. Verb Semantics and Lexical Selection. *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994. [`https://arxiv.org/abs/cmp-lg/9406033`].

[82] Kushal Kafle and Christopher Kanan. An analysis of visual question answering algorithms. *Proceedings of the IEEE International Conference on Computer Vision*, 2017. [`https://arxiv.org/abs/1703.09684`].

[83] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *ACL*, 2001. [`https://www.aclweb.org/anthology/P02-1040/`].

[84] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2015. [`https://www.aclweb.org/anthology/W14-3348/`].

[85] Mmf: A modular framework for vision and language multimodal research from facebook ai research (fair). [`https://mmf.sh/`].

[86] Amanpreet Singh, Vedanuj Goswami, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Mmf: A multimodal framework for vision and language research, 2020. [`https://github.com/facebookresearch/mmf`].

[87] Zhou Yu, Jing Li, Tongan Luo, and Jun Yu. A pytorch implementation of bottom-up-attention, 2020. [`https://github.com/MILVLG/bottom-up-attention.pytorch`].

[88] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. [`https://github.com/facebookresearch/detectron2`].

[89] Pytorch bottom-up attention with detectron2. [`https://github.com/airsplay/py-bottom-up-attention`].