Open-Ended Visual Question-Answering

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

In partial fulfilment
of the requirements for the degree in
SCIENCE AND TELECOMMUNICATION TECHNOLOGIES
ENGINEERING

Author: Issey Masuda Mora
Advisors: Xavier Giró i Nieto, Santiago Pascual de la Puente

Universitat Politècnica de Catalunya (UPC)
June 2016

Abstract

This thesis studies methods to solve Visual Question-Answering (VQA) tasks with a Deep Learning framework.

As a preliminary step, we explore Long Short-Term Memory (LSTM) networks used in Natural Language Processing (NLP) to tackle Question-Answering (text based).

We then modify the previous model to accept an image as an input in addition to the question. For this purpose, we explore the VGG-16 and K-CNN convolutional neural networks to extract visual features from the image. These are merged with the word embedding or with a sentence embedding of the question to predict the answer.

This work was successfully submitted to the Visual Question Answering Challenge 2016, where it achieved a 53,62% of accuracy in the test dataset.

The developed software has followed the best programming practices and Python code style, providing a consistent baseline in Keras for different configurations. The source code and models are publicly available at `https://github.com/imatge-upc/vqa-2016-cvprw`.

Resum

Aquesta tesis estudia mètodes per resoldre tasques de *Visual Question-Answering* emprant tècniques de *Deep Learning*.

Com a pas preliminar, explorem les xarxes *Long Short-Term Memory* (LSTM) que s'utilitzen en el Processat del Llenguatge Natural (NLP) per atacar tasques de *Question-Answering* basades únicament en text.

A continuació modifiquem el model anterior per acceptar una imatge com a entrada juntament amb la pregunta. Per aquest propòsit, estudiem l'ús de les xarxes convolucionals VGG-16 i K-CNN per tal d'extreure els descriptors visuals de la imatge. Aquests descriptors són fusionats amb el *word embedding* o *sentence embedding* de la pregunta per poder predir la resposta.

Aquest treball ha estat presentat al *Visual Question Answering Challenge 2016*, on ha obtingut una precisió del 53,62% en les dades de test.

El software desenvolupat ha emprat bones pràctiques en programació i ha seguit les directrius d'estil de Python, proveïnt un projecte base en Keras consistent a diferents configuracions. El codi font i els models són públics a `https://github.com/imatge-upc/vqa-2016-cvprw`.

Resumen

Esta tesis estudia métodos para resolver tareas de *Visual Question-Answering* usando técnicas de *Deep Learning*.

Como primer paso, exploramos las redes *Long Short-Term Memory* (LST) que se usan en el Procesado del Lenguaje Natural (NLP) para atacar tareas de *Question-Answering* basadas únicamente en texto.

A continuación modificamos el modelo anterior para aceptar una imagen como entrada junto con la pregunta. Para este propósito, estudiamos el uso de las redes convolucionales VGG-16 y K-CNN para extraer los descriptores visuales de la imagen. Estos descriptores son fusionados con el *word embedding* o *sentence embedding* de la pregunta para poder predecir la respuesta.

Este trabajo se ha presentado al *Visual Question Answering Challenge 2016*, donde ha obtenido una precisión del 53,62% en los datos de test.

El software desarrollado ha usado buenas prácticas de programación y ha seguido las directrices de estilo de Python, proveyendo un proyecto base en Keras consistente a distintas configuraciones. El código fuente y los modelos son públicos en `https://github.com/imatge-upc/vqa-2016-cvprw`.

Acknowledgements

I would like to thank to my tutor Xavier Giró i Nieto for his help during the whole project, for letting me join his research group and initiating me in this amazing field and for his patience in my moments of stubbornness.

I also want to thank Santiago Pascual de la Puente for the countless times he helped me throughout the project with his wise advises and knowledge in Deep Learning.

My partners in the X-theses group deserve also a mention here as talking with them week after week about the project and listening what they have been researching has enriched this project. Together with them I would also want to thank Albert Gil for his help and support regarding the GPI cluster usage.

I would also like to thank to Marc Bolaños, Petia Radeva and the rest of the Computer Vision group in Universitat de Barcelona for their advice and for providing us with very useful data for our experiments.

Last but not least, I want to thank to my family and friends for being there when needed.

Revision history and approval record

Revision	Date	Purpose
0	11/06/2016	Document creation
1	20/06/2016	Document finalization
2	24/06/2015	Document revision & corrections
3	25/06/2015	Document revision & corrections
4	26/06/2015	Document revision & corrections
5	27/06/2015	Document revision & corrections

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Issey Masuda Mora	issey.masuda@alu-etsetb.upc.edu
Xavier Giró i Nieto	xavier.giro@upc.edu
Santiago Pascual de la Puente	santiago.pascual@tsc.upc.edu

Written by:		Reviewed and approved by:		Reviewed and approved by:	
Date	20/06/2016	**Date**	27/06/2016	**Date**	27/06/2016
Name	Issey Masuda Mora	**Name**	Xavier Giró i Nieto	**Name**	Santiago Pascual de la Puente
Position	Project Author	**Position**	Project Supervisor	**Position**	Project Co-supervisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Statement of purpose

In the last few years the number of published papers and job offers related to Deep Learning have exploded. Both the academic world and the industry are pushing forward to speed up the developments and the research in this area. The reason is that Deep Learning has shown a great performance solving a lot of problems that were previously tackled by more classic Machine Learning algorithms and it has also opened the door to more complex tasks that we could not solve before.

We humans are constantly asking questions and answering them. This is the way we learn, we transfer knowledge and lastly we communicate with each other. This basic framework of communication has inspired other ways of communications such as the HTTP protocol which is basically a combination of a request (question) and a response (answer). Frequently Asked Questions (FAQ) also uses this format.

But what about machines? Artificial Intelligence is a huge science-fiction topic and it is recurrently all over the news and media, but the reality is not that far from there. Deep Neural Networks are nowadays used in our everyday life when we surf the net, when we use recommendation systems or automatic translation systems. This has also been extended to tackle Question-Answering tasks from the Natural Language Processing perspective (*e.g.* Facebook AI Research presented a set of tasks, called bAbI [23], to evaluate AI models' text understanding and reasoning).

Visual Question-Answering has emerged as an evolution of these text-based QA systems. These models aim to be able to answer a given natural question related to a given image. One of the interests in such models is that in order to succeed in these visual QA tasks (or even just text-based QA), they need to have a much deeper level of *reasoning* and understanding than other similar models, for example image captioning models. An example of a VQA task is shown in Figure 1.1

This thesis studies new models to tackle VQA problems. The common point of all these models is that they use Convolutional Neural Networks (CNN) to process the image and extract visual features, which are a summarized representation of the image, and Long Short-Term Memory networks (LSTM), a flavor of Recurrent Neural Network (RNN), to process the question sequence.

Based on the given context, the main objectives of this project are:

- Explore the techniques used for text-based Question-Answering

- Build a model able to perform visual question-answering

- Compare which approach is better to process the question: word embedding or sentence embedding. These are two different techniques to project a text into a space with semantic
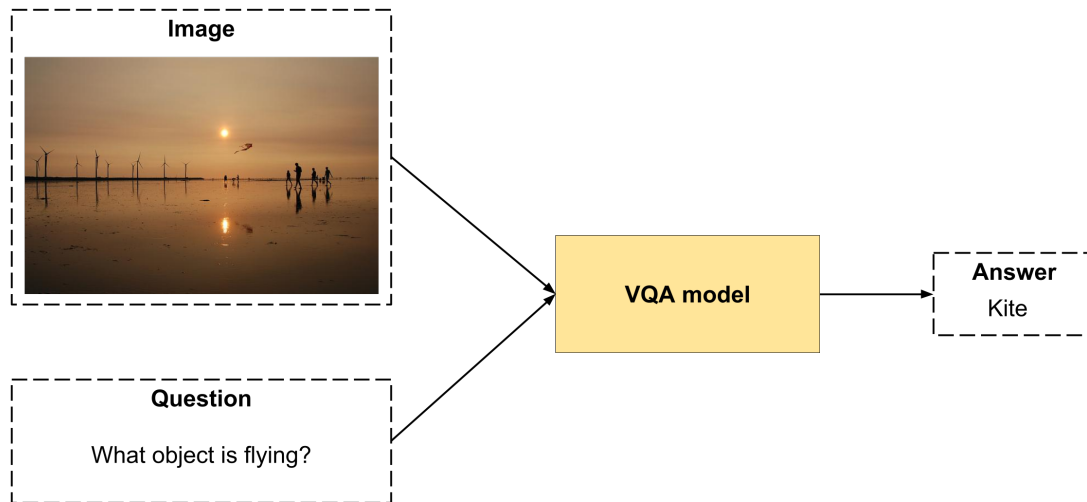
Figure 1.1: Real example of the Visual Question-Answering dataset. The complexity of the task and the required abilities are appreciable in this example where, in order to succeed, the model needs to solve an object retrieval-like task but with the addition of having to understand the scene and the mentions of the question, *e.g.* the relationship between the word 'flying' and the object position

  relations, meaning that you can perform some arithmetic operations with those embedding and the result will have semantic sense

- Try different architectures and parameters to increase model's accuracy

- Develop a reusable software project using programming good practices

- Participate in Visual Question Answering Challenge[1], hosted as workshop in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016.

Regarding the last item, we presented our results to the challenge with an accuracy of 53,62% (details on the results chapter 4 and the model employed will be discussed further on the methodology chapter 3).

We also achieved an additional goal, which was not planned at the beginning of the project. We submitted an extended abstract (you can find it in the appendices 7) to the CVPR16 VQA workshop[2] and it was accepted by the organizers. Due to this fact, we were invited to present our extended abstract and a poster in the workshop.

1.2   Requirements and specifications

One of the main blocks of this project is the software developed to participate in the challenge and to be able to create and test different models.

---

[1]http://www.visualqa.org/challenge.html
[2]http://www.visualqa.org/workshop.html

Regarding with this, the requirements of this project are the following:

- Develop a software that can be used in the future to keep doing research in this field, having a skeleton/base project to start with

- Build a deep neural network model that uses NLP and CV techniques to process the question and the image respectively

- Try different model configurations to increase the accuracy of the original model

- Submit results to the CVPR16 VQA Challenge

The specifications are the following:

- Use Python as a programming language

- Build the project using a deep learning framework. Keras[3] has been chosen as the framework and it can run upon Theano[4] or TensorFlow[5] backends.

1.3 Methods and procedures

This thesis represents the first attempt at solving the Visual Question Answering problem by GPI and TALP research groups at the Universitat Politècnica de Catalunya. We started to develop the project from scratch (in terms of the project itself) but using a deep learning framework called Keras. Keras is a neural network library for Python that has been build to be easy to use and allow fast prototyping. It accomplished this by building a wrapper around another deep learning python library that is in charge of managing tensors and the low level computations. This second library, that works as a *backend*, can be either Theano or TensorFlow. We have run our experiments using Keras over Theano.

Apart from these libraries, the only resource developed by other authors are the visual features of our last model. The Computer Vision group at Universitat de Barcelona provided us with the precomputed visual features of the images from the VQA dataset. They extracted these features using a special kind of CNNs called Kernelized CNN (KCNN) as proposed by Liu [11]. The KCNN method aims to provide a better vectorized representation of images than vanilla CNNs as they have a lack when the image has complex content. This model uses CNNs to extract features and then aggreagate them into a vectorial representation using the Fisher vector model.

1.4 Work Plan

The project was developed as a collaboration between the GPI and TALP research groups of the Universitat Politècnica de Catalunya. Discussions and decisions about the project were held

---

[3]http://keras.io/
[4]http://deeplearning.net/software/theano/
[5]https://www.tensorflow.org/

in a regular weekly meeting, which was complemented with a second research seminar of two hours per week with other students developing their bachelor, master or Phd thesis at GPI.

The following is the workplan of this project and its deviations from the original plan. These deviations are explained in detail in the Incidents and Modifications subsection 1.5.

1.4.1 Work Packages

- WP 1: Project proposal and work plan.

- WP 2: Introduction to deep learning and Python

- WP 3: Introduction to (visual) question-answering tasks and Keras framework

- WP 4: First VQA model

- WP 5: Critical Review of the project

- WP 6: Participate in the CVPR16 VQA Challenge

- WP 7: Final report of the project

- WP 8: Presentation and oral defense

1.4.2 Gantt Diagram

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**WP 1: Project proposal and work plan**	**19 días**	**sáb 30/01/16**	**jue 25/02/16**
2	Project definition	12 días	sáb 30/01/16	lun 15/02/16
3	Project proposal writing	6 días	mar 16/02/16	mar 23/02/16
4	Project proposal revision and approval	2 días	mié 24/02/16	jue 25/02/16
5	**WP 2: Introduction to deep learning and Python**	**78 días**	**vie 15/01/16**	**mar 03/05/16**
6	Coursera's Andrew Ng course on Machine Learning	57 días	vie 15/01/16	sáb 02/04/16
7	Stanford online course: Convolutional Neural Networks for Visual Recognition	22 días	lun 04/04/16	mar 03/05/16
8	Codeacademy Python course	8 días	mar 01/03/16	jue 10/03/16
9	Python code style and best practices	2 días	vie 11/03/16	lun 14/03/16
10	**WP 3: Introduction to (visual) question-answering tasks and Keras framework**	**85 días**	**mar 01/03/16**	**lun 27/06/16**
11	Keras documentation	6 días	lun 14/03/16	dom 20/03/16
12	Build text-based question-answering model	12 días	lun 21/03/16	mar 05/04/16
13	State of the art research	85 días	mar 01/03/16	lun 27/06/16
14	**WP 4: First VQA model**	**23 días**	**mié 06/04/16**	**dom 08/05/16**
15	Define VQA model architecture	6 días	mié 06/04/16	mié 13/04/16
16	VGG-16 on Keras	6 días	mié 06/04/16	mié 13/04/16
17	Implement VQA model	13 días	jue 14/04/16	sáb 30/04/16
18	Train model	6 días	lun 02/05/16	dom 08/05/16
19	**WP 5: Critical review of the project**	**6 días**	**dom 01/05/16**	**lun 09/05/16**
20	Critical review writing	5 días	dom 01/05/16	jue 05/05/16
21	Critical review revision and approval	2 días	vie 06/05/16	lun 09/05/16
22	**WP 6: Participate in the CVPR16 VQA Challenge**	**30 días**	**dom 24/04/16**	**dom 05/06/16**
23	Extended abstract	7 días	dom 24/04/16	dom 01/05/16
24	Try different model parameters	21 días	lun 09/05/16	dom 05/06/16
25	Define different architecture	5 días	vie 27/05/16	jue 02/06/16
26	Implement and train new architecture	2 días	vie 03/06/16	dom 05/06/16
27	**WP 7: Final report of the project**	**12 días**	**vie 10/06/16**	**lun 27/06/16**
28	Final report writing	7 días	vie 10/06/16	dom 19/06/16
29	Final report revision and approval	6 días	lun 20/06/16	lun 27/06/16
30	**WP 8: Presentation and oral defense**	**14 días**	**mar 28/06/16**	**vie 15/07/16**
31	Presentation structure	3 días	mar 28/06/16	jue 30/06/16
32	Prepare presentation	7 días	vie 01/07/16	dom 10/07/16
33	Rehearsal of the presentation	5 días	lun 04/07/16	vie 08/07/16
34	Oral defense	5 días	lun 11/07/16	vie 15/07/16

Figure 1.2: Gantt Diagram of the project

1.5 Incidents and Modifications

During the project we needed to modify some work packages definition and tasks as we wanted to focus more on the Visual Question-Answering Challenge.

Initially, the goal of the project was developing a system capable of generating both questions and answers from an image. This would have medical application in patients with mild cognitive impairment (early stages of Alzheimer), who may receive an automatized reminiscence therapy based on the images captured by egocentric cameras. However, solving the VQA challenge was more feasible in terms of annotated datasets, metrics and potential impact, so it was decided to

address this task first. The described medical applications are planned to be explored by other students during Fall 2016.

We also included the new task of writing an extended abstract for the VQA workshop. We decided to write and submit the extended abstract as this would gave me some expertise on paper composing and this way we could share with the community some of our ideas.

Chapter 2

State of the art

In the past years, multidisciplinary problems of vision, language and reasoning have emerged as a trend in Artificial Intelligence (AI) research. This tasks join Computer Vision (CV), Natural Language Processing (NLP) and Knowledge Representation and Reasoning (KR) to be able to build models that can interact with both image and language input/output. However, this models still fail achieving accuracies close to human level.

Visual Question-Answering has appeared as a problem where models need to be able to perform different sub-problems of the above three fields in order to succeed. To solve this problems the models need a much deeper understanding and comprehension of the scene in the image, what the question is referring to and how the items are related.

We will revise some of the literature involved in the process of building a VQA model, from image and text processing, to the state-of-the-art approaches for VQA tasks.

2.1 Image processing

Deep Convolutional Neural Networks (CNN) have been proved to achieve state-of-the-art results in typical Computer Vision tasks such as image retrieval, object detection and object recognition.

A common approach when dealing with images is to use an off-the-shelf model (VGG [21], AlexNet [10], GoogLeNet [22], etc.) pre-trained to do such tasks with some large image dataset such as ImageNet[1] [4] and use some of the inner-layer's outputs as a representation of the visual features of the image.
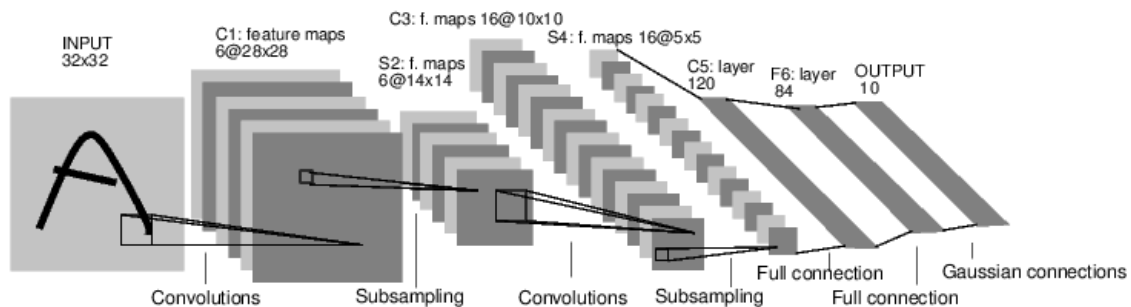

Figure 2.1: LeNet, an example of Convolutional Neutal Network

Typically these models have different types of layers, amongst the most common convolutional layers (that give the name to the CNNs) and fully-connected layers.
The convolutional layers used in image processing perform 2D convolutions of the previous layer

---

[1] http://www.image-net.org/

output (which can be an image) where the weights specify the convolution filter.

In contrast, fully-connected layers take each output from the previous layer and connect them to all of its neurons, losing the spatial information so they can be seen as one dimensional. One of the most common fully-connected layers is the so called softmax layer, which is a regular fully-connected with the softmax as activation function. Its output follows a distribution-like shape, taking values from 0 to 1 and being the addition of all of them equal to 1.

2.2 Text processing

In order to process sequences of text, different approaches are used. For the sake of the simplicity, we will only review two of them that are important for this work.

The first one is the word embedding representation using Skip-gram technique presented by Mikolov *et. al.* [15][16]. This method is used to learn high-quality word vector representations. The input is usually the index of the word in a dictionary (*i.e.* its one-hot code), a vector as large as the size of the vocabulary which is zero-valued except at the index position corresponding to the word. These vectors are high-dimensional (as the dictionary size can have thousands or hundred of thousands of words) and sparse due to the nature of the one-hot representation. The word embedding projects this vector into a semantic space where each word is represented by a dense vector with less dimensions. This technique captures semantic and syntactic relationships between words and also encodes many linguistic patterns based on the context where the words appear. These patterns can be expressed as algebraic operations, *e.g.* embed("King") - embed("Man") + embed("Woman") has as the closest vector the embedding of "Queen".

The logical evolution of this representation is what is called sentence embedding. Word embedding fails at capturing the long-term dependencies between words that appear together in a sentence. To solve this problem, sentence embedding uses Recurrent Neural Networks (RNN) with Long Short-Term Memory cells (LSTM) to increasingly accumulate richer information about the sentence.

Note that RNN are deep neural networks with a memory state that allows them to retain temporal context information, so they take care of the dependence of the current prediction based on the current input and also the past one. LSTM where proposed by Hochreiter *et. al.* to improve the quality of the long-term memory that these models have by means of gating mechanisms that control the flow of information getting in and out of the network. For further details address [6]. The RNN sentence embedding method presented by Palangi *et. al.* [19] takes the one-hot representation for each of the words in the text sequences, obtains its word embedding and then feeds the LSTM with them, one at each timestep, keeping the same order as presented in the sequence. The LSTM will update its state based on this embedding and therefore will be accumulating the information of each word and its own context. At the end of this process the LSTM state will have a condensed representation of the whole sentence.

Such dense representations of sequences of text have also been addressed by Cho *et. al.* [3] for statistical machine translation with GRU cells, which are a similar approach to that of LSTM. They proposed a RNN architecture called Encoder-Decoder. The first stage encodes a sequence into a fixed-length vector representation and the other decodes the vector into another sequence of arbitrary length. The resulting vector after the encoder stage can be used to represent the sentence.

2.3 Visual Question Answering

Visual Question Answering is a novel problem for the computer vision and natural language communities, but is has received a lot of attention thanks to the dataset and metrics released with the VQA challenge, together with the large investments of pioneering tech companies such as Microsoft, Facebook or Sony.

The most common approach is to extract visual features of the image using a pretrained off-the-shelf network and process the question using word embeddings or sentence embedding [24][20][12][8][25][1][26].

Antol *et. al.* [1], the organizers of the VQA challenge and the creators of the VQA dataset, propose as their baseline a model that uses VGG-16 [21] to extract the visual features of the image. They use as the representation the output of the last hidden layer of this model. This features are then l_2 normalized and given to a fully-connected layer to transform the incoming vector into a common space with the question representation. For the question, they use a 2-layer LSTM that takes as the input the word embedding of each question token, timestep by timestep, and when the whole question has been introduced into the LSTM, it outputs its last state as the question embedding. This vector (dimension 2048) is also given to a fully-connected (similarly with what they do with the image) to project it to the same space. Both features are combined using an element-wise multiplication for later use by a fully-connected layer and a softmax that will predict the class answer. Here the 1000 most common answers in the training set have been selected as the classes to predict.

A simple bag-of-words and word embedding model that uses GoogLeNet [22] for the image processing and a concatenation of both visual and textual features is what Zhou *et. al.* [25] present in their paper as a basic approximation to VQA.

A quite different method from the ones presented above is what Noh *et. al.* [18] propose, called Dynamic Parameter Prediction Network (DPPnet). They state that in order to solve VQA tasks, different *networks* need to be used as the model need to perform different tasks depending on the question. To accomplish this, they use the question to predict the weights of one of the layers in the networks, thus changing it at test time for each sample. They take the VGG-16 pretrained with ImageNet model as their starting point. Then, they remove the softmax layer and add three fully-connected layers (the last one being a softmax). They have named 'classification network' to this modified VGGnet. The interesting point is that the second one of those new fully-connected layers is a dynamic parameter layer. This means that at test time, the weights of this layer will be changing from sample to sample. These weights are predicted by a network (parameter prediction network) composed by a Gated Recurrent Unit (GRU) [2] connected to a fully-connected layer. GRU is another kind of RNN similar to LSTM. This layer takes the embedding of each word in the question as its input and when the whole question has passed through the network, its last state is given to a fully-connected layer which predicts the weight candidates for the dynamic parameter layer in the classification network (the VGG-16 based network). To reduce the number of parameters to predict, a hashing function is used to map from the predicted weights of the prediction network to the actual weights of the fully-connected. Figure 2.2 depicts the whole architecture of their model.

Other authors propose attention models to improve the performance of the whole model, stating that most of the questions refer to specific image locations [26][24][12]. In [26] the visual features (the output of fc7 of VGG-16) are treated as if they were the first word in the question,
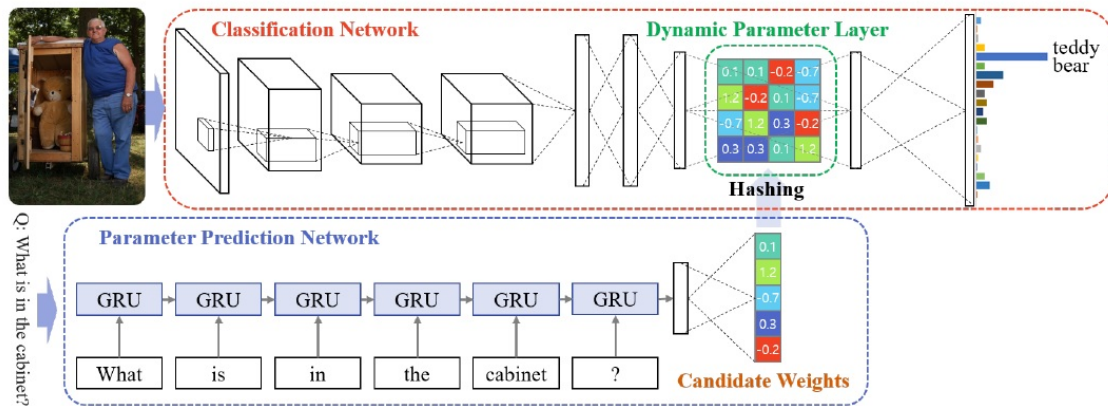
Figure 2.2: DPPnet, the question is used to learn how to predict parameters for a dynamic parameter layer in the classification network.

that is fed into a LSTM word embedding by word embedding. The attention model depends on the LSTM state and is used to weight convolutional features of the image (output of the last conv layer of the VGG-16), that are again introduced in the LSTM merged (using addition) with a word embedding. Ren *et. al.* [20] present a similar but simplified method that also treats the image as the first word of the question but that does not have an attention model. Xiong *et. al.* [24] present a model based on Dynamic Memory Networks (DNM), that is a modular architecture with attention models. They created a new input module to be able to perform VQA tasks apart from text-based QA and improve the memory module. They use bidirectional GRU so that each feature (textual or visual) has a full context representation thus representing local and global information.

All these methods present visual attention models but as proposed by Lu *et. al.* [12], attention in the question can also be applied to increase the model performance. This method, called co-attention together with a hierarchical representation of the question helped them to achieve state-of-the-art accuracy with the VQA dataset (and using their evaluation metric). Recently another method has outperformed their results.

More sophisticated approaches have also been presented, such as Multimodal Residual Learning applied to VQA [8] that uses Deep Residual Learning to build complex and very deep networks. Other works propose learning methods for specific sub-problems of VQA such as human action prediction and then apply those trained models for VQA tasks [14].

At writing time, the model that achieves state-of-the-art accuracy is the one proposed by Fukui *et. al.* [5] which uses Multimodal Compact Bilinear pooling (MCB) to merge the visual features and the information from the question. They hypothesize that the typical merge actions (addition, element-wise product, concat...) do not express correctly all the information. Using MCB to merge those features they achieve an accuracy of 66,2% on the Real Open-ended test-dev dataset. A MCB is also used to create two different attention maps that are concatenated before feeding the main MCB.

Chapter 3

Methodology

This chapter presents the methodology used to develop this project and the process followed to achieve our final results. The baseline for the results that has been taken into account is the one provided by the CVPR16 VQA Challenge.

3.1 A programmer's word

During the first stages of this thesis, when we were looking for some baseline code to perform VQA tasks which we could start with, we found out that the open-sourced projects for VQA were not reusable at all. As a matter of fact, it seems that the vast majority of research code out there has not been developed with programming best practices or with reusability in mind.

That is why we decided to develop our code having in mind some important things: modularity, abstraction, reusability. We intended to apply as many good practices as possible given that we had a restriction in time as we wanted to present our results in the VQA Challenge. As it always happen with software projects, the time variable was crucial in terms of *how much* modular, abstract or reusable was our code at the end.

Nevertheless, we think that the final work is going to be very useful as a starting point for future projects related with VQA and also as a good end-to-end Keras' example. With that we mean that sometimes there is a lack of examples with some degree of complexity that cover the whole process of building a model, training, validating and testing.

3.1.1 Choosing the best language

In terms of actually coding the project we decided to use Python as a programming language. We considered this was the best language to approach this project in terms of prototyping speed and tools available.

C++, Lua and Python were the finalists of this search. C++ was discarded as sometimes is a little bit cumbersome to prototype things fast, for the syntax itself and for the fact that it is a compiled language. Lua and Python have a quite similar syntax, both being a high-level and scripting programming language with a fast learning curve and fast for prototyping. At the beginning, all the open-sourced projects that we found that had something to do with VQA where written in Lua using a deep learning framework called Torch[1]. This seemed a good reason to choose Lua over Python but then, having a look into where the community was going to, we found out that frameworks like Theano or TensorFlow were having great success and the developers and research community was moving towards them. Both frameworks are for Python, which made us choose Python as a programming language for the project. Then, we were recommended to use Keras, a library able to work upon Theano or TensorFlow to expedite the prototyping process.

---

[1]`http://torch.ch/`

3.1.2 A Pythonic project

Having chosen Python as the programming language, one of the things we wanted to do to increase the readability and reusability of this project was to follow a code style guide.

In the programming world there are many languages and for everyone of them there are tons and tons of styles that the developers tend to program with, and we programmers are picky. That is why the most popular languages usually have a code style guide that define *how the code should look* and *what is a good practice* in that language. Using these code style guidelines increases the readability of the code and helps you to develop a better code that will be more easily extended or used.

For Python, this code style guideline is called PEP8[23]. The code presented with this project follows the PEP8 guideline.

3.1.3 An eight-leg cat called octocat

To develop the project as professional as possible and to keep track of the changes we made we have used Git[4] as a Version Control System (VCS). Git give us the ability to work in parallel when needed and to prototype things without the fear of not being able to restore our previous work or having to do all those annoying manual backups. Using Git we have created an historic of our project development process.

To store our Git repository we have used GitHub as it allows us to open-source the project once finished and to enable community contributions. After the VQA challenge deadline we published our GitHub repository[5] as public so everyone can use the code.

3.2 Dataset

In order to train a model (in supervised learning) we need a very large amount of data. This data are example of input-output pairs. In our case, the input are both the image and the question and the output is the answer.

To train our models we have used the real image VQA dataset[6], which is one of the largest visual question-answering datasets. This dataset is provided by the organizers of the VQA Challenge and is splitted in the typical three subsets: train, validation and test. The train subset is composed by 82.783 images, 248.349 questions and 2.483.490 answers; the validation by 40.504 images, 121.512 questions and 1.215.120 answers; and finally the test set is composed of 81.434 images and 244.302 questions. The whole explanation on how the organizers created this dataset can be found in their paper [1].

All the images are part of the Microsoft Common Objects in Context (MS COCO) image

---

[2]https://www.python.org/dev/peps/pep-0008/
[3]There are different PEP guidelines, each one of them dictating best practices for different tasks
[4]https://git-scm.com/
[5]https://github.com/imatge-upc/vqa-2016-cvprw
[6]http://www.visualqa.org/download.html

dataset[7] and the questions and answers have been generated by different workers. MS COCO was chosen as the image are very different (size, color and black&white, quality...) and they are rich in contextual information. The questions are of different types, being the most common the 'what...?', 'is...?', 'how...?' and 'are...?'.

As it is usual, the train subset has been used to learn the model parameters, at the same time that the validation set was used to check on the model's generalization to unseen data. By using this information, we could improve some parameters and present the ones that achieved higher accuracy on the validation set. The test set does not have answers as it defines the problem to be solved during the challenge. The answers predicted for the test set were the ones that we submitted to the VQA 2016 challenge.

3.3   Text-based QA toy example

As we already stated in our work plan 1.4, we started familiarizing with VQA tasks and how Keras library works through a text-based QA model.

The kind of text-based QA problem that we addressed was a toy example were a short story and a question related to that story is given to the model so it can predict a single word answer.

3.3.1   Tokenization

The first step to take is transforming the words (from the story and the question) into numbers that can be feed into the model. We did this preprocessing with a tokenizer provided by Keras that is in charge of tokenizing the text sequences. By tokenizing here we mean splitting the whole string into words, remove the unnecessary ones (punctuation for example) and transform each word into a number. This number is the index of that word in a dictionary that we created previously. The dictionary or vocabulary of our tokenizer can be a predefined one or not. We did not use a predefined dictionary but created our own one using the training data.
To create such a dictionary, its size is important, the number of unique words that it can include. A special word will also be included, which is the one representing 'unknown' words, *i.e.* words that are not in the previous list.
From this point on, a word is no longer a string such as 'garden' but a number representing its position in the dictionary.

3.3.2   Model architecture

Once we have tokenized the story and the question, their representation is a list of numbers. These numbers are the input of our model. The architecture of this model is represented in Figure 3.1. The model has two branches, one for the story and the other one for the question, that are merged together to produce the output.

This model has an encoder structure (see Cho *et. al.* [3] for a complete encoder-decoder architecture for automatic machine translation). We take the input, which is high dimensional
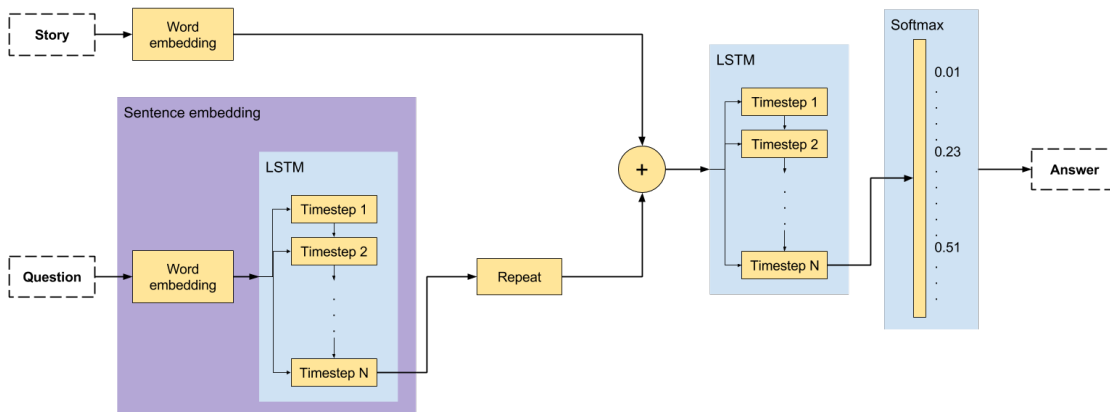
---

[7] http://mscoco.org/

Figure 3.1: Schema of the model used for text-based QA

(the size of the vocabulary, a typical value is 20.000) and we encode it into a much smaller representation, a vector in a continuous space of a dimension that we have to specify beforehand (it is fixed and it is an hyper-parameter that we have to adjust manually), for example 100. This single vector holding the whole information of the question and the story is our encoding. This encoded vector is given to the fully-connected layer, a softmax, that will predict the one-hot representation of the answer. A one-hot representation is simply a vector with all zeros and just a one in a specific location. In our case, this vector has dimension equal to the vocabulary size and the one is placed in the location equal to the word's index in the dictionary. The softmax will not predict 0 and 1 but a distribution between 0-1. We took the maximum value as our output.

Lets dig a little bit deeper into the encoder. The story branch has only a word embedding block [16]. The word embedding learns how to do a dense representation of the input word as a vector in a continuous space of the specified dimension. This reduces the dimensionality of our input as it is projected into a space with less dimensions. Such space has a very interesting property which is one of the reasons that we use these embeddings. The vectors in that space are not only dense but they are also a semantic representation of the word. One possible example of this is that the embeddings of words with similar meaning are close to each other (the distance between the vectors is small). After the word embedding we will have a sequence of vectors representing the story, a vector for each word.

The question branch is a little bit more complex. We start with the same word embedding than the story. The output of that block is then given to a LSTM [6], which is a Recurrent Neural Network (RNN). RNNs have the advantage of having memory over *time*, *i.e.* they have a state that is kept in memory and it is updated in each iteration and their output is somehow dependent of this state. LSTMs are widely used to process sequences for these reasons. Specifically, we are using a non-stateful LSTM that means that this state is not preserved from batch to batch, it is resetted. We have also configured the LSTM so it only outputs its last state. We set the maximum sequence length to the LSTM so it knows when the question has finished and it can output a value.

The output of the LSTM is a representation in a single vector of the whole question. This vector is then given to a block that repeats the vector as many times as specified, in our case, the maximum story length (in tokens). This combination of the word embedding and a LSTM that sees all the question words and then outputs its memory state is known as a sentence embedding [19].

This way, at the merge stage there is a sequence of word embeddings from the story branch, and a sequence of the question embedding repeated. Each iteration in the sequence is what we call a *timestep*. That being said, at each timestep we are summing up the embedding of a story word and the embedding of the whole question. To be able to do so, both vectors need to have the same dimension and that forces that both word embeddings (story and question) must have the same dimension as hidden units in the LSTM that encodes the question (which determines the output dimension of it).

The result of adding both embeddings is given to another LSTM which is in charge of the last encoding. This LSTM is also non-stateful and it will accumulate all the merged features until it has seen the whole story and then it will output its state. This last vector is our encoding of the whole story and question merged and it is what we use to predict the answer, as we have explained before.

As an addition, this model also uses drop outs with a 0,3 value to prevent overfitting.

3.3.3 Model up and running

Once we had this model built, we trained it with 10.000 sample for the QA1 task defined by Weston *et. al* [23] and we did some small tests. We did not run extensive experimentation at this stage as this was not our objective and because the parameters of the model and the dataset we used were designed more as a toy example than a real-world solution. This stage allowed us to train the model as fast as possible and check that the whole process was working.

3.4 Extending the text-based QA model for visual QA

Taking as a starting point the previous text-based QA model, we modified it so it could be used for visual QA. Notice that the architecture shown in Figure 3.1 has been built around the idea that we have a story that gives some information to the model and then we ask a question about that story. The model uses the information retrieved from the story to be able to answer the question. In visual QA our story is the image, is what give us the information needed to answer the question.

3.4.1 Image processing

With the idea of our image being the "story" from which we have to extract information, we changed the story branch for an image branch. In such a branch we use the VGG-16 convolutional neural network proposed by Simonyan *et. al.* [21], an off-the-shelf model, to extract the visual features, as you can see in Figure 3.2. We did not use the output of the whole model but we truncated it until the last convolutional layer, before the fully-connected fc-4096. Using the output of the conv layers instead of the fully-connected ones is a common practice to extract visual features maps.

In order to be able to combine this visual information with the one obtained from the question, we need to turn this 2D map into a vector. We used a Flatten layer to do so and then we give
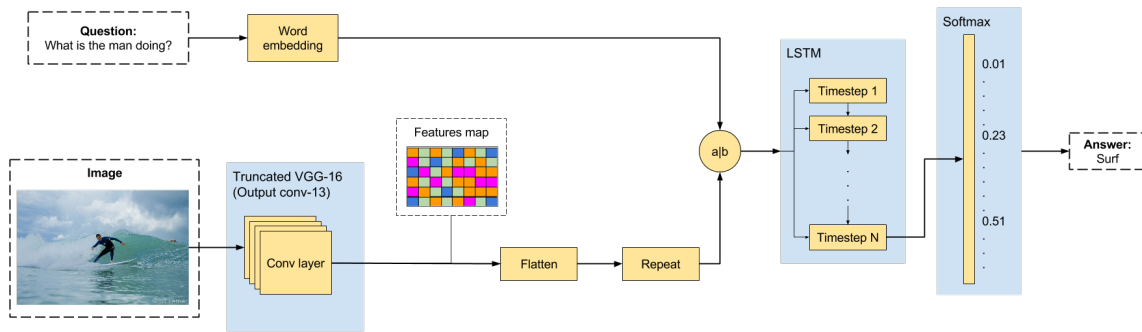
Figure 3.2: First visual QA model's schema

this vector to the repeat block. Notice that now we are repeating the image (our visual story) instead of the question. We are doing this as the question is our only sequence in this model and this way the model will *see* the whole image for each question word.

3.4.2 Modifications on the question branch

As shown in Figure 3.2, the question branch has only a word embedding now. This means that in each timestep a question word will be feed into the model and, because the visual features are repeated, each one will be merged with the information of the whole image. The dimension of the word embedding and the visual features is different so our merge process now is not a summation but a concatenation of both vectors.

3.4.3 Model parameters

The important parameters of this model are: vocabulary size, LSTM hidden units, embedding size, question maximum length, learning rate and batch size. It is also important which optimizer to use.

We set the batch size to be the maximum that we could save in the GPU RAM, having a value of just 32 samples. We need to consider that we also need to fit the compiled model (its weights) in the GPU RAM and this is very expensive as some of our layers, and thus its weights, are huge as we will see now.

The learning rate for the network's parameter was governed by the Adam optimizer [9] which modifies the learning rate through the training. We only need to specify the starting learning rate, and we chose the default one proposed by Kingma *et. al.* in the original paper which is 0,001.

For the question maximum length we have taken the length of the largest question in the training subset. This parameter is used in the last LSTM so it knows when it has seen the whole question and can output its state. We found that for the training set, the maximum question length is 22. The questions that have a smaller length have been left-padded with 0, so the input is 'inactive' and then it is activated with the question tokens. The network has been configured to ignore these padding zeros.

The vocabulary size is crucial for the softmax layer as this will set the number of neurons of this layer. A value of 20.000 was chosen as it is quite common and respects the tradeoff between number of words (which give more flexibility to the model) and number of weights to train (time consuming, training problems, memory constraints).

For this model we chose the number of LSTM hidden units and the embedding size to be the same, with a value of 100. We used this value for simplicity and due to some experience of a team member regarding these parameters.

We have also changed the dropout rate from 0,3 to 0,5.

3.4.4   Implementation and training

We built the whole model including the truncated VGG-16 and we used pretrained weights for this module, that we froze at training time. This weights were the result of training the whole VGG-16 on ImageNet[8] one of the biggest image datasets existing nowadays. As we freeze the VGG-16 weights we did not make a fine-tunning of it, we only trained our own layers.

To train this model we started making use of the computational service of the Image Processing Group (GPI) at the Universitat Politecnica de Catalunya. Some memory constraints were faced as the size of the vocabulary imposed the need to create huge vectors representing the answers and the size of the compiled model reflected this too. As we could only fit 32 samples per batch, the training process was at a rate of 17-24 hours per epoch using NVidia Titan X GPUs, equipped with 12 GB of RAM. This forced an evolution to the next model as having this model train for a reasonable number of epochs (40) was not a valid option. Notice than an epoch is defined as a single pass of all the examples in the training set through the model under training.

In terms of software, we created an Image, Question, Answer, VQASample and VQADataset Python classes to hold the information of these entities and to allow single responsability and modularity of the code. Using these classes we encapsulate the logic in modules that we can easily control, instead of working with plain data such as Python dictionaries, lists, numpy arrays, etc.

3.5   Model improvement: towards the final model

The prohibitive duration of the training process made us opt for precomputing the visual features of the image. This approach made sense as we were not modifying the values of the VGG-16 convolutional network that was in charge of extracting these features.

Instead of precomputing ourselves the visual features using an isolated VGG-16, our partners from the Computer Vision group at the Universitat de Barcelona (UB) provided us with these features extracted with a new kind of CNN called Kernelized CNN (Liu *et. al.* [11]). You can find a short description in the Methods and procedures section 1.3. The dimension of the output vector from the KCNN module is 1024. The rest of the parameters and functionality remains the same as in the architecture described in Section 3.4.
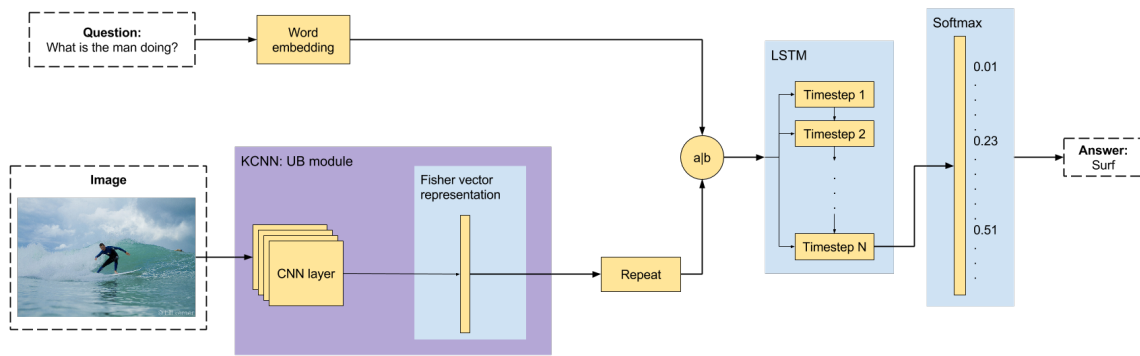
---

[8]http://www.image-net.org/

Figure 3.3: Replacement of the VGG-16 by the KCNN model

3.5.1 Implementation and training

We also changed the way we programmed the model. In the previous case, we were using an abstract model implementation of Keras called Sequential[9] which is basically a stack of layers. This model also allows the possibility of merging two sequential models into one, that is what we used to create the two input branches. For this modified model we changed to the more flexible Functional API[10] which is thought to build more powerful models in a graph approximation. This new interface let us work with the tensors themselves so it is now easier to modify and make the model more complex.

By using the precomputed visual features and this new implementation, we reduced the training time of an epoch to less than an hour (40 minutes approximately).

3.5.2 Batch normalization and reducing the learning rate

One of the first modifications we tried was adding a batch normalization layer (Figure 3.4) after the merging process, as this helps the training process and usually increases the accuracy. Ioffe and Szegedy propose to introduce the normalization of the layers' input distribution inside the model architecture [7]. They introduce this normalization using their novel layer (batch normalization) which reduces the internal covariate shift.

We also reduced the initial learning rate sequentially from 0,001 to 0,0003 and to 0,0001 and we found that the last one was giving the best accuracy as we will explain later in the results chapter 4.

3.6 The final model: sentence embedding

Our last model was the one that predicted the answers with higher accuracy and presented to the VQA challenge. Several changes were introduced with respect to the preliminary prototypes so lets have a look into the different blocks, depicted in Figure 3.5.

---

[9]http://keras.io/getting-started/sequential-model-guide/
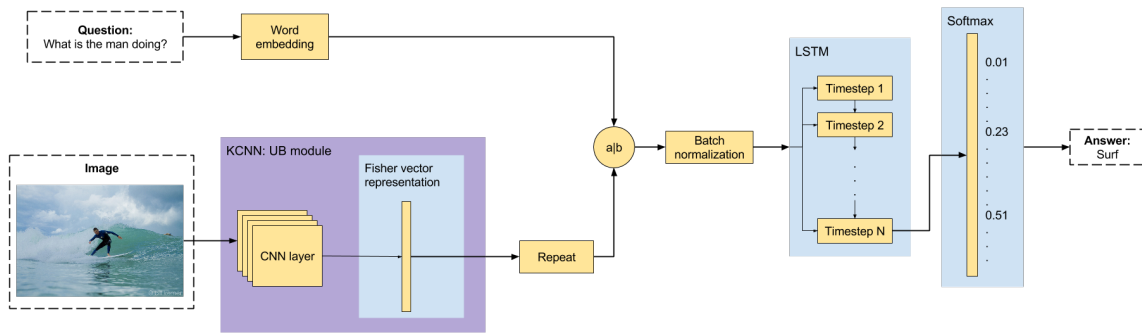[10]http://keras.io/getting-started/functional-api-guide/

Figure 3.4: Batch normalization added to help the training process

3.6.1 Question embedding

The question branch was modified by adding a LSTM at the end of the word embedding, thus creating a sentence embedding, in our case the question embedding. The resulting vector of the sentence embedding module is a dense and semantic representation of the whole question as it was in our text-based QA model 3.1. The difference here is that we did not choose the same value for the word embedding dimension and the number of LSTM hidden units. We set 100 as the word embedding dimension and 256 as the number of LSTM hidden units, which is a common value. We increased the number of hidden units as this can help increasing the accuracy in the condensed representation of the questions but we did not change the embedding dimension as this could decrease the density of the word embedding representation.

3.6.2 Image semantic projection

We decided to add a fully-connected layer after the KCNN module to be able to project the visual features into a space of the same dimension as the question embedding. The fully-connected layer can be seen as a matrix operation which projects the features' 1024-vector into a 256-vector in the semantic space. We have chose ReLU as the activation function for this layer.

3.6.3 Merging and predicting

As both textual and visual features were projected into a 256-dimensional space, we can sum up them together and merge these features. Now that both the question and image are represented by a single vector and not by a sequence of vectors, there is no need to add a LSTM after the merge and we can feed the resulting merged vector to the softmax so it can predict the answer.

3.6.4 Other modifications

The learning rate of this model was initialized to 0,0001 against the 0,001 of the first KCNN model. We also tried to reduce the learning rate to 0,00001 and to add a batch normalization stage after the merging process but as we will see in the following chapter, neither of those

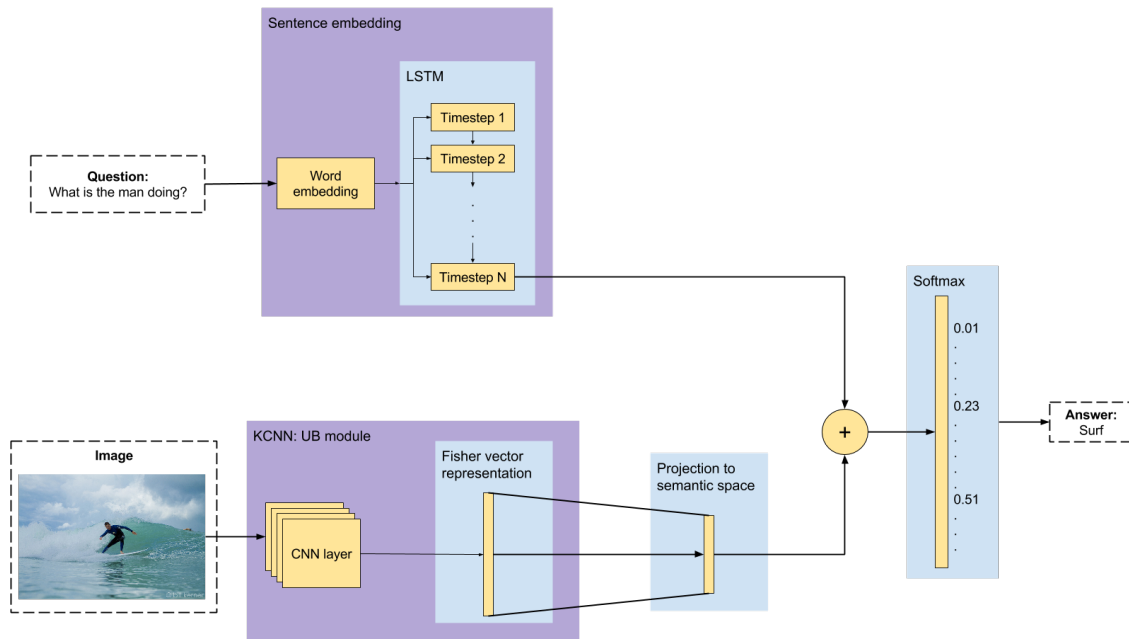Figure 3.5: Final model. A sentence embedding is used for the question and the visual features are projected into the same semantic space than the question

increased the accuracy of the original final model.

Before submitting to the VQA challenge over the test set, we also tried to train the model with the whole training subset and the 70% of the validation subset but this did not help either.

Chapter 4

Results

In this chapter, the results of the different models exposed in the Methodology chapter 3 will be presented.

4.1 Evaluation metric

The models have been evaluated using the metric introduced by the VQA challenge organizers. As they state in their evaluation page[1], this new metric is robust to inter-human variability in phrasing the answers.

The new accuracy formula per answer is the following:

$$Acc(ans) = min\left(\frac{\#\text{humans that said ans}}{3}, 1\right) \qquad (4.1)$$

The accuracy over the whole dataset is an average of the accuracy per answer for all the samples.

The interpretation of equation 4.1 is as follows: an answer is given as correct (accuracy equals 1) if the same exact answer was given by at least three human annotators. Zero matches equals zero accuracy and from there each match gives 0,33 points to the accuracy with a maximum of 1.

4.2 Dataset

At this point is worth summarizing the dataset characteristics mentioned in 3.2:

- **Training dataset**: 82.783 images, 248.349 questions, 2.483.490 answers

- **Validation dataset**: 40.504 images, 121.512 questions, 1.215.120 answers

- **Test dataset**: 81.434 images, 244.302 questions

Notice that for each image there are three questions and for each question there are ten answers. These ten answers were provided by human annotators and the most frequent ones were selected. Most of the answers are the same but rephrased.

The organizers also provide a Python script to evaluate the results[2] the same way that they do when you submit the test results. This script preprocess the answers to avoid format-like

---

[1]http://www.visualqa.org/evaluation.html
[2]https://github.com/VT-vision-lab/VQA/

mismatches. As an example, they make all characters lowercase, remove articles, convert number words to digits...

This script needs a specific JSON file with the ground truth answers and another one with the machine generated answers (what the model has predicted) in a predefined format. We used this script to perform an evaluation of our model over the validation set (because we do have the answers for this subset).

4.3   Models architectures and setups

In the following section we will refer to the models by a number in order to be more clear and concise. These identifiers are defined here with a description of the model/configuration:

Identifier	Description
0	First VQA model. Uses VGG-16 to extract visual features. Described in 3.4 and schema in Figure 3.2
1	Improvement of model 0 using KCNN. Described in 3.5 and schema in Figure 3.3
2	Model 1 using with batch normalization. Schema in Figure 3.4
3	Model 1 with a learning rate of 1/10 the original (0,0001)
4	Final model using sentence embedding for the question and a projection to the semantic space for the image. Described in section 3.6 and schema in Figure 3.5
5	Model 4 using batch normalization

Table 4.1: Models identifiers and descriptions

Results for model 0 will not be presented as we only completed the building stage but we did not finish the training process for the problems already explained in Chapter 3. We only include it here to state that this was our base VQA model.

4.4   Training and validation losses

One of the earlier results that helped us to improve our models was the training, and most important, the validation loss. In the following figures you can see the evolution of the training and validation loss per epoch

UNIVERSITAT POLITÈCNICA
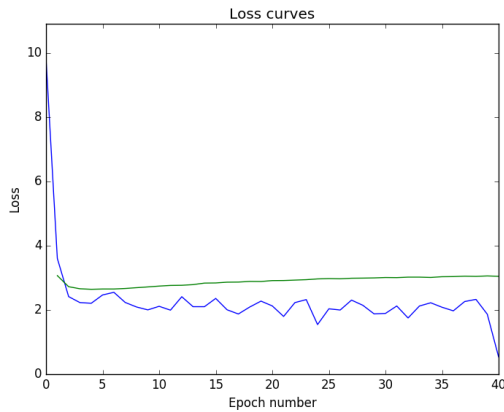DE CATALUNYA
BARCELONATECH
UPC

telecom
BCN

Figure 4.1: Training losses (blue) and validation losses (green) for model 1
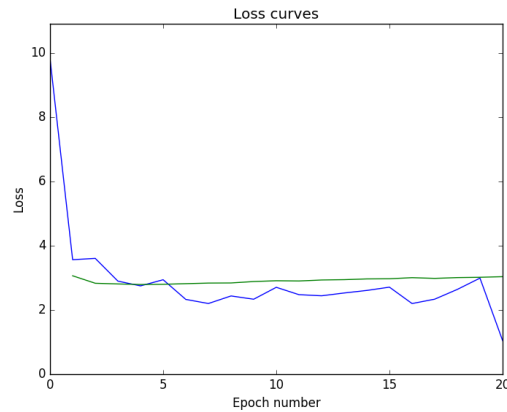
Figure 4.2: Training losses (blue) and validation losses (green) for model 2

For model 1 and 2 we have that the validation loss increases from epoch 3 until the end. We can also appreciate that after the second epoch the model is not learning anymore, the training loss gets stucked around a fixed value with some "noise" and also that in the first epoch the model experiments a huge decrement in the training loss. Both factors are an indicator that the models are slowly diverging and thus the learning rate is too high.
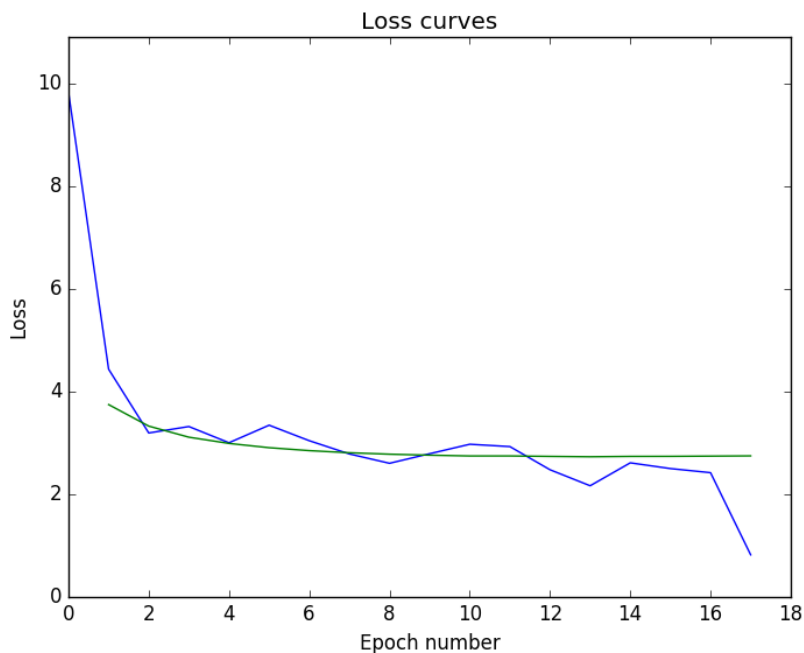
Figure 4.3: Training losses (blue) and validation losses (green) for model 3

In the model 3 we decrease the learning rate to 1/10 of the original one, having a value of 0,0001. As we can easily see in the plot in Figure 4.3 we experiment a slower decrease in the training loss and it does not stop learning after the first epochs. Even if the average training loss is higher than in the previous models, the validation loss (which is the one that helps us measure how good our model generalizes to unseen data), is lower in this model and does not increase

33

over the iterations.

Changing to a sentence embedding and projecting the visual features to the same space than the question reduced the validation loss. Having a look at Figure 4.4 we can see that the validation loss slowly decreases epoch after epoch and it reaches the lowest value of the past models.

Adding a batch normalization layer did not help us in order to obtain better results with the model 5 (Figure 4.5).

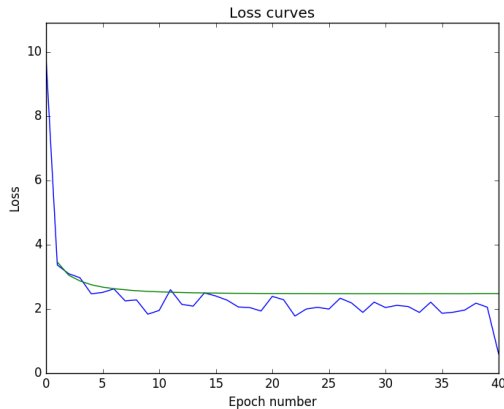Figure 4.4: Training losses (blue) and validation losses (green) for model 4
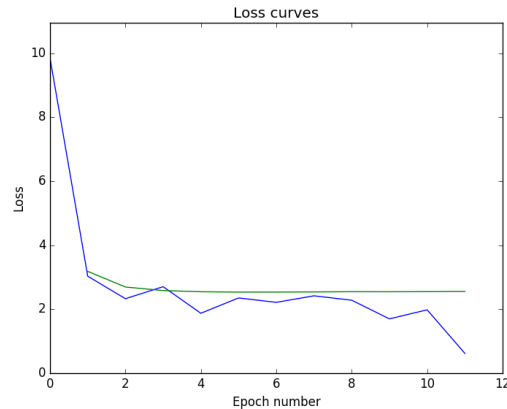
Figure 4.5: Training losses (blue) and validation losses (green) for model 5

4.5 Quantitative results in the VQA Challenge 2016

The model we presented to the CVPR16 VQA Challenge was the model number 4. We get an accuracy of **53,62%** over the test dataset. In table 4.2 we present a comparison between our accuracy and the accuracy of the baseline model and the top one.

As we did not submit all the results from the different models, we do not have test accuracies for some of them (model 2 and 3).

4.5.1 A general overview

The first interpretation of these results is that the gap between the accuracy of the baseline model and the best one (UC Berkeley & Sony) is quite small, only a 12,41%. What this means is that it is very hard to create models good at solving visual QA as the model needs to have a deep understanding of the scene and the question and also quite good reasoning abilities.

Another fact to notice is that there is a performance difference between humans and models performing such tasks, and that means that there is still space for further research in this area. Related with this, it is worth mentioning that human accuracy using this metric is quite low, comparing with what one would expect it to be (close to 1). This may imply that the metric used to evaluate this tasks may not be the best one to use as it does not reflect correctly the

Model	Validation set				Test set			
	Yes/No	Number	Other	Overall	Yes/No	Number	Other	Overall
Baseline All yes	-	-	-	-	70,97	0,36	1,21	29,88
Baseline Prior per question type	-	-	-	-	71,40	34,90	8,96	37,47
Baseline Nearest neighbor	-	-	-	-	71,89	24,23	22,10	42,85
Baseline LSTM&CNN	-	-	-	-	79,01	35,55	36,80	54,06
UC Berkeley & Sony	-	-	-	-	83,24	39,47	58,00	66,47
Model 1	71,82	23,79	27,99	43,87	71,62	28,76	29,32	46,70
Model 3	75,02	28,60	29,30	46,32	-	-	-	-
Model 2	75,62	31,81	28,11	46,36	-	-	-	-
Model 5	78,15	32,79	33,91	50,32	78,15	36,20	35,26	53,03
Model 4	**78,73**	**32,82**	**35,5**	**51,34**	**78,02**	**35,68**	**36,54**	**53,62**

Table 4.2: Results of the four baselines provided by the VQA challenge organizer, the state-of-the-art (UC Berkeley & Sony) and our five models. Model 4 was the one submitted to the challenge leaderboard as the results of our team

performance in such tasks. This could also be a problem on how the dataset is built. If we check the human accuracy using the metric and dataset presented by Zhu *et. al.* [26] we can see that it is 96%, much more logical a priori.

4.5.2 Our results

Now, if we evaluate our results we can see in table **??** that our model performs slightly worse than the baseline provided by the VQA challenge organizers. This has a reason underneath it.

The first reason is that our model predicts only single word answers. This means that we will not have a 100% accuracy in multi word answers as we will never have a complete match. It is true that the VQA dataset's answers are mostly single word but it is also true that we already start with fewer accuracy due to this fact. The VQA dataset answer average length is 1,1 word with a deviation of 0,4.

The second and most important reason is that the baseline model and many of other models presented in the challenge ([12], [18], [13], [5], [8], [25], [1]), including the top ones, are a classifier built upon the N (in their case 1000) most frequent answers. This means that they were taking the 1000 answers from the training set that appear more frequently and assign a class label to each one of them. Then, they train their model to learn how to predict which one of these classes is the answer. At test time they predict a class label that it is matched with a predefined answer. Notice that their classes are the whole answer, not words. What this implies is that the model can only predict some of the answers that it has already seen in the training subset but it can not generate new ones, thus being very limited to this specific dataset.

In contrast, our model was built with the idea of being able to generate any word of the vocabulary as an answer, even if during training time that word was never used as an answer.

We accomplished that by having the model output a word instead of a predefined answer. As our model has an encoder structure, we could also attach at the end a decoder stage to predict a multi word answer with a generative language model (future work 6).

We decided to use this kind of model knowing that our accuracy was going to be lower as we thought that our model was more innovative and more capable of being applied in *real life*, meaning that we did not improved our model towards the VQA challenge or VQA datasets but to Visual Question-Answering tasks in general and to our ambitious goal (which is out of the scope of this thesis but is our future objective) of generating question-answer pairs from images. To do so, we certainly need a model able to answer with unseen answers from the training subset, to generate them. We believe that our model outperforms other participants of the challenge in flexibility and in interest from the research point of view.

4.5.3 Per answer type results

The VQA dataset annotations (answers) are classified in three different types: yes/no, number or other. Each question has assigned one of these three answer types, that allows us to better understand how our model acts given different types of questions and how good is it answering them.

Analyzing the results per answer type shown in Table 4.2 we can see a huge difference when it comes to accuracy between the yes/no answers and the number or other answer types. The latest usually need a higher comprehension of the image and the question to be able to answer them due to the type of questions (why...?, what is...?) as opposed to the more common question type *is this...?* for the yes/no answer type. These difference can be better understand with the qualitative results in the following section.

4.6 Qualitative results

In this section we will present some examples of the results obtained for our best model. These results are from the validation subset, as we do not have the answers for the test subset. The following examples are grouped by accuracy, having three examples of each accuracy, one per question type (yes/no, number and other). The VQA evaluation script punctuate with 5 different accuracies (0, 30, 60, 90, 100) following equation[3] 4.1.

These examples have been chosen randomly from the results in order to obtain a sample as representative as possible of the whole dataset.

---

[3]Even if the challenge provide the mentioned formula to compute the accuracy, it seems that the script is rounding the result of $(\#humansthatsaidans)/3$ to the closest lower integer

Question: Is this a healthy breakfast?

Prediction: No

Ground truth:
"yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes"

Accuracy: 0

Question: What time does the clock show?

Prediction: 11

Ground truth:
8:26", "8:26", "8:26", "8:26", "8:26", "8:26", "8:26", "08:26", "8:27", "8:26"

Accuracy: 0

Question: What is bobbing in the water other than the boats?

Prediction: Boats

Ground truth:
"buoys", "buoys", "moorings", "bouiees", "buoys", "balls", "buoys", "buoys", "balloons", "buoys"

Accuracy: 0

Figure 4.6: Result examples with accuracy 0

Question: Is the train getting ready to leave?

Prediction: No

Ground truth:
"yes", "yes", "yes", "yes", "yes", "no", "yes", "yes", "yes", "yes"

Accuracy: 30

Question: What is the speed limit?

Prediction: 20

Ground truth:
"20 km/h", "20 km/h", "20 km/hr", "20 km/h", "20", "20 km/h", "10 km/h", "20 km/h", "20 km/h", "20 km/h"

Accuracy: 30

Question: What is in the box?

Prediction: Food

Ground truth:
"donuts", "pastries", "food", "donuts", "donuts", "donuts", "donuts", "donuts", "donuts", "doughnuts"

Accuracy: 30

Figure 4.7: Result example with accuracy 30

Question: Is the game sponsored?

Prediction: Yes

Ground truth:
"unknown", "no", "no", "no", "no", "no", "yes", "yes", "no", "no"
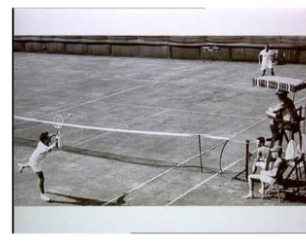
Accuracy: 60

Question: How many horses are covered with blankets?

Prediction: 2

Ground truth:
"3", "3", "2", "3", "3", "2", "1", "3", "3", "3"

Accuracy: 60

Question: What is the role of the people who are sitting?

Prediction: Umpire

Ground truth:
"umpire", "judge", "retrieve balls", "umpire, line judge", "umpire", "fergie", "judge", "judges", "judges", "umpires"

Accuracy: 60

Figure 4.8: Result example with accuracy 60

Question: Are these cats playing?

Prediction: Yes

Ground truth:
"no", "no", "no", "yes", "no", "no", "yes", "no", "no", "yes"

Accuracy: 90

Question: How many lanes are there?

Prediction: 3

Ground truth:
"3", "3", "2", "4", "4", "4", "4", "2", "2", "3"

Accuracy: 90

Question: What is on the shelf in the background?

Prediction: Books

Ground truth:
"books", "television", "books", "videos", "movies", "vhs tapes", "videos", "tapes", "books", "videos"

Accuracy: 90

Figure 4.9: Result example with accuracy 90

Question: Are these women proud of their donuts?

Prediction: Yes

Ground truth:
"yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes"

Accuracy: 100

Question: How many tags are on the suitcase?

Prediction: 3

Ground truth:
"3", "3", "3", "3", "3", "3", "3", "2", "3", "3"

Accuracy: 100

Question: What is in the mug?

Prediction: Coffe

Ground truth:
"coffee", "coffee", "coffee", "coffee", "coffee", "coffee", "coffee", "coffee", "coffee", "coffee"

Accuracy: 100

Figure 4.10: Result example with accuracy 100

38

From these examples we can see that the images in the VQA dataset (that are MS COCO) are rich in information and very variate throughout the dataset. The orientation, ratio, resolution and number of channels vary from example to example, as well as the kind of content appearing.

The questions are very different in terms of what task does the model need to do in order to answer the question. These questions show perfectly the deep understanding of both the image and the question and how they are related needed to answer them. Different tasks need to be performed in order to succeed, such as sentiment analysis (Figure 4.10, first example), object recognition (Figure 4.8, second example and Figure 4.7, second example), Optical Character Recognition (OCR) (Figure 4.6, second example), activity recognition (Figure 4.9, first example) and so on.

As for the answers, we can appreciate why the metric provided by the challenge maybe it is not the best one to use in this task. The second example of Figure 4.7 and the last one in Figure 4.8 show that the predicted answer was indeed correct, but due to rephrasing and some details, only 30 and 60 of accuracy was given to them. The annotation errors also distorts the results, as in Figure 4.8 second example, where the correct answer is 3 and, even if we predicted 2, the script evaluated our answer with 60.

4.7   Some words at the VQA Challenge

At date of 26 of June, 2016, some of our team's members assist to the Computer Vision and Pattern Recognition 2016 at Las Vegas, USA, to present our extended abstract "Towards Automatic Generation of Question Answer Pairs from Images" in the Visual Question-Answering workshop. In this workshop several speakers explained their research in the VQA field, including the winner of the VQA challenge.

We want to highlight some remarkable comments from the last session.
Margaret Mitchell, from Microsoft Research, mentioned the interest of generating questions and answers from image as an extension to VQA. Mostafazadeh *et. al.* (including Mitchell) have recently published a paper where they propose a model to generate natural questions [17] (we presented our extended abstract before this paper was published).
Another interesting comment, coming from Trevor Darrell (UC Berkeley), was his concern about solving VQA with a closed set of answers. This is building a classifier upon the most common seen answers in the training set (which is what a lot of the participants did) as opposite of our model which generates new answers even if the model have not seen them before in training time.

These comments supports the route we have taken to accomplish this thesis.

Chapter 5

Budget

This project is a research study and it has not been developed with a product or service in mind that could be sold in the marketplace. We have used the computational resources provided by the *Grup de Processat d'Imatge* of UPC, so there has not been any additional cost in terms of hardware.

The hardware resources needed for this project were a CPU and a GPU with at least 12GB of GPU RAM and over 50GB of regular RAM. To be able to estimate the cost of the hardware in this project we will use the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) service as they offer cloud computing resources per hour and they are a common solution for this needs. The EC2 instance more similar to our specifications is the g2.8xlarge which provides 60GB of RAM and 4 GPUs with 4GB of RAM each one. The cost of this service is $2,808 per hour which is $67,40 per day. We spend 60 days approximately using the computing resources, thus giving an approximate cost of $4.043,52 for the hardware needs.

Regarding software, everything we have used is open-source and thus this does not add any cost.

Being said that, the only *real* cost we can deduce from this project could be the salary of the team involved in developing it. Basically three members have formed this team: a senior engineer as the advisor, a junior engineer as the co-advisor and myself as a junior engineer.

As presented in the workplan's Gantt 1.2 the total duration of the project has been 24 weeks but the first weeks of work were only personal research. The other difference in the number of weeks is due to the fact that the co-advisor joined the project after few weeks.

	Weeks	Wage/hour	Dedication	Total
Junior engineer	24	10,00 €/h	25 h/week	6,000 €
Junior engineer	16	10,00 €/h	4 h/week	640 €
Senior engineer	20	20,00 €/h	4 h/week	1.600 €
			Total	8.240 €

Table 5.1: Budget of the project

Chapter 6

Conclusions

When we started this thesis we had three main goals in mind. The first one was to be able to build a model for VQA and present our results to the CVPR16 VQA Challenge. The second was to, through the process of building the VQA model, have a better understanding of the techniques used to process text in the deep learning framework. We also wanted to explore how to combine text and visual features together. Our last goal was to build the software around this project as modular, reusable and following best practices as possible.

Looking back to our results and all the work presented here, we believe that we have accomplished all three goals successfully. This has not been an easy journey and we are not saying that there is no space for improvements. As we have seen in table **??** there is still a big gap to fill in terms of accuracy for VQA models.

At the beginning we started the project with little knowledge of how Recurrent Neural Networks (RNN) worked and how to apply them to process sequences of text. Building our first QA model only for text gave us the needed expertise to move forward to the more complex systems that we wanted to face, VQA. A remarkable point here was the incorporation of Santiago Pascual to the team, which helped us a lot in the process of gaining this understanding of the RNN and the NLP world.

Having this QA model as a starting point, we started developing new VQA models that could merge the text and visual information, but not without reaching dead ends such as the use of VGG-16 (which, even if it is possible to use it, we could not due to timing constraints). We tried different model's configuration parameters and architectures and through this iterative process of modifying the model and checking its performance we gain this notion of how the model is affected by those parameters and also we noticed that the complexity of the task does not give much space in terms of adjusting the parameters. With that we mean that the models worked at some specific range of values which was not very large.

Finally we could train and publish a model with a similar accuracy of the baseline one defined by the VQA organizers but more prone to extending it and improving it.

We would like to highlight that during the course of this project we presented an extended abstract to the CVPR16 VQA workshop and it was accepted. This extended abstract with its own poster was presented in the VQA workshop at 26th June 2016. The extended abstract exposed one of our ideas of future work.

Having this in mind, as a future work we are planing to take this last model and attach a generative language model at the end so it can predict multiple word answers. We believe that making this improvement we will be able to outperform the baseline.
Another improvement that we are thinking about is to change the sentence embedding of the question to a character embedding, which exploits even more information of the words such as the relation between prefix and sufixes.
As we have already mention, we also want to actually implement the ideas in our extended abstract to create a model that is able to generate Question-Answer Pairs (QAP) from an image.

Chapter 7

Appendices

In this appendices you will find our accepted extended abstract presented in the CVPR16 VQA workshop and the poster we presented in Las Vegas, Nevada (USA) on June 26, 2016.

Towards Automatic Generation of Question Answer Pairs from Images

Issey Masuda Mora, Santiago Pascual de la Puente, Xavier Giro-i-Nieto
Universitat Politecnica de Catalunya (UPC)
Barcelona, Catalonia/Spain

issey.masuda@alu-etsetb.upc.edu, santiago.pascual@tsc.upc.edu, xavier.giro@upc.edu

Abstract

This extended abstract presents our research in the generic field of Visual Question-Answering (VQA) focusing on a new branch that aims to generate question-answer pairs based on an image. To do so, we use the VQA dataset provided for the VQA challenge to train a Deep Neural Network which has the image as an input and two different outputs, the question and its associated answer.

1. Introduction

Recurrent Neural Networks (RNNs) are widely used in many Natural Language Processing (NLP) tasks nowadays (e.g. language modelling, machine translation) for their good capacity to model sequences while Convolutional Neural Networks (CNNs) are more often used to process images in Computer Vision. Here we explore a model that joins both visual and textual features using Long Short-Term Memory networks (LSTM), a RNN extension, and an off-the-shelf model for visual features extraction in order to generate image-related Question-Answer Pairs (QAP).

2. Related Work

Different network architectures have been presented to solve Visual Question-Answering tasks [1, 2, 5, 6]. As a common approach, they all extract visual features from an image using CNNs, which have been proven to achieve state-of-the-art performance in image classification tasks. These visual features are combined with a vectorized representation of the question. The method to vectorize the question ranges from a simple Bag-Of-Words (BOW) representation [5] to the use of RNNs [2] to obtain an embedding of the question.

As for the generation of QAP, only a text-based study has been found. Serban *et al.* [3] propose an encoder-decoder architecture using GRU RNNs to generate a question from a fact, which is a tuple of subject, relationship, object. The model is trained in such a way that the answer to the gener-

ated question is the object from the fact.

3. Model

Figure 1 depicts our model architecture that is discussed below.

Visual features extraction. We consider using the model VGG-16 net [4] to extract the visual features from the input image. These features are not the output of the final softmax layer of the net but the output of the last max-pooling (before the fc-4096 layer).

Question generation. The visual features are injected into a Long Short-Term Memory (LSTM) RNN that will learn how to generate an embedding of the question. This embedding will then be fed into a softmax layer of dimension equal to our vocabulary size, which in turn will predict the n-th word of the question. At each time-step, the same visual features are shown to the model, which will produce one question word at a time. The maximum question length is then a parameter to specify to the model, but the question can have less words if an End of Sentence (EOS) is predicted. Notice that at the output of the LSTM we will have an embedding of the question word being predicted.

Answer prediction. The question embedding will then be given to another LSTM in charge of producing the answer to the question. This module also receives the visual features of the image extracted with the VGG-16 as an input. Only the last state of this LSTM will be used to predict the answer, as only at that moment the network will have seen all the question. Again, a softmax layer will be used to obtain the predicted word.

Training We train the model end-to-end (E2E) freezing the VGG-16 layer weights. The dataset used to train the model is the one provided for the Visual Question Answering challenge[1], which provides three questions per image and ten answers per question.
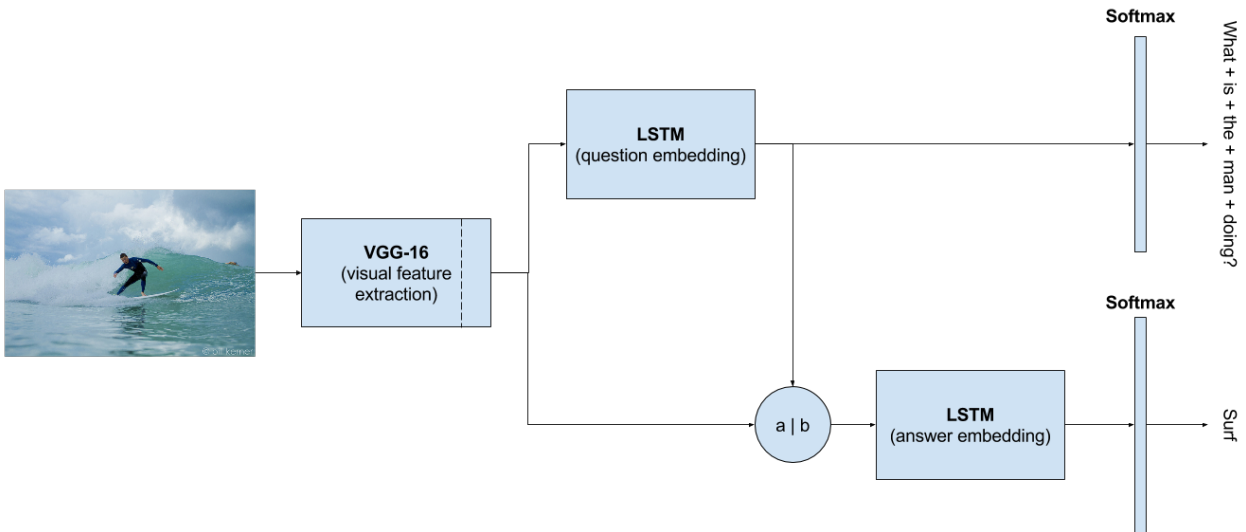
---

[1] http://www.visualqa.org/

1

Figure 1. Automatic QA generation model. LSTMs are used to generate sequences of words

4. Applications

A model able to generate question-answer pairs present a wide number of different interesting applications:

- **Synthetic dataset generation:** Similarly to [3], a corpus of Visual-Question-Answer tuples can be generated using this approach to increase existing datasets such as VQA dataset. With a technique able to produce more examples for visual question-answering tasks, we can offer to the research community a broader dataset to train their models with. Given the nature of this dataset, it could be use to train the models but not for testing purposes.

- **Pedagogy applications for kids:** Building a model that can ask questions related to images to the kids (such as 'What is on the table?') can be a very effective tool to help them learn the language in an interactive way and more important, in a customize way. Using own images could ease the process of linking together objects/spaces with their names.

- **Language learning:** With the same spirit than the previous example, applications to help users to learn a new language can be very appealing.

5. Conclusions

We have proposed a model that can generate QAPs given an image. We use an off-the-shelf model to extract the visual features from the image and two RNN blocks to predict the question and answer. We also expose some of the applications where such a model could be used to.

References

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

[2] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *IEEE conference on Computer Vision and Pattern Recognition*, 2016.

[3] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

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

[6] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *IEEE conference on Computer Vision and Pattern Recognition*, 2016.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament de Teoria del Senyal
i Comunicacions

Towards Automatic Generation of Question Answer Pairs from Images

Issey Masuda Mora   Santiago Pascual de la Puente   Xavier Giró-i-Nieto

Motivation

- Reminiscene Therapy with Werable Cameras.

- A multimedia problem.

Computer Vision
Natural Language Processing
Knowledge Representation and Reasoning
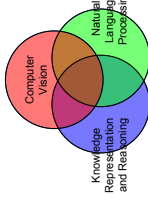
- Related work generating questions

GRNN

From text [1]

CNN
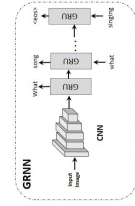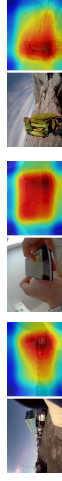
From images [2]

Project overview

- Summarization [3,4].

- Visual memorability [5].
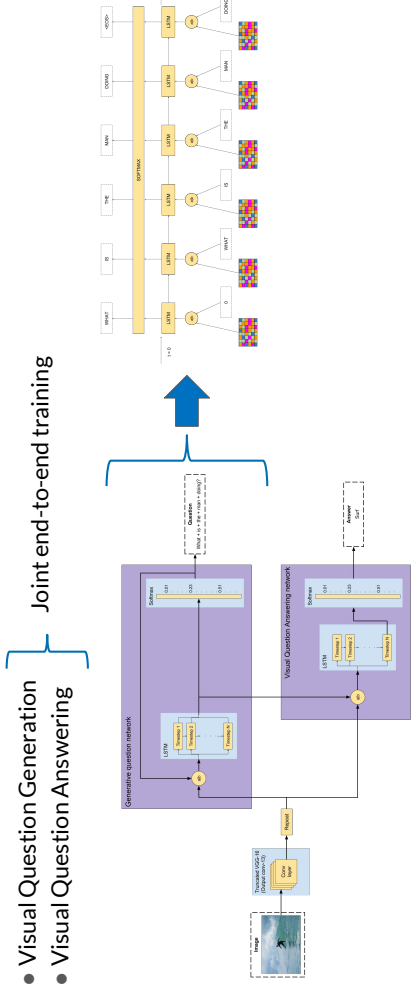
- System pipeline.

Narrative Camera

Egocentric images

Visual Summary

Check Validity

Relevant images of a day

Choose Memorables

Retrieve Answer

Formulate Questions

Reminiscence Therapy
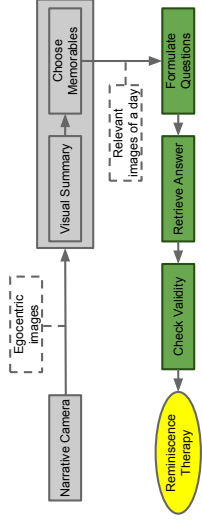
Visual Question and Answer Generation

Joint end-to-end training

- Visual Question Generation
- Visual Question Answering

Generative question network

Visual Question Answering network

Trained VGG-16 (Output conv7)

Conv layer

Image

Other applications

Hello Hola Bonjour Hello Hola Hola

Acknowledgements

[1] Serban et al, "Generating factoid questions with recurrent neural networks", arXiv 2016.
[2] Mostafazadeh et al, "Generating Natural Questions About an Image", arXiv 2016.
[3] Bolaños et al, "Visual Summary of Egocentric Photostreams by Representative Keyframes", WEsAX 2015.
[4] Lidon et al, "Semantic Summarization of Egocentric Photo Stream Events", arXiv 2015.
[5] Carné et al, "EgoMemNet: Visual Memorability Adaptation to Egocentric Images" CVPRW FPV 2016.

nVIDIA.

GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

Generalitat de Catalunya

Bibliography

[1] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[2] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[5] 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. *arXiv preprint arXiv:1606.01847*, 2016.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] Jin-Hwa Kim, Sang-Woo Lee, Dong-Hyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Multimodal residual learning for visual qa. *arXiv preprint arXiv:1606.01455*, 2016.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Zhen Liu. Kernelized deep convolutional neural network for describing complex images. *arXiv preprint arXiv:1509.04581*, 2015.

[12] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *arXiv preprint arXiv:1606.00061*, 2016.

[13] Lin Ma, Zhengdong Lu, and Hang Li. Learning to answer questions from image using convolutional neural network. *arXiv preprint arXiv:1506.00333*, 2015.

[14] Arun Mallya and Svetlana Lazebnik. Learning models for actions and person-object interactions with transfer to question answering. *arXiv preprint arXiv:1604.04808*, 2016.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[17] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*, 2016.

[18] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. *arXiv preprint arXiv:1511.05756*, 2015.

[19] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707, 2016.

[20] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961, 2015.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[23] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[24] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*, 2016.

[25] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.

[26] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. *arXiv preprint arXiv:1511.03416*, 2015.