

FACULTY OF MATHEMATICS AND
INFORMATICS
DEPARTMENT OF COMPUTER
SCIENCE

N° :29.....



DOMAINE: Mathematics and Informatics

FILIERE: Informatics

OPTION: INFORMATION SYSTEMS AND
SOFTWARE ENGINEERING

.....

**A Dissertation in Fulfillment
For the Requirements of the Degree of Master
In Computer Science**

By:

BENAICHE Abla

Entitled

**Object Detection task in Visual
Question Answering**

Presented publicly to the jury:

DEBBI Hicham

University of M'sila

Supervisor

BOURAHLA Mustapha

University of M'sila

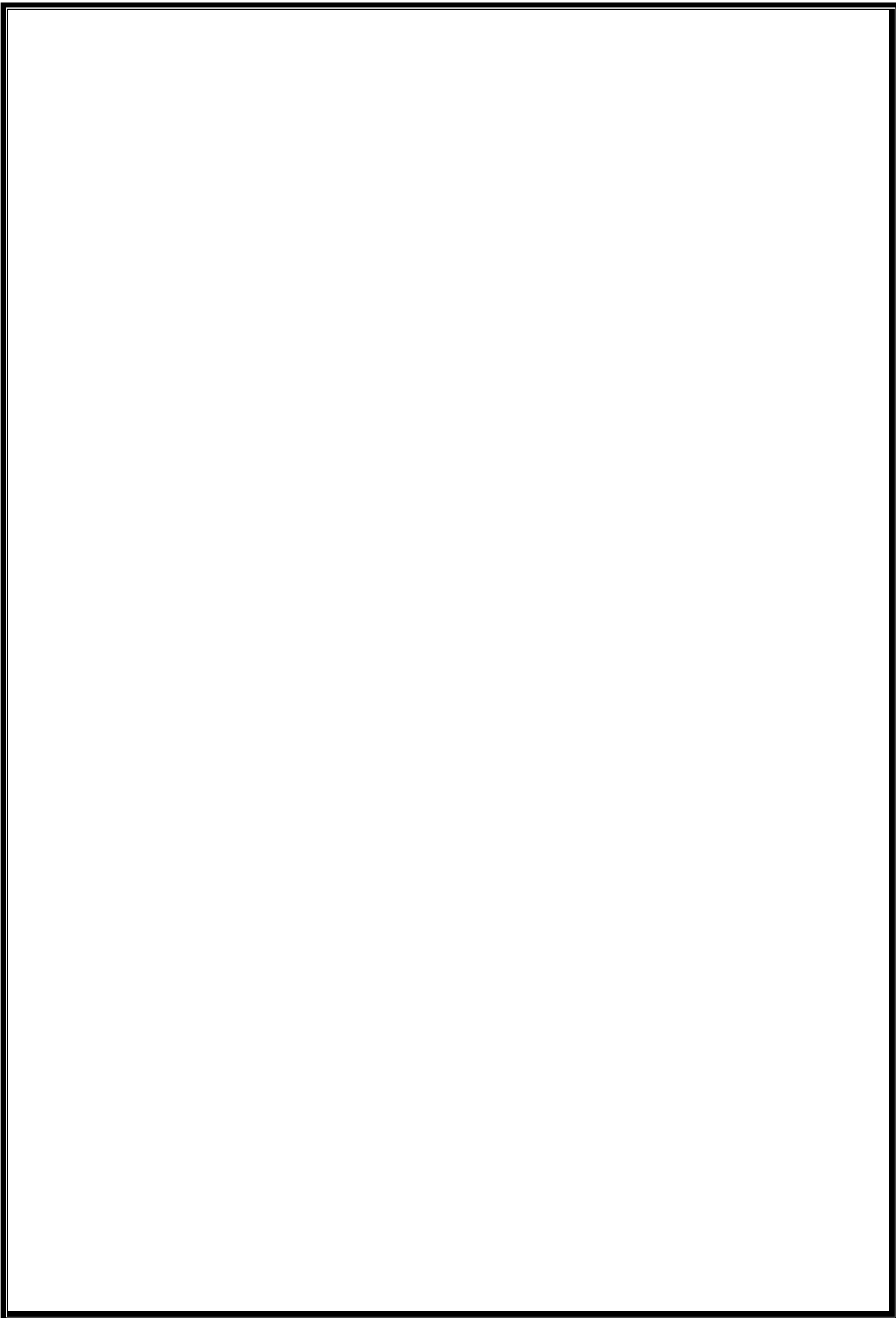
President

MELIOUH Amel

University of M'sila

Examiner

Academic Year: 2022/2023



DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

FACULTY OF MATHEMATICS AND
INFORMATICS
DEPARTMENT OF COMPUTER
SCIENCE

N° :.....29.....



DOMAINE: Mathematics and Informatics

FILIERE: Informatics

OPTION: INFORMATION SYSTEMS AND
SOFTWARE ENGINEERING

.....

A Dissertation in Fulfillment
For the Requirements of the Degree of Master
In Computer Science
By:

BENAICHE Abla

Entitled

Object Detection task in Visual
Question Answering

Presented publicly to the jury:

DEBBI Hicham

University of M'sila

Supervisor

BOURAHLA Mustapha

University of M'sila

President

MELIOUH Amel

University of M'sila

Examiner

Academic Year: 2022/2023

Acknowledgements

First of all, I would like to thank ALLAH the most merciful, who gave me the help, patience, and courage to accomplish this work.

I would like to thank my supervisor Dr. DEBBI Hichem for his supervision, continuous caring, and guided discussion. I thank the jury members Mr. BOURAHLA Mustapha and Ms.MELIOH Amel for their time and dedication.

Special thanks to my teacher and sister LOUANAS Salma for her help, encouragement, and her valuable assistance during my project.

Finally, I would like to thank all my family, friends, and all my precious people for their sacrifices, aid, support, and encouragement.

Abstract

This research proposes a novel approach for Visual Question Answering (VQA) by incorporating object detection features into the model as image features instead of traditional CNN features. The aim is to leverage specific information about objects present in the image to improve the VQA task. The experiments yielded accuracy values of 76% for Yes/No questions, 43% for counting questions, and 47% for other questions. Overall, this research enhances the understanding and processing of visual information by incorporating object detection features, leading to improved accuracy and performance in answering questions based on images.

Keywords: Visual Question Answering (VQA), object detection, image features, CNN, VQA v2 dataset.

ملخص: تقترح هذه الدراسة نهجًا جديدًا للإجابة على الأسئلة المرئية (VQA) من خلال دمج ميزات كشف الكائنات في النموذج كميزات الصورة بدلاً من ميزات CNN. الهدف هو الاستفادة من المعلومات المحددة حول الكائنات الموجودة في الصورة لتحسين مهمة الإجابة على الأسئلة المرئية. أظهرت التجارب قيم دقة بنسبة ٧٦% للأسئلة نعم و٣٤% للأسئلة التي تتطلب العد، و٧٤% للأسئلة الأخرى. بشكل عام، تعزز هذه الدراسة فهم ومعالجة المعلومات المرئية من خلال دمج ميزات كشف الكائنات، مما يؤدي إلى تحسين الدقة والأداء في الإجابة على الأسئلة استنادًا إلى الصور.

الكلمات الرئيسية: الرد على الأسئلة المرئية، اكتشاف الكائن، خصائص الصورة، CNN، مجموعة بيانات VQA v2.

Résumé: Cette recherche propose une nouvelle approche pour la Réponse aux Questions Visuelles (VQA) en incorporant les caractéristiques de détection d'objets dans le modèle en tant que caractéristiques d'image au lieu des caractéristiques du CNN. L'objectif est d'exploiter des informations spécifiques sur les objets présents dans l'image pour améliorer la tâche de VQA. Les expériences ont donné des valeurs de précision de 76% pour les questions Oui/Non, 43% pour les questions de décompte et 47% pour les autres questions. Dans l'ensemble, cette recherche améliore la compréhension et le traitement des informations visuelles en incorporant les caractéristiques de détection d'objets, ce qui permet d'améliorer la précision et les performances dans la réponse aux questions basées sur des images.

Mots clés : Réponse aux Questions Visuelles (VQA), détection d'objets, caractéristiques d'image, CNN, ensemble de données VQA v2.

Contents

Acknowledgements	I
Abstract	II
General Introduction	1
1 Deep Learning	3
1 Introduction	4
2 Deep learning	4
2.1 Deep Neural Network	5
2.1.1 Definition	5
2.1.2 Activation function	6
2.1.3 Fundamental concepts in the neuron network	9
2.2 Deep Neural Network Types	9
2.2.1 Convolutional Neural Network	9
2.2.2 Recurrent Neural Network	12
2.3 Deep Learning Platforms	14
2.3.1 TensorFlow :	14
2.3.2 Keras :	15
2.3.3 PyTorch :	15
3 Natural Language processing	16
3.1 Natural language processing	16
3.2 Components of NLP	17

3.2.1	Natural Language Understanding (NLU)	17
3.2.2	Natural Language Generation (NLG)	17
3.3	Phases of NLP	17
3.3.1	Lexical Analysis and Morphological	18
3.3.2	Syntactic Analysis (Parsing)	19
3.3.3	Semantic Analysis	19
3.3.4	Discourse Integration	19
3.3.5	Pragmatic Analysis	19
3.4	Applications of NLP	19
3.4.1	Machine Translation	19
3.4.2	Speech Recognition	20
3.4.3	Sentiment Analysis	20
3.4.4	Chabot	20
3.4.5	Question Answering	21
3.5	NLP Techniques for Text	21
3.5.1	Tokenization	21
3.5.2	Normalization	22
3.5.3	Stop words	22
3.5.4	Regex	22
3.5.5	Bag-of-words	23
3.5.6	Word Embeddings	23
3.6	NLP APIs	26
3.7	NLP Libraries	26
4	Computer Vision	27
4.1	Computer Vision	27
4.2	Computer vision and CNNs	27
4.3	Tasks in Computer Vision	28
4.4	Computer vision Libraries	29
4.5	Computer Vision Applications	30
5	Conclusion	32
2	Visual Question Answering	33
1	Introduction	34
2	Visual question answering	34

3	Major issues in visual question answering	35
3.1	Evaluation of open-ended and multiple-choice questions	35
3.2	Dataset Biass	36
3.3	Balanced Binary Questions	36
3.4	Image Featurization from real VQA datasets	36
3.5	Need for application-oriented datasets	36
3.6	Questions requiring reading scene-text	36
3.7	Conversational questions	37
3.8	The imbalance between answerable and unanswerable classes	37
4	Applications of visual question answering	37
5	VQA Datasets	38
5.1	The COCO dataset	38
5.2	The DAQUAR dataset	38
5.3	The VQA dataset	39
5.4	The COCO-QA dataset	40
5.5	The CLEVR dataset	40
5.6	The KVQA dataset	41
6	The approaches in VQA	42
6.1	Feature extraction	42
6.1.1	Image Featurization	42
6.1.2	Question featurization	43
6.2	The fusion block	43
7	Conclusion	45
3	Object Detection in Computer Vision	46
1	Introduction	47
2	Object detection	47
3	The principle of object detection	47
4	Applications of object detection	48
5	Important Object Detection Concepts	49
5.1	Région d'intérêt (RoI)	49
5.2	Intersection Over Union (IoU)	49
5.3	Non-Max Suppression (NMS)	50
5.4	Mean Average Precision (mAP)	51

5.5	Bounding-box regression	51
6	Object detection algorithms	51
6.1	TWO-STAGE TARGET DETECTION FRAMEWORK	51
6.1.1	R-CNN	51
6.1.2	SPP-Net	52
6.1.3	Fast R-CNN	53
6.1.4	Faster R-CNN	53
6.2	ONE-STAGE TARGET DETECTION ALGORITHM	54
6.2.1	YOLOv1	54
6.2.2	YOLOv2	55
6.2.3	YOLOv3	55
6.2.4	SSD	55
6.2.5	YOLOv4	56
7	Conclusion	57
4	Implementation and Results Analysis	58
1	Introduction	59
2	Dataset Overview	59
3	Our model	60
3.1	Image presentation	61
3.2	Question presentation	64
3.2.1	Preprocess	64
3.2.2	Question Encoded	68
3.3	Fusion process	69
3.4	MLP Classifier	70
3.5	Output Generation	70
4	Training and validation of the proposed model	71
5	Experimental Results	73
6	Overview of the development environments used	75
6.1	Hardware environment	75
6.2	Software environment	76
6.2.1	Choice of programming language	76
6.2.2	Libraries	76
6.2.3	Software and Tools	77

7	Conclusion	80
	General Conclusion	81

List of Figures

1.1	The relationship between AI, ML, and DL [1].	5
1.2	A general architecture of a simple network with one hidden layer and b a deep neural network with multiple hidden layers[2].	6
1.3	Linear Activation Function [3].	6
1.4	Sigmoid Activation Function [3].	7
1.5	Tanh Activation Function [4].	7
1.6	Relu Activation Function [4].	8
1.7	Softmax Activation Function [5].	8
1.8	CNN architecture [6].	10
1.9	Recurrent Neural Network[7]	13
1.10	A chain of LSTM cells[8]	14
1.11	Unfolded Gated Recurrent Unit (GRU) [8]	14
1.12	Tensorflow [9]	15
1.13	Keras [10].	15
1.14	Pytorch [11].	16
1.15	Natural Language Processing [12].	16
1.16	Phases of NLP [13].	18
1.17	TF-IDF [14].	24
1.18	Word2Vec [15]	25
1.19	OpenCV [16]	29
1.20	SimpleCV [17].	30
1.21	Scikit-image [18]	30

2.1	VQA as a multidisciplinary research problem [19].	34
2.2	An example of a workflow of the VQA method is on the left and some of the CV tasks and NLP tasks involved in VQA are on the right. [20].	35
2.3	Examples of annotated images from the COCO dataset [21].	38
2.4	Sample of images from the DAQUAR dataset [22].	39
2.5	VQA Dataset [23].	40
2.6	CLEVR Dataset [23].	41
2.7	K-VQA dataset[24].	41
2.8	Simplified scheme of VQA approaches [21].	42
2.9	Combining CNN/RNN for VQA[23]	44
3.1	The principle of object detection [25].	48
3.2	Examples of areas of application [26].	48
3.3	Regions of Interest [27]	49
3.4	Object detection predictions based on IoU [28]	50
3.5	NMS Application Process [29]	50
3.6	R-CNN architecture [30].	52
3.7	SPP-Net architecture [30].	53
3.8	Fast R-CNN architecture [30].	53
3.9	Faster R-CNN architecture[31]	54
3.10	YOLOv1 architecture [30]	55
3.11	SSD architecture [32].	56
3.12	Architecture global YOLO v4 [33]	56
4.1	Our model.	60
4.2	Accessing and Manipulating the H5 file.	61
4.3	Creating a Dataset for Features using “fd.create_dataset”.	61
4.4	Reading and Accessing TSV File	62
4.5	Extracting and storing the image Widths and Heights into their dataset	62
4.6	Encoded features to store them in the dataset	63
4.7	Loading the annotations and questions from multiple JSON files	64
4.8	Creating the function of Punctuation for handling punctuation and stripping Marks	65
4.9	Creating a vocabulary dictionary	65

4.10	define the encode function for indexing questions	66
4.11	Loading GloVe Embeddings from Pickle file	67
4.12	Converting GloVe matrix to NumPy array	67
4.13	Saving processed data and GloVe embeds to Pickle file	68
4.14	BiGRUEncoder for questions.	69
4.15	Fusion Process, The element-wise product represents the multiplication of the features of the image box V_i with the values of the question's hidden state q (each image includes 36 boxes)	70
4.16	Initializing Adam optimizer with learning rate and weight decay in PyTorch .	72
4.17	Implementation of Exponential Learning Rate Decay using the <code>lr_schedulermodule</code>	72
4.18	Python [34]	76
4.19	Anaconda Navigator [35].	77
4.20	Jupyter Notebook [36]	78
4.21	PyTorch [11]	79
4.22	Kaggle [37].	80

List of Tables

4.1	Datasets of H5 file and their properties.	63
4.2	The hyperparameters of the model.	71
4.3	Experiments questions.	74
4.4	Experiments results.	74
4.5	Our obtained results and the results from the paper.	75

General Introduction

In recent years, Visual Question Answering (VQA) has emerged as a captivating field at the intersection of computer vision and natural language processing. VQA aims to enable machines to comprehend and respond to questions posed in natural language about images. This technology holds great potential for applications in areas such as assisting visually impaired individuals, intelligent image retrieval, and interactive virtual assistants. Existing VQA models have achieved promising results by leveraging deep learning techniques and large-scale datasets. However, these models often struggle with providing accurate answers in some cases.

Our work fits into this context, dealing with exploring and evaluating the effectiveness of incorporating object detection features within the VQA task to enhance the representation of the visual content and improve the performance of VQA systems. By integrating object detection features into the VQA pipeline, we aim to enrich the understanding of the image and improving the reasoning ability of the VQA models.

The objective of our work is to develop a neural network model that seamlessly integrates object detection features into the VQA pipeline. We aim to explore how these features can enhance the performance of existing VQA models and improve their ability to reason about image content. By leveraging the extracted object detection features, the VQA model can have a more focused and detailed understanding of the objects present in the image, enabling it to generate more accurate and context-aware answers.

To provide a comprehensive understanding of the topic, the content of our dissertation will be organized as follow:

Chapter 1: Deep Learning

Chapter 2: Visual Question Answering

Chapter 3: Object Detection in computer vision

Chapter 4: Implementation and Results Analysis

Chapter 1

Deep Learning

1 Introduction

Nowadays, deep learning occupies the center of attention of many researchers and specialists. It has emerged as a transformative force in many fields, such as natural language processing and computer vision. , revolutionizing the way we analyze, interpret, and generate visual and textual data. By leveraging the power of deep neural networks, these fields have witnessed remarkable advancements in recent years, enabling machines to perform complex tasks with unprecedented accuracy and efficiency.

This chapter provides an overview of deep learning, natural language processing (NLP), and computer vision. We discuss deep neural networks, including CNNs and RNNs, and their fundamental concepts. Additionally, we explore NLP components, phases, and techniques, such as word embeddings. We also cover computer vision tasks, libraries like OpenCV, and applications.

2 Deep learning

Deep learning (DL) is a subfield of machine learning (ML), and the latter is a subfield of artificial intelligence (AI) (We illustrate this imbrication between the three branches in Figure 1.1. In other words, Deep Learning is a new trend in learning where we can teach computers to process and analyze data in a way inspired by the human brain, using artificial neural networks.

Nowadays, deep learning has proven its effectiveness in several areas and complex tasks: automatic natural language processing, Speech recognition, Image processing and object recognition, machine translation, etc.

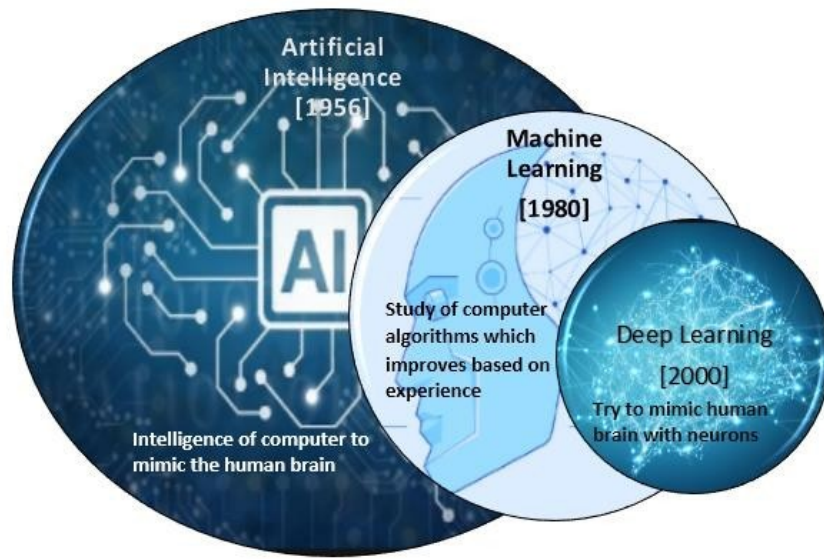


Figure 1.1: The relationship between AI, ML, and DL [1].

2.1 Deep Neural Network

2.1.1 Definition

Artificial neural networks (ANNs) with multiple layers, also known as deep neural networks (DNNs). These neural networks aim to simulate the functioning of the biological neuron of the human brain.

A neuron forms the basic structure of a neural network. It is organized into layers. Layers are three types- An input Layer, an Output layer, Hidden Layer.

- Input layer is the first layer of the network which receives the input.
- Hidden layers called processing layers are the ones that perform specific tasks on the incoming data and pass on the output generated by them to the next layer.
- Output layer is the final layer of the network and generates the output, which is sent to other neurons for further processing or is the final output.

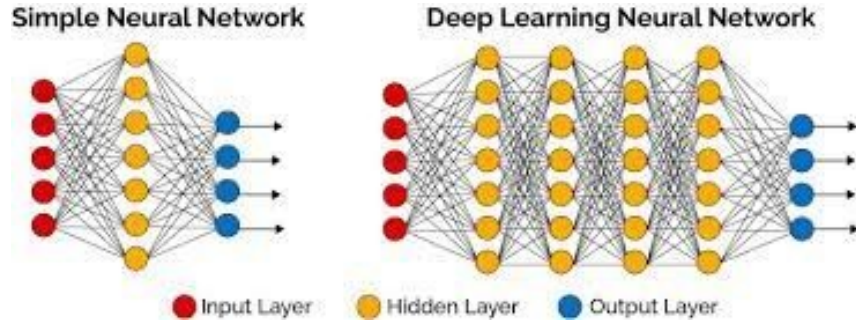


Figure 1.2: A general architecture of a simple network with one hidden layer and b a deep neural network with multiple hidden layers[2].

2.1.2 Activation function

The activation function plays an important role in the neuron network. It decides what should be the node's output based on input signals.

There are many types of activation functions, we discuss some of them in the following:

- **Linear activation function:** Linear or Identity Activation Function is a function which directly proportional to the input value. Therefore, the output can take any value between -infinity and infinity. We can define the function as:

$$output = y * x$$

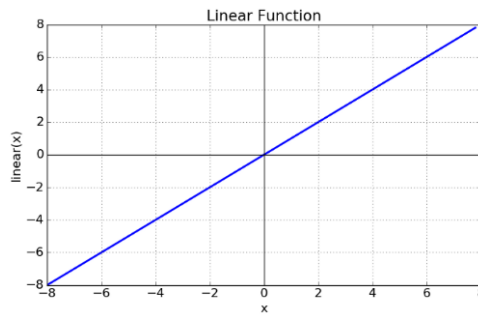


Figure 1.3: Linear Activation Function [3].

- **The Sigmoid activation function:** Sigmoid is the non-linear or logistic activation function used for neural networks in which the output values are either binary or in the range between 0 and 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

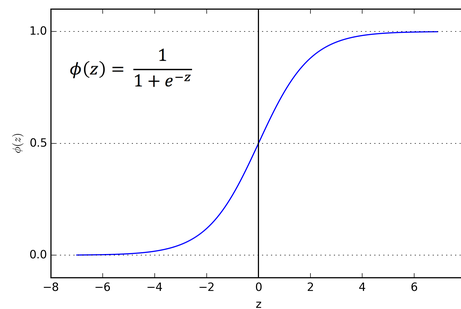


Figure 1.4: Sigmoid Activation Function [3].

- **Tanh (Tangent Hyperbolic function) function:** The tanh function is also a non-linear activation function like a sigmoid function. The inputs to the next layers will not always be of the same sign because the range of values is between -1 and 1.

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

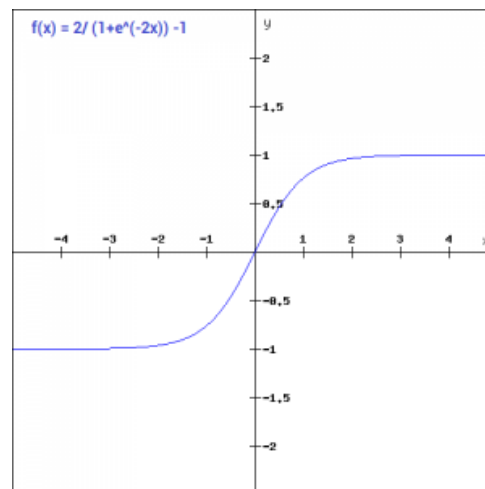


Figure 1.5: Tanh Activation Function [4].

- **Rectified Linear Unit (ReLU) activation function:** ReLU is one of the most popular activation functions. ReLU is used in the recent neuron networks for the hidden layers. This function returns 0 if the input x is less than 0 and returns x

itself if it is greater than 0 (the result can range from 0 to infinity).

$$f(x) = \max(0, x)$$

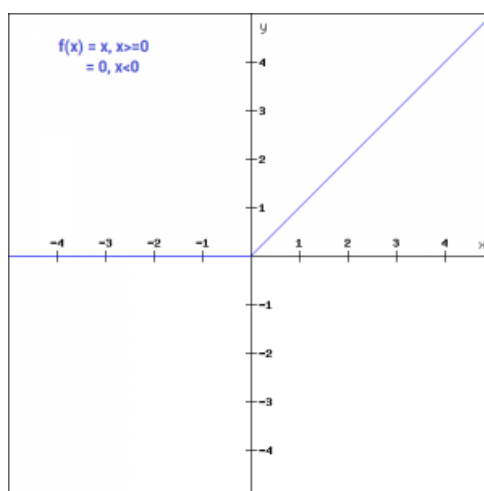


Figure 1.6: Relu Activation Function [4].

- **Softmax activation function:** Softmax is an expansion of a sigmoid function and it is useful for multi-class classification problems. It calculates probabilities of class membership.

$$output = e^x / (sum(e^x))$$

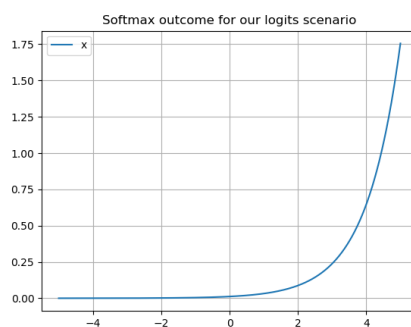


Figure 1.7: Softmax Activation Function [5].

2.1.3 Fundamental concepts in the neuron network

- **Forward propagation and Backpropagation:** In the neural network, Forward propagation is the way data moves from the input layer to the output layer to get our output values. We compare these obtained results with the expected results, and then we calculate the errors and work Backpropagation. This allows us to update the weights and train the network. In Backpropagation, the information propagates back from the output layer to the input layer, to train the network, to reduce the cost function by adjusting the weights.
- **Cost function :** The cost function is an important parameter used in the network to evaluate the performance of the model .it measures the error between obtained results and the expected results. The goal while running the network is to get the predicted result and to reduce the error then minimize the cost function.
- **Gradient descent:** Gradient Descent is an algorithm used in machine learning (ML) and deep learning (DL) to optimize the cost function (error) of the model. The gradient descent algorithm performs two steps to reduce the cost function. Firstly, it calculates the gradient (slope) at a starting point (This is the first-order derivative of the function. Secondly, It makes a scaled step opposite the gradient, so the slope increases from the current point by alpha times. Alpha is called the learning rate which is a parameter used in the optimization process to determine the length of the steps.
- **Batches and epochs :** While training a network, the dataset (the input) is divided into several parts or batches of equal size. The number of these batches that pass through the neural network at one time is called batch size. When we pass the entire training data (all batches) forward and backward, this is called an epoch.

2.2 Deep Neural Network Types

2.2.1 Convolutional Neural Network

- General on CNNs:

A Convolutional Neural Network (CNN) is a special Feedforward Neural Network that can be characterized by a convolutional layer, which differs from other neural networks. CNN networks have a grid-like topology and focus on the fact that the

input will be imaged.

CNN is used in the area of image recognition, object detection, natural language processing, etc. The capability of automatically discovering essential features from the input without the need for human intervention makes it more powerful than a traditional network.

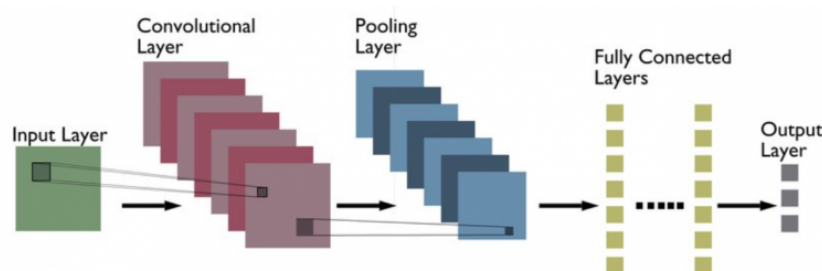


Figure 1.8: CNN architecture [6].

The three main types of layers to build this type of network are the convolutional layer, pooling layer, and fully connected layer.

The convolutional layer is the most important layer in CNN, which consists of a collection of matrices called filters or (also kernels). It uses at least one kernel to convolve it with the input images to extract the most useful information (features) for the task and stored it in feature maps. The colored input image (RGB image) with three channels is convolved with filters, which must have three channels as well.

To modify the amount of movement a filter has over the input image, we use a hyper-parameter called stride. In addition, to obtain the output image that has the same size as the input, we add an extra layer of zeros across the input images. This action gets by another hyper-parameter called padding.

The last component of the convolutional layer is an activation function. Generally, we used here ReLu function or Tanh function.

The pooling layer is called also the sub-sampling technique, which reduces the dimensions of the feature map generated by the Convolutional layer. The most common types of pooling are average pooling and max pooling.

Similar to the convolutional process, both the stride and the filter are initially

sized before the pooling operation is executed.

The fully connected layer is the last in the convolutional neural network. It is characterized by the Fully Connected (FC) approach, which means that all input neurons are connected to every output layer neuron.

The input in this layer is the output from the final Pooling or Convolutional Layer, which is flattened (unroll all its values into a vector) and then connected to the fully connected layer.

After all fully connected layers, we use the softmax activation function, which predicts a multinomial probability distribution.

- architectures of CNN:

Lenet: LeNet is the first CNN architecture. It was developed in 1998 by Yann LeCun, Corinna Cortes, and Christopher Burges for handwritten digit recognition problems. The LeNet architecture consists of multiple convolutional and pooling layers, followed by a fully-connected layer. The model has five convolution layers followed by two fully connected layers. LeNet was the beginning of CNNs in deep learning for computer vision problems. However, LeNet could not train well due to the vanishing gradients problem. To solve this issue, a shortcut connection layer known as max-pooling is used between convolutional layers to reduce the spatial size of images which helps prevent overfitting and allows CNNs to train more effectively [6].

AlexNet: Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton developed the first work that popularized convolutional networks in computer vision [38]. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (16% top 5 error compared to second place with a 26% error). The network had a very similar architecture to LeNet, but it was deeper, larger, and had convolutional layers stacked on top of each other (previously it was common to have only one CONV layer immediately followed by a POOL layer) [39].

GoogLeNet (Residual network): The winner of ILSVRC 2014 was a convolutional network by Szegedy et al [40]. Their main contribution was the development of a launch module, which significantly reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Also, this

document uses the average clustering instead of the fully connected layers at the top of the ConvNet, eliminating a lot of parameters that seem unimportant. There are also several follow-up versions of the GoogLeNet, most recently Inception-v4 [39].

VGG Net (Visual geometry group): VGGNet is the CNN architecture that was developed by Karen Simonyan, and Andrew Zisserman et al. at Oxford University. Real-world applications/examples of VGGNet CNN architecture include the ILSVRC 2014 classification task, which was also won by GoogLeNet CNN architecture. The VGG CNN model is computationally efficient and serves as a strong baseline for many applications in computer vision due to its applicability for numerous tasks including object detection. Its deep feature representations are used across multiple neural network architectures like YOLO, SSD, etc [6].

ResNet: ResNet is the CNN architecture that was developed by Kaiming He et al [41]. to win the ILSVRC 2015 classification task with a top-five error of only 15.43%. CNNs are mostly used for image classification tasks with 1000 classes, but ResNet proves that CNNs can also be used successfully to solve natural language processing problems like sentence completion or machine comprehension, where it was used by the Microsoft Research Asia team in 2016 and 2017 respectively. Real-life applications/examples of ResNet CNN architecture include Microsoft's machine comprehension system, which has used CNNs to generate the answers for more than 100k questions in over 20 categories. The CNN architecture ResNet is computationally efficient and can be scaled up or down to match the computational power of GPUs [6].

2.2.2 Recurrent Neural Network

- **General on RNN:**

Recurrent Neural Network (RNN) is one of the most popular neural networks, which is used to model sequence data and produced in it predictive results that other algorithms cannot. Like feed-forward and CNN, recurrent networks learn from training input, however, distinguished by their “memory”, which allows them to impact current input and output by using information from previous inputs. Unlike typical DNN, which assumes that inputs and outputs are independent of

one another, the output of RNN is reliant on prior elements within the sequence[2]. RNN has proven successful in several areas such as script generation handwritten, text data, and speech recognition.

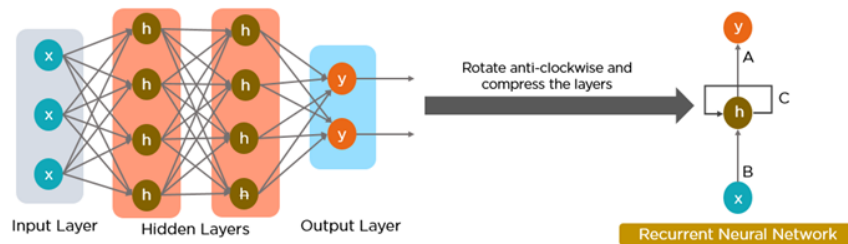


Figure 1.9: Recurrent Neural Network[7] .

- **variants of RNN:**

The most common variants of the recurrent network that minimizes the issues (like the Vanishing Gradients issue) and perform well in many areas are long short-term memory (LSTM), and Gated recurrent units (GRUs).

Long short-term memory (LSTM):

A long Short-Term Memory Network is a variant of an RNN, which solves the main problem of vanishing gradients. LSTM brought about changes in the architecture of the network by adding gates to the cell are the input gate, the output gate, and the forget gate.

These gates work as filters, make minor linear actions about the flow of information and decide which information is disregarded or kept in the cell state (the cell state is an important key in LSTM passed from the input to the output of a cell).

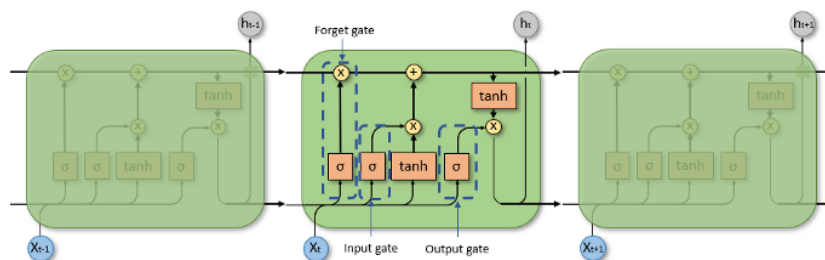


Figure 1.10: A chain of LSTM cells[8] .

Gated recurrent units (GRUs):

A Gated Recurrent Unit (GRU) is another popular type of recurrent neural network that solves the vanishing gradient problem like LSTM. However, the difference in LSTMs is that the GRU only relies on the hidden state as memory and does not have a cell state. Moreover, it only has two gates reset gate and the update gate.

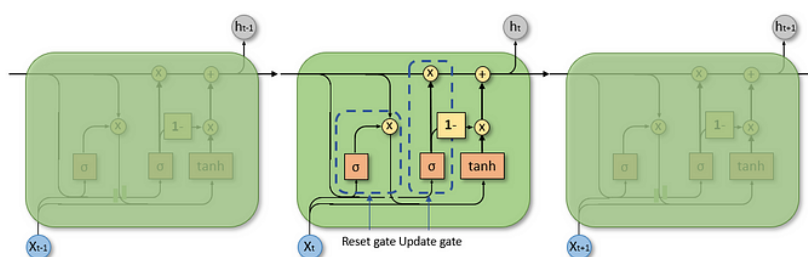


Figure 1.11: Unfolded Gated Recurrent Unit (GRU) [8] .

2.3 Deep Learning Platforms

2.3.1 TensorFlow :

Tensorflow is a free, open-source framework created by the GoogleBrain team and released in November 2015. It provides a convenient front-end API for building applications in Python or JavaScript while executing those applications in high-performance C++.

Tensorflow has seen rapid growth and is now by far the most popular framework used for deep learning, computer vision, and natural language processing (NLP) projects. It makes acquiring data, training models, serving predictions, and refining future results

easier.



Figure 1.12: Tensorflow [9] .

2.3.2 Keras :

Keras is one of the most popular open-source libraries written in Python, with a large community of coders behind it. It includes several features, such as support for both convolutional and recurrent networks, as well as easy and fast prototyping.

Keras provides an intuitive and easy-to-use interface for building NLP and computer vision models, including text classification, sentiment analysis image Classification and segmentation, semantic image clustering, 3D image classification, handwriting recognition, and more.



Figure 1.13: Keras [10].

2.3.3 PyTorch :

PyTorch is a popular open-source deep learning framework written in Python and created primarily by Facebook's AI research team. It is compatible with C++ and uses core Python concepts such as classes, structures, and conditional loops. In addition, it provides a dynamic computational graph (CPU and GPU computations) and a flexible framework for building, training, and deploying neural network models. Pytorch is widely used for various tasks, including computer vision and natural language

processing (NLP).



Figure 1.14: Pytorch [11].

3 Natural Language processing

3.1 Natural language processing

Natural language processing (NLP) is a subfield of computer science and artificial intelligence. A computer/system can truly understand human language and process it in the same way that a human does [42] .

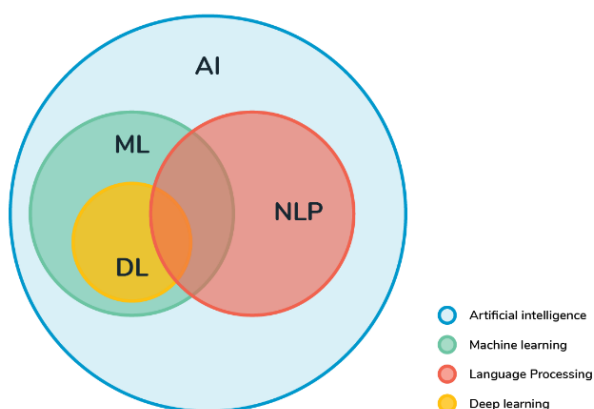


Figure 1.15: Natural Language Processing [12].

3.2 Components of NLP

There are mainly two components of NLP in artificial intelligence, which are Natural Language Understanding, and Natural Language Generation.

3.2.1 Natural Language Understanding (NLU)

Natural Language Understanding helps computers to understand and analyze human languages by extracting the text from content such as keywords, concepts, emotions, relations, semantics, etc. NLU is mainly used in the Business area to understand what people are saying in both spoken and written language, saving businesses countless hours and resources when analyzing troves of customer feedback.

It involves understanding the aspects of the language and mapping the input in natural language into valid representations.

3.2.2 Natural Language Generation (NLG)

NLG is another subcategory of NLP that turns computer-readable data into human-readable text. It involves Text planning, Sentence planning, and Text Realization to consider language rules based on morphology, lexicons, syntax, and semantics.

3.3 Phases of NLP

These are the general five steps of natural language processing:

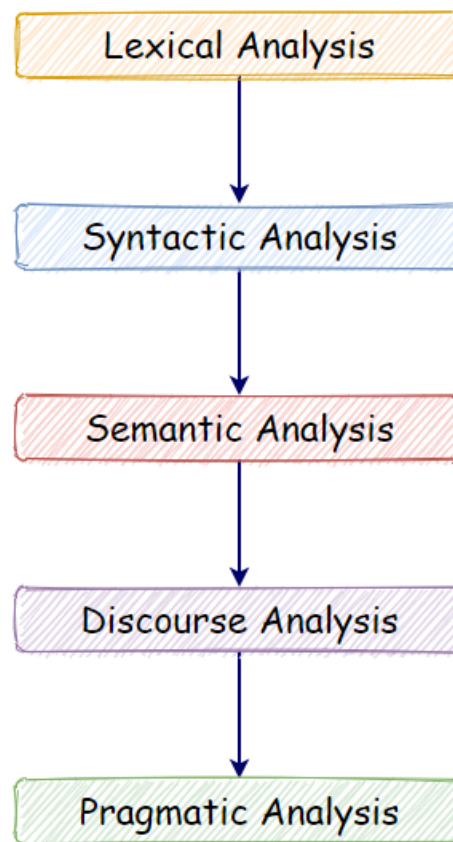


Figure 1.16: Phases of NLP [13].

- Lexical Analysis and Morphological,
- Syntactic Analysis (Parsing),
- Semantic Analysis,
- Discourse Integration,
- Pragmatic Analysis.

3.3.1 Lexical Analysis and Morphological

Is the first step of NLP, which divides text or sound waves into paragraphs, sentences, words, and other units. In other words, it is the process of transforming

a series of characters into a series of tokens. The analysis is performed to identify and describe the structure of words.

3.3.2 Syntactic Analysis (Parsing)

Syntactic Analysis is used to verify grammar and word arrangements and to show how words are related to one another. Hence, there is a need to analyze the grammatical meaning of the words in any sentence.

3.3.3 Semantic Analysis

Semantic analysis determines the exact meaning of words and assesses the text's meaning. It extracts only meaningful information from the text and rejects/ignores sentences that do not make sense.

3.3.4 Discourse Integration

Discourse analysis is a process that involves text or language analysis, text interpretation, and understanding of social interactions. On the other hand, it denotes an understanding of the context. The meaning of any single sentence is dependent on another sentence. It also considers the following sentence's meaning.

3.3.5 Pragmatic Analysis

The pragmatic analysis is concerned with total communication and language explanation. It means that what was said is re-interpreted to reflect what it truly meant and deriving those aspects of language that require real-world knowledge.

3.4 Applications of NLP

3.4.1 Machine Translation

Machine translation (MT) is one of the first uses of natural language processing which made many improvements to it. It is the conversion of text or speech from one

human language to another. Translation software is especially useful in business because it improves communication, enables the company to expand its customer base, and helps people understand foreign documentation quickly and at the lowest cost.

Machine Translation Tools Examples: Google Translate, Microsoft Translator, Facebook Translation, Amazon Translate.

3.4.2 Speech Recognition

Speech recognition, as well defined as automatic speech recognition (ASR), computer speech recognition, is a machine or program's ability to convert human speech or spoken words into written text. It is used in virtual assistants, like Google Assistant, and Alexa from Amazon. In addition, we use it in mobile, home automation, video recovery, dictating to Microsoft Word, and voice biometrics.

3.4.3 Sentiment Analysis

Sentiment Analysis or opinion mining is an analytical technique that uses natural language processing, and statistics to identify the view or emotion behind a situation. It means to automatically determine subjective information, such as opinions, emotions, feelings, and intent behind a text or speech.

This technique is used in various applications. It is used on the web to evaluate the user's attitude, actions, and emotional state. In addition, is frequently used to assist businesses in monitoring brand and product sentiment in customer feedback and understanding customer needs.

3.4.4 Chabot

One of the major uses of NLP is the chatbot. It assists businesses in achieving their goal of providing a positive customer experience and keeping them satisfied. Initially, chatbots were only used to answer customers' questions, but they have now evolved into personal companions. Chatbots can do everything from recommending a product to soliciting customer feedback.

3.4.5 Question Answering

Question Answering (QA) systems allow machines to automatically respond to questions in natural human language. A few question-answering models can produce answers that are devoid of context.

3.5 NLP Techniques for Text

NLP is a diverse field that necessitates the application of a variety of methods to successfully function and understand human language. Below, we discuss a selection of the techniques commonly used in NLP technology such as tokenization, normalization, stop words, etc.

3.5.1 Tokenization

Tokenization is one of the most fundamental NLP techniques used in natural language processing. It divides long strings of text into smaller parts called chunks or tokens after punctuation or special characters are removed. The main purpose of tokenization is to help in analyzing and interpreting the meaning of the text.

Tokenization can be applied to either separate words or sentences. In other words, larger text can be tokenized into sentences, and sentences can be tokenized into words.

- **Sentence Tokenization:**

Tokenization of sentences or sentence segmentation is the process of dividing the text into meaningful sentences. This appears to be a very simple concept. In some languages like English, French, etc, whenever we see a punctuation mark, we can split apart the sentences.

However, due to the use of full-stop characters for abbreviations, this problem is not trivial even in English. When processing plain text, tables of abbreviations with periods can assist us in avoiding incorrect sentence boundary assignments. In many cases, we use libraries to do that for us, so don't sweat the details for now.

- **Word Tokenization:**

Similar to the above process where we try to extract each sentence from the entire text, Word tokenization is used to separate the sentence into individual words or tokens. Word tokenization output can be converted to a Data Frame for better

text understanding in NLP applications.

3.5.2 Normalization

Normalization is one of the important techniques in the NLP field which is a pre-processing step aimed at improving the quality of the text and reducing its randomness by removing word inflection. In other words, it converts a token into its base form (standard). This reduces the amount of different information that the computer has to deal with, improving efficiency.

Two major strategies used for normalization are stemming and lemmatization.

- **Stemming:**

Stemming is a fundamental technique for applying text normalization in NLP. It is the process of eliminating the inflectional part (affixes like prefixes, and suffixes) from a word to obtain a word stem irrespective of meaning. For example, "playing", "played", and "plays" are all stemmed from "play". Stemming helps to reduce vocabulary, which improves accuracy.

- **Lemmatization:**

Lemmatization is an important NLP technique in the preprocessing pipeline. It aims to group different forms of a word by converting them to their root form. This root form is called lemma which is the dictionary form of the word. eg, For 'running', 'runs' will be converted to 'run'. Similarly, "Hates", and "hating" are forms of the word "hate"

3.5.3 Stop words

Stop words are words that are typically filtered out before processing natural language to avoid noise. These words make up the majority of human speech but do not add much information to the text.

Stop words are typically like determiners: the, a, an, another, or conjunctions: for, an, nor, but, or, yet, so, etc. It can change depending on the application.

3.5.4 Regex

A regular expression, regex, or regexp (RE) forms part of the basic techniques in NLP, which is a string of characters that specifies a text search pattern. Using a specialized syntax stored in a pattern, RE allows us to match or find other strings or sets of strings.

3.5.5 Bag-of-words

The Bag of Words (BoW) model is a Natural Language Processing approach, which is a flexible method of extracting features from texts. This model involves two points:

- A vocabulary of well-known words or tokens.
- A metric for the presence of known words.

A bag of words is a representation of text that is concerned with the frequency of words within a document. We only keep track of word counts, ignoring grammatical details and any information about the order or structure of words. That's why it's called a "bag" of words. It is used as a feature in tasks such as topic identification and sentiment analysis.

3.5.6 Word Embeddings

Word Embeddings also called Distributional vectors are one of deep learning's key breakthroughs in difficult natural language processing problems. It is a method of representing words and documents in a numeric vector where words that have the same meaning have a similar representation. The most popular techniques of word embeddings are TF-IDF (Term Frequency — Inverse Document Frequency), Word2vec, and Glove (Global Vectors).

- **TF-IDF:**

TF-IDF stands for term frequency-inverse document frequency, which consists of two metrics: term frequency (TF) and inverse document frequency (IDF). It is a statistical measure used to find word relevance in text, which can be in the form of a single document or a collection of documents known as a corpus.

TF or Term frequency signifies the occurrence of a term (word) in a document. It is calculated by dividing the number of times a word appears in a document by the total number of words in the corpus.

$$TF(term) = \frac{\text{Number of times a word appears in a document}}{\text{total number of words in the corpus}}$$

DF or inverse document frequency assesses the significance of a word in the corpus. It is calculated by dividing the total number of documents by the number

of documents that contain a term.

$$IDF(term) = \log \left(\frac{\text{Total number of documents}}{\text{Number of documents that contain a term}} \right)$$

The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently, in general [43].

$$w_{x,y} = tf_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

Figure 1.17: TF-IDF [14].

- **Word2Vec :**

Word2vec is one of the most approaches to word embeddings. It generates word vectors, which are distributed numerical representations of word features. Word2vec transforms text into vectors that capture semantics and word relationships. Word2vec proposed two models CBOW and skip-gram. The CBOW model predicts the target word according to its context whereas The skip-gram model predicts a target word thanks to a nearby word.

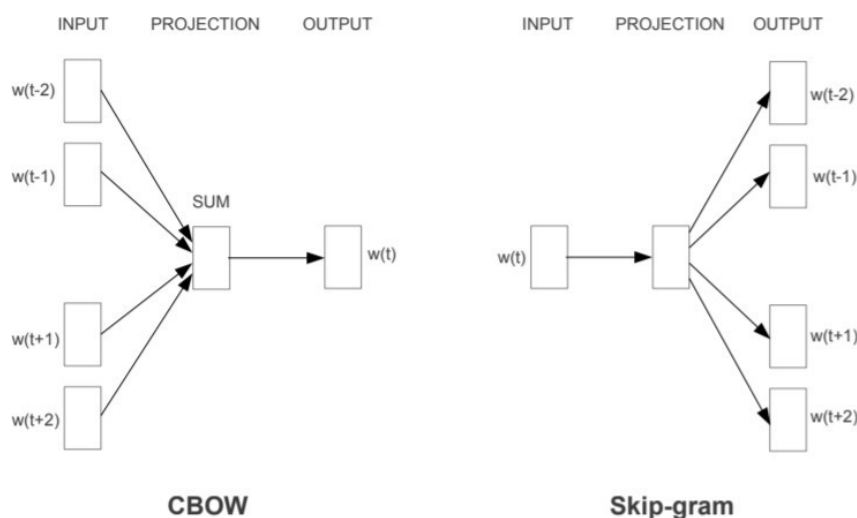


Figure 1.18: Word2Vec [15] .

- **GLOVE :**

Glove or Global Vectors is a kind of word embedding. It is a method developed at Stanford University, which focuses on the co-occurrence probability ratio between two words and encoded it as vector differences. The Glove is based on matrix factorization techniques, which determine the semantic relationship between words. The resulting embeddings highlight intriguing linear substructures of the word vector space.

In this technique, we take the corpus, reiterate through it, and get the co-occurrence of each word with other words in the corpus. We get a co-occurrence matrix through this. The words, which do come to each other, get a value of 1 if they're one word piecemeal also $1/2$ if two words piecemeal also $1/3$, and so on.

GLOVE and Word2Vec Both are fundamentally the same and give similar results for many tasks. The GLOVE model is based on global word-to-word co-occurrence counts across the entire corpus. However, Word2vec makes use of co-occurrence in the local context (neighboring words). This means that GloVe is a count-based model whereas Word2vec is a predictive model.

3.6 NLP APIs

Natural Language Processing APIs enable developers to integrate human-to-machine communications and perform a variety of useful applications such as speech recognition, sentiment analysis, chatbots, and so on. Some of the NLP APIs are:

- IBM Watson API,
- Cloud NLP API,
- Speech-to-text API,
- Translation API by SYSTRAN,
- Chatbot API,
- Text Analysis API by AYLIEN,
- Google Cloud Natural Language API,
- Sentiment Analysis API.

3.7 NLP Libraries

There are many natural language processing libraries, including the following:

- **Natural Language Toolkit (NLTK):** NLTK is the main tool for natural language processing categorizing text, analyzing linguistic structure, and more. It offers simple interfaces to over 50 corpora and lexical resources, including WordNet, as well as a suite of text processing techniques in Python, including tokenization, lemmatization, stemming, POS tagging, etc. NLTK is open-source and accessible for Windows, Mac OS, and Linux.
- **TextBlob:** It offers a simple interface for learning basic NLP tasks such as sentiment analysis, Part-of-Speech tagging, Parsing, and Noun phrase extraction.
- **Gensim:** Gensim is a famous Python library for performing natural language processing activities. It includes a special feature that uses vector space modeling and the topic modeling toolkit to determine the semantic similarity between two documents.

- **Spacy:** spaCy is a Python-based open-source NLP library. It is extremely fast and efficient, and it is focused on offering production software.
- **OpenNLP:** OpenNLP is a machine learning-based toolkit for natural language text processing. It can perform the most popular NLP tasks, as well as tokenization, part-of-speech tagging, sentence segmentation, named entity extraction, chunking, and parsing.

4 Computer Vision

4.1 Computer Vision

Computer vision is a field of artificial intelligence (AI) and machine learning (ML) that enables computers and systems to decide meaningful information from digital images, objects, videos, or other visual input and take the necessary action on that information. On the other hand, artificial intelligence allows computers to think and computer vision helps computers to observe and understand the visual world.

The main purpose of Computer Vision is to recognize and extract data/information and interpret them for our purposes. Typically, this entails developing methods to mimic the capability of human vision.

4.2 Computer vision and CNNs

Computer vision is required a large amount of data. It needs to analyze this data repeatedly until it detects different objects and, eventually, recognizes images. For example, we supply visual data from millions of cat images. In that case, the computer learns from the data by analyzing each photo, shape, and so on, and thus identifies patterns similar to cats and generates a model. As a result, for each input image, this computer vision model can now accurately detect whether or not it contains a cat. Two key techniques are used to achieve this: deep learning, and convolutional neural networks (CNNs).

With the help of pre-programmed algorithmic, a system of deep learning may automatically learn about the interpretation of visual data. The model can learn to distinguish between similar images if it's given a large enough dataset. Algorithms make it possible for the system to learn on its own, rather than for someone programming it in tasks like image recognition.

Convolutional neural networks help computer vision work by dividing images into smaller parts called pixels that are given markers. It performs convolutions with the help of these markers and makes predictions about what it is “seeing”. The neural network performs convolutions and evaluates the accuracy of its recommendations with each cycle.

4.3 Tasks in Computer Vision

Computer vision is the broad parent field for any calculations involving visual content – that means images, vids, icons, and anything different with pixels involved. This elaboration of machine vision saw the large-scale formalization in so many fields and uses.

Some common tasks for which computer vision can be used are as follows:

- **Image Classification:** Image classification is the simplest computer vision task used to identify the content of an image and then classify the objects into one or more different classes. It means that we can directly with accuracy predict the class of an object present in an image.

Image classifier principally takes an image as input and tells about different objects present in that image, similar as a person, tree, auto, etc. still, it would not give you other further information about the image data, similar to how numerous persons are there, tree, item positions, etc.

- **Image Segmentation:** Another important task of computer vision is image segmentation. It is the method used to divide an image into various regions to identify objects or boundaries to reduce the complexity of the image and analyze it more efficiently. Technically, segmentation is the process of classifying each pixel in an image to determine what objects or other important elements are present.

Image segmentation is used for many practical applications including medical, autonomous vehicles, image analysis, video surveillance, face recognition and detection, and satellite image analysis.

- **Object Detection:** Object detection is another popular computer vision technology that can be used after image classification. It is used to identify and locate objects in an image or video. This task can be used also to count objects in a scene, determine and track their exact locations, and label them accurately.

When humans look at images or videotape, we can detect objects of interest within a matter of moments. The thing of object discovery is to replicate this intelligence using a computer.

- **Object Recognition:** Object recognition is one of the most popular techniques of computer vision. It is used to recognize and identify the object's location concerning images or videos. When humans look at a snap or watch a videotape, we can readily spot people, objects, scenes, and visual details. The thing of work on object recognition technology is frequently to educate a computer to do what comes naturally to humans and the objects that people do.

This technique is useful in a variety of applications like driverless cars, robotic vision, and disease identification in bioimaging.

4.4 Computer vision Libraries

Many computer vision libraries are available such as OpenCV, SimpleCV, and scikit-image.

- **OpenCV:** (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS [44].



Figure 1.19: OpenCV [16] .

- **SimpleCV:** SimpleCV is an open-source framework for building computer vision applications. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap

storage. It's written in Python and runs on Mac, Windows, and Ubuntu Linux [17].



Figure 1.20: SimpleCV [17].

- **Scikit-image:** Scikit-image is a collection of algorithms for image processing in Python. It is available free of charge and free of restriction [18].

This image-processing library offers a well-documented API in Python and implements algorithms and utilities for use in research, education, and industry applications.



Figure 1.21: Scikit-image [18] .

4.5 Computer Vision Applications

Computer vision is one of the most advanced inventions of artificial intelligence and machine learning. It has a massive impact on different areas, including, healthcare, security, agriculture, automotive, etc. The following are some of the most common computer vision applications:

- **Radiology:** Computer vision is revolutionizing the radiology sector by making it more efficient and accurate and using it in several cases. Radiology is all about using medical imaging to test or analyze the human body. While the analysis and processing of medical data by humans are limited, Computer vision systems can analyze medical images like CT scans, X-rays, MRIs, mammograms, or surgery planning more accurately and quickly. This can help with a cancer diagnosis by detecting cancers that are not visible to the naked eye, as well as retinopathy screening for diabetics to prevent blindness.
- **Self-driving cars:**

Computer vision is the core of Self-driving car technology. It uses a combination of sensors, artificial intelligence, radars, and cameras making the entire driving process safer, easier, and without any human intervention. computer vision uses various algorithms to facilitate self-driving vehicles, which helps them to detect other cars and objects, read traffic signals, pedestrian paths, etc.,
- **Surveillance :**

With The demand for computer vision and its uses growing rapidly, surveillance is another application of it. This task is applied by making cameras everywhere, including, streets, roads, highways, shops, stores, etc., to decision support security monitoring in public places, critical infrastructure, etc, and to spot various doubtful or criminal activities.

Surveillance assists in providing live footage of public places to devious behavior, identify dangerous objects, and prevent crimes by upholding law and order.
- **Biometrics :** Computer vision is moving forward to play a bigger part in biometric and Fingerprint recognition technologies. It's the capability to identify or corroborate every individual rightly by using physiological(face, fingerprint, iris, particular region, win print, knuckle, observance, retina, footmark, gait, DNA, dental biometrics.).

The integration between biometrics and computer vision is directly linked with multiple operations like services, cyber security, and safety including facial recognition from a distance, intelligent videotape surveillance, 3D face modeling and recognition, biometrics for border control, and forensic biometrics...

- **Automatic Weeding :**

Automatic Weeding is one of many smart agriculture tasks realized through computer vision technology. It removes unwanted plants or weeds around the crops using robots, which cuts down costs and ensures higher yields. This method avoids different problems like contaminating neighboring healthy plants, water, or animals that happen by traditional methods (spraying pesticides).

- **Optical character recognition (OCR) :**

Optical character recognition, or OCR, is a significant area of research in artificial intelligence, pattern recognition, and computer vision. It is used to scan and convert any kind of image containing written text into machine-readable text data. It also allows us to extract text from documents such as invoices, bills, articles, and so on and compare it to databases.

OCR technology became prominent in different activities. It can be applied in products that have barcodes on their packaging to detect, verify, convert, and translate barcodes into readable text. In addition, it helps to identify wrongly labeled products, provide information about expiration dates, inform about product quantity in the magazine, and track packages at all stages of product development.

5 Conclusion

In conclusion, deep learning is the most emerging field of machine learning and has made significant contributions to several fields of research such as natural language processing and computer vision. Computer vision, powered by deep learning, has made remarkable strides in enabling machines to perceive and interpret visual information. Similarly, deep learning has transformed natural language processing, enabling machines to understand, generate, and interact with human language. In this chapter, we have introduced the important concepts related to deep learning (definition, deep neural network types...), natural language processing (like components, phases, libraries, etc.), and computer vision (tasks, applications ...). The next chapter deals with the details of the visual question-answering task.

Chapter 2

Visual Question Answering

1 Introduction

In an era where images and videos play a significant role in our lives, the ability to teach machines to understand and answer questions about visual content has become increasingly important. Visual question answering VQA addresses this challenge by leveraging deep learning techniques to build intelligent systems that can understand both images and text.

VQA combines the power of computer vision and natural language processing (NLP) to enable machines to comprehend visual information and provide meaningful answers to questions posed in human language. It represents a significant leap forward in bridging the gap between visual perception and linguistic understanding, unlocking new possibilities for human-machine interaction.

In this chapter, we delve into the field of Visual Question Answering (VQA), exploring its major issues such as evaluation, dataset bias, and various question types. We discuss the applications of VQA and highlight popular datasets used for training and evaluation. Additionally, we cover the approaches employed in VQA, including feature extraction and fusion techniques.

2 Visual question answering

In recent years, artificial intelligence has witnessed the emergence of multiple research problems, especially in the field of image and video explanation, which combines multi-discipline (Computer Vision (CV), Natural Language Processing (NLP), and Knowledge Representation & Reasoning (KRR)). This field is known as Visual Question Answering (VQA).

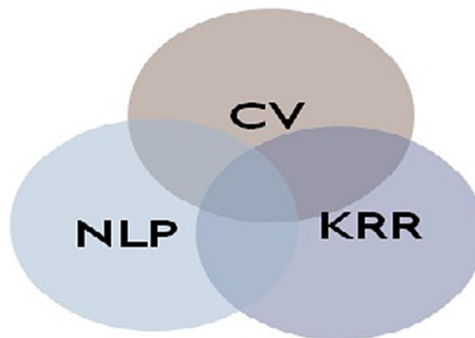


Figure 2.1: VQA as a multidisciplinary research problem [19].

In other words, VQA is a task based on various algorithms that take an image and a human language question about it as input and produces a natural language answer as output. It may also necessitate the use of knowledge representation and reasoning (KRR) strategies for natural language response generation.

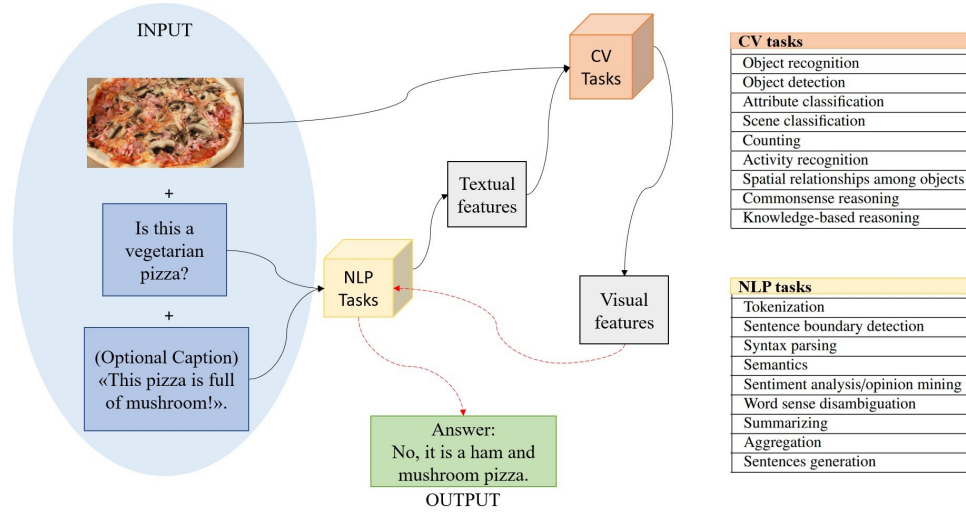


Figure 2.2: An example of a workflow of the VQA method is on the left and some of the CV tasks and NLP tasks involved in VQA are on the right. [20].

3 Major issues in visual question answering

3.1 Evaluation of open-ended and multiple-choice questions

Most of the state-of-the-art VQA systems have the capability to evaluate the multiple choice questions using conventional accuracy metrics. But, these systems are not capable enough to evaluate the open-ended questions. Also, there is an issue in the evaluation of multiple-choice questions as the problem is reduced to just determining the correct answer instead of actually giving the answer to the question. In the case of multiple-choice questions, choices must be given to the system in such a way that questions require reasoning about the content of an image rather than just interpreting the answer from the choices given. [19]

3.2 Dataset Bias

The bias in popular VQA datasets is one of the major issues in visual question answering. Some of the SOTA VQA models are more dependent on the question than the picture content. In some cases, these datasets' biases are used to the advantage of designing the prediction algorithm. Although bias can improve algorithm performance, this practice has an impact on the original research goals of created algorithms.

3.3 Balanced Binary Questions

The use of binary questions to evaluate VQA systems is widely used as they are easy to answer by human annotators and lack complex questions. But, creating balanced binary questions is a challenge as these questions are biased towards 'yes' in the majority of datasets like COCO-VQA [19].

3.4 Image Featurization from real VQA datasets

Real VQA datasets, such as VizWiz, contain low-quality images because they were captured by blind people. To answer compositional questions based on these low-quality images, a single network would not be sufficient to extract image features.

3.5 Need for application-oriented datasets

The majority of SOTA VQA models were evaluated using benchmark datasets such as DAQUAR and VQA. The models are incapable of dealing with real-world applications such as assisting blind people, assisting a data analyst in providing useful content from massive amounts of data, guiding children to learn a concept on smart devices, and communicating with a robot. As a result, more publicly available goal-oriented datasets, such as VizWiz and VQAMe, are required [19].

3.6 Questions requiring reading scene-text

In visual question-answering VQA tasks, there are many types of questions. One of them necessitates the ability to read the text present in images. Localization of scene text is a difficult and important task because there is no prior knowledge of the location, size, direction, typeface, layout, and color of texts in a scene image in general. Furthermore, recognition of scene text is difficult because of diverse exotic font styles. Moreover, due to blurring, perspective, low resolution, and non-uniform lighting, many characters are distorted, which extracts distorted and degraded visual features and generates incorrect

scene text detection and recognition.

3.7 Conversational questions

In real-world applications like helping visually-impaired users, the questions can be from verbal interactions. Such questions can have a large average length and may contain many uninformative terms. Removing these terms in a pre-processing step and learning techniques to better represent the question is a challenge [19].

3.8 The imbalance between answerable and unanswerable classes

The vast majority of publicly available VQA datasets are artificially created. In those kinds of datasets, every visual question has at least one correct answer and is answerable. However, in real-world applications, a question may not always have a correct answer. Apart from the VizWiz dataset, the other VQA datasets do not present unanswerable classes or samples [19].

4 Applications of visual question answering

VQA methods have multiple real-world applications.

First, one of the most important aspects of VQA is its close relationship to human perception and learning. An educational robot is developed by using VQA to ask questions and begin a teaching dialog based on the external environment. This robot helps the young kids to know about specific objects and activities.

Second, among the goals of several VQA applications proposed in recent years is assistance to blind and visually impaired people. VQA can respond to each day's questions that could assist visually impaired people to live without visual barriers.

Third, VQA has attracted great interest, especially in medical image understanding. It supports medical staff in making clinical decisions and improving patient engagement. The medical VQA can know free-form human language questions and generate reliable and user-friendly answers, whereas other medical AI applications are limited to pre-defined maladies or organ kinds.

Fourth, VQA can be used also in other areas such as smart cameras on food images, autumors driving, summarizing visual data for analysis, and interacting with, organizing, and navigating visual data.

5 VQA Datasets

As the VQA task has attached more attention from researchers in recent years, several datasets have been proposed for visual question answering. Below, we discuss some of them:

5.1 The COCO dataset

The COCO dataset contains images from the Microsoft Common Objects in Context (COCO), with 328,000 images having 91 object types that would be easily recognizable by a 4-year-old, with a total of 2.5 million labeled instances. The COCO dataset can simplify and accelerate the process of building a VQA dataset. However, it is not an easy task. For example, collecting varied, convenient, and nonambiguous questions is a great challenge. Besides the variety and precision problem, a good dataset also has to avoid biases that could be exploited. For example, given a dataset with only yes/no answers, where 90% of the answers are yes, a trivial most frequent class strategy will obtain an accuracy of 90%, without solving anything in terms of VQA. To get deeper into the VQA complexity, let's see some of the most important datasets released [21].



Figure 2.3: Examples of annotated images from the COCO dataset [21].

5.2 The DAQUAR dataset

The first significant VQA dataset was the DATaset for Question Answering on Real-world images (DAQUAR). It contains 6794 training and 5674 test question-answer pairs, based on images from the NYU-Depth V2 Dataset. That means about 9 pairs per image on average. Although it is a great initiative, the NYU dataset contains only indoor scenes with, sometimes lightning conditions that make it difficult to answer the questions. In fact, the evaluation on humans shows an accuracy of 50.2%. The other drawback of the DAQUAR dataset is that its size makes it unsuitable for training and evaluating complex

models[21].



Figure 2.4: Sample of images from the DAQUAR dataset [22].

5.3 The VQA dataset

The VQA Dataset includes both real images from MS-COCO and abstract clipart images. It is the most widely used dataset for VQA tasks and was released publicly as a part of the VQA challenge. VQA dataset for real images consists of overall 6.14.163 questions out of which 2.483.49 questions are used for training, 1.21.512 for validation, and 2.44.302 for testing. Questions for each image were generated by Amazon Mechanical Turk (AMT), and answers to these questions were given by a different group of workers. This dataset includes three questions corresponding to an image and ten answers corresponding to a given question. Ten independent annotators answered each question given to them. The total number of images is 2, 04,721. The longest question consists of 32 words and the longest answer is 20 words. VQA dataset for clipart images consists of 50,000 abstract images with 1, 50,000 questions. These abstract images are made from over 20 human cartoon models. 100 different objects and 30 different animal models. This dataset has both open-ended and multiple-choice questions for both types of images, i.e. real and clipart images. Eighteen different choices are also provided for the multiple-choice questions. These questions consist of the same QA pairs [19].



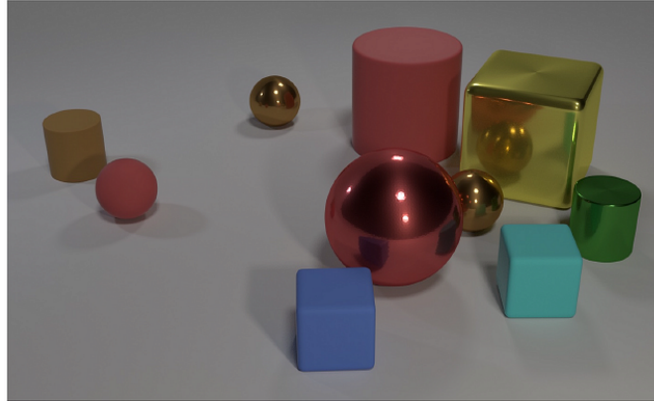
Figure 2.5: VQA Dataset [23].

5.4 The COCO-QA dataset

COCO-QA (Ren et al., 2015) The COCO-QA dataset is another dataset based on MS-COCO. Both questions and answers are generated automatically using image captions from MS-COCO and broadly belong to four categories: object, number, color, and location. There is one question per image and answers are single-word. The dataset contains a total of 123,287 images. Evaluation is done using either accuracy or WUPS score[45].

5.5 The CLEVR dataset

CLEVR is a synthetic dataset to test the visual understanding of the VQA systems. The dataset is generated using three objects in each image, namely a cylinder, sphere, and cube. These objects are in two different sizes, two different materials and placed in eight different colors. The questions are also synthetically generated. Based on the objects placed in the image. The dataset also accompanies the ground-truth bounding boxes for each object in the image [46].



Q: Are there an **equal number** of large things and metal spheres?
 Q: What size is the cylinder **that is left of the brown metal thing that is left of the big sphere**? Q: There is a sphere with the **same size as the metal cube**; is it **made of the same material as the small red sphere**?
 Q: **How many** objects are either **small cylinders or metal things**?

Figure 2.6: CLEVR Dataset [23].

5.6 The KVQA dataset

The recent interest in common-sense questions has led to the development of a Knowledge-based VQA dataset. The dataset contains questions targeting various categories of nouns and also requires world knowledge to arrive at a solution. Questions in this dataset require multi-entity, multi-relation, and multi-hop reasoning over large Knowledge Graphs (KG) to arrive at an answer. The dataset contains 24,000 images with 183,100 question-answer pairs employing around 18K proper nouns[46].

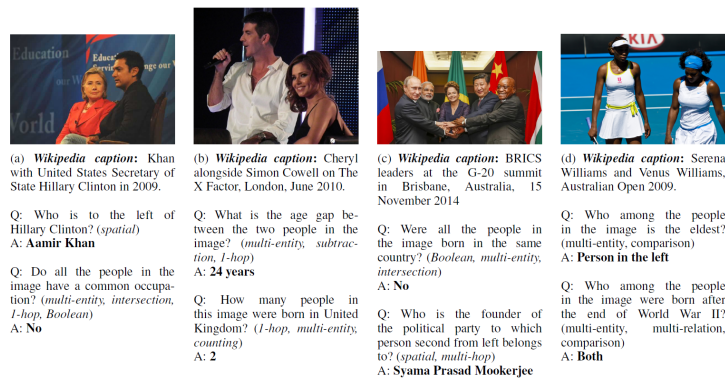


Figure 2.7: K-VQA dataset[24].

6 The approaches in VQA

In the VQA task, to answer a question that is related to the image, we need to understand the content of the image and correlate it with the context of the question asked. In other words, we need first to extract features from the question and the image after that combine them to generate an answer.

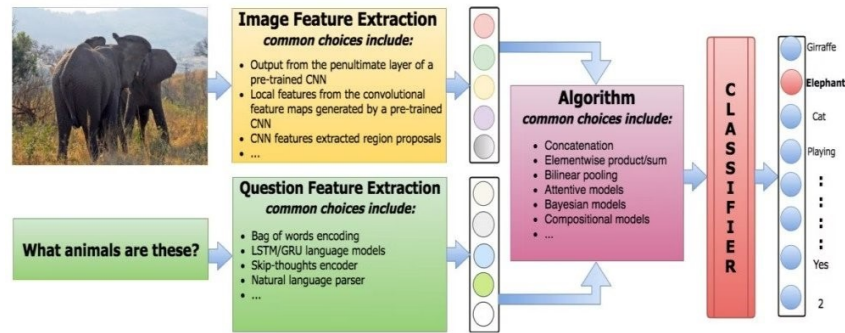


Figure 2.8: Simplified scheme of VQA approaches [21].

6.1 Feature extraction

Feature extraction from questions, images, and videos plays an important role in VQA. It is a technique used to reduce a large input data set into meaningful features for processing and training models. Deep learning models have been shown to be very successful at extracting complex and discriminative features in recent years, and they have also become the dominant approaches in the domain of VQA.

6.1.1 Image Featurization

Image feature extraction is one of the most important tasks of VQA. It is the process of representing an image as a vector in order to perform a variety of mathematical operations on it. For extracting image features, various convolutional neural networks (CNNs) such as LeNet, AlexNet, GoogLeNet, VGG-Net, and ResNet have been developed.

The LeNet model consists of two convolutional layers and a simple network structure. It is primarily used for the task of recognizing handwritten digits. AlexNet, a deeper network with 5 convolutional layers, was the first deep network to significantly improve classification accuracy. It used ReLu (Rectified Linear Unit) as the activation function,

which can be trained much more quickly than sigmoid or tanh. It wins the ILSVRC2012 championship (ImageNet Large Scale Visual Recognition Challenge). In ILSVRC2014, GoogLeNet and VGG-Net were two of the best models for image classification, and consequently, they have become the new mainstream approaches to automatically extracting effective image features. Both of them deepen the convolutional network with more layers to capture more informative features. Moreover, GoogLeNet also increases the width of the network. Instead of concatenating convolved features directly, it applies 11 convolution to reduce dimension and accelerate the training speed [47].

6.1.2 Question featurization

Text features in VQA models can be extracted by using techniques such as Bag-Of-Words (BOW) or Long Short-Term Memory (LSTM). In addition, VQA systems use word embeddings to generate a numeric feature vector for a given natural language question in order to easily it handled by computers. Word embeddings are mainly employed for generating language models and extracting textual features in natural language processing (NLP). Moreover, it is used to extract morphological, semantic, and contextual information. Also, LSTM, GRU, and Bi-LSTM (RNN variants) are most effective when dealing with sequential data in many models.

6.2 The fusion block

The most important component of the VQA task is the fusion block. The fusion block receives both the question and image features as inputs and outputs the fused multi-modal feature. This combination is accomplished through various methods such as attentive models, elementwise multiplication of image and question feature vectors, and bilinear pooling.

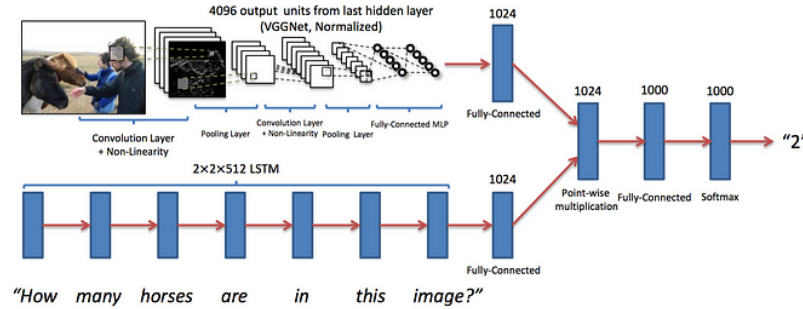


Figure 2.9: Combining CNN/RNN for VQA[23] .

Baseline fusion methods include concatenation, element-wise addition, and element-wise multiplication where the last two require compatibility in the feature vector dimensions. Malinowski et al. (2017), tried all these three multimodal fusion methods and found that element-wise multiplication has more accuracy. Another important finding is that the L2 normalization of visual features has a significant impact on fusion performance, especially for concatenation and summation. According to their results, summation has high accuracy after normalization. Shih et al. (2016) use the dot product of region-wise visual features and question embeddings. Saito et al. (2017) introduced a hybrid way of multimodal fusion. They integrate element-wise summation and element-wise multiplication by implementing a polynomial function. Lioutas et al. (2018) proposed an ensemble attention model where multimodal fusion is done via the concatenation of question, answer, and image features where question and answer features are first multiplied element-wise[48].

In the bilinear pooling method, the outer product between two vectors is computed to provide all possible interactions between each element of both vectors. A huge amount of memory and high computation cost occurs in order to compute the interaction between each pair of visual feature and textual feature vectors. To handle the issue of large parameters, proposed Multimodal Compact Bilinear Pooling (MCB) to compact bilinear models. They used the count sketch function and Fast Fourier Transformation (FFT) to avoid computing outer products explicitly [19]. **There** are other models used for the fusion block like fusion based on neural networks, and attention-based models.

7 Conclusion

In conclusion, this chapter provided an overview of Visual Question Answering (VQA), a multidisciplinary field that combines computer vision and natural language processing. We explored the definition of VQA, which involves generating accurate answers to questions based on visual content. Throughout the chapter, we discussed the major issues in VQA, including the challenges posed by the ambiguity of natural language questions and the complexity of visual content.

The chapter highlighted the diverse applications of VQA, ranging from assistive technologies for visually impaired individuals to smart cameras, medical image understanding, and robotics. These applications demonstrate the potential impact of VQA in enhancing human-computer interaction and understanding visual content. Furthermore, we discussed various datasets that have been curated specifically for VQA research, such as VQA, VQA 2.0, CLEVR, COCO-QA, and KVQA. These datasets provide standardized benchmarks for evaluating the performance of VQA models and offer a wide range of question types and visual concepts.

Lastly, we explored different approaches in VQA, including feature extraction and fusion methods. As the field of VQA continues to evolve, we can unlock new applications and pave the way for more sophisticated and context-aware visual question-answering systems. The next chapter will focus on the object detection task.

Chapter 3

Object Detection in Computer Vision

1 Introduction

In recent years, the challenge of object detection has developed rapidly and left a profound impact on the entire field of computer vision. It stands as a fundamental and captivating task that enables machines to perceive and interpret the visual world. It empowers a wide range of applications, including autonomous robotics, video surveillance, image understanding, and content-based image retrieval.

In this chapter, we delve into object detection, a crucial component of Visual Question Answering (VQA). We explore the principles and applications of object detection, covering important concepts such as RoI, IoU, NMS, mAP, and bounding-box regression. We discuss two-stage algorithms like R-CNN, SPP-Net, Fast R-CNN, and Faster R-CNN, as well as one-stage algorithms such as YOLOv1, YOLOv2, YOLOv3, SSD, and YOLOv4.

2 Object detection

Object detection is one of the most active areas of research in the field of computer vision (CV). It allows for identifying and locating objects in images or videos. Specifically, object detection draws bounding boxes around detected objects, allowing us to locate those objects (or know how they are moving) in a given scene.

Object detection is often confused with image recognition. Object recognition is the process of identifying objects in images and videos. Object Detection, on the other hand, specifies the location of multiple objects in an image.

With the increased use of object detection in several applications and areas such as video surveillance, robotics, self-driving, etc. It has become necessary to develop more accurate and faster systems. New algorithms continue to outperform old ones in terms of speed and accuracy.

3 The principle of object detection

The principle of object detection is as follows: for a given image, we divide it into various regions. We will then consider each region as a separate image and classified using an image classification model (classify them into various classes). The regions of the original image with good classification results are kept and the others are thrown. Thus, to have a good method of detecting objects, it is necessary to have a good image classification algorithm, and a strong region detection algorithm.

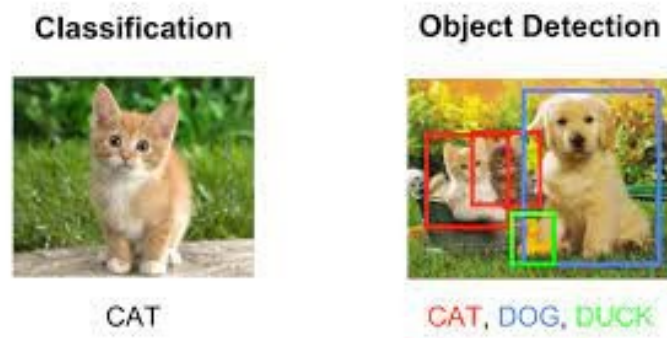


Figure 3.1: The principle of object detection [25].

4 Applications of object detection

Object detection has a wide range of applications in real-world scenarios and various fields, including:

- Military,
- Robotics,
- Analysis of medical images,
- The autonomous car,
- And driver drowsiness detection on the highway.



Figure 3.2: Examples of areas of application [26].

5 Important Object Detection Concepts

5.1 Région d'intérêt (RoI)

Region of interest also known as RoI, or bounding box proposal, is a proposed region from the original image. This means it is a rectangular region containing an object or not. There are multiple methods and algorithms to generate these proposals like selective search, region proposal network (RPN), and edge box detection (Edge Box Detection). A bounding box is represented as a 4×1 vector containing its central location, its width, and its height (x, y, w, h) . By using this technique in object detection models, multiple instances of different types of objects in the same image are detected and provided with coordinates in the image where these objects are located.

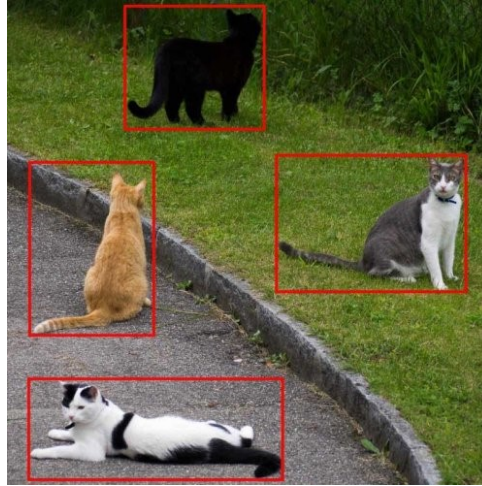


Figure 3.3: Regions of Interest [27] .

5.2 Intersection Over Union (IoU)

Intersection over Union or IoU, is a measure based on the Jaccard index that quantifies the degree of overlap between two boxes, the predicted bounding box $\mathbf{B_p}$ and the true bounding box $\mathbf{B_a}$. It is defined as follows:

$$IoU = \frac{\text{Area of intersection of 2 boxes}}{\text{Area of union of 2 boxes}}$$

$$IoU(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

By convention, $\text{IoU} \in [0, 1]$ and the prediction $\mathbf{B}_p$ of a bounding box is considered satisfactory if we have $\text{IoU}(\mathbf{B}_p, \mathbf{B}_a) \geq 0.5$.

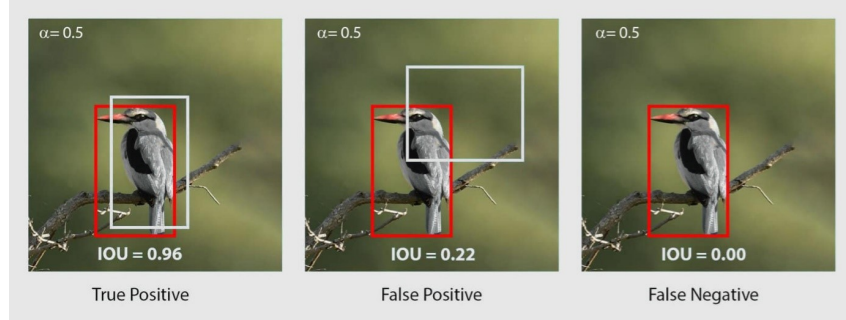


Figure 3.4: Object detection predictions based on IoU [28] .

5.3 Non-Max Suppression (NMS)

Non-max suppression is a technique used in object detection that aims at removing overlapping bounding boxes that predict the same object and selecting the most relevant areas.

This technique is based on two iterative steps. Firstly, we choose the area with the greatest probability of prediction, and then we remove any area having $\text{IoU}(B_p, B_a) \geq 0.5$ with the area chosen previously.

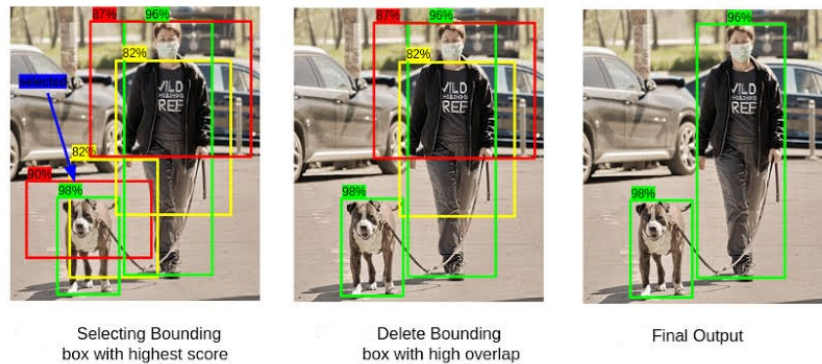


Figure 3.5: NMS Application Process [29] .

5.4 Mean Average Precision (mAP)

Mean Average Precision or mAP is a performance metric used for evaluating object detection models. It is the most widely used measure in standard challenges such as PASCAL VOC, COCO, ImageNET challenge, Google Open Image Challenge, and others. Mean Average Precision (mAP) is built around the following sub-metrics:

- Intersection over Union(IoU),
- Confusion Matrix,
- Precision,
- And recall.

5.5 Bounding-box regression

Bounding box regression is a popular technique used in object detection approaches. It allows the refinement or predicts localization boxes in an image by training a model to predict the coordinates of the bounding box surrounding the object. To estimate the confidence in the existence of an object in the box, a bounding box regressor is frequently followed by a bounding box classifier (confidence score). It is also possible for the classifier to be class-specific or class-independent.

6 Object detection algorithms

Because CNNs have strong feature representation power, they are used in object detection architectures, which are classified into two types. The first is the two-stage object detection architecture, which separates the object location and classification tasks. It creates the region proposal first, then classifies it. The second type is one-stage object detection architecture, which locates and classifies objects directly using CNNs without separating them into two parts.

6.1 TWO-STAGE TARGET DETECTION FRAMEWORK

6.1.1 R-CNN

In 2014, AI researchers at the University of California, Berkley proposed the Region-based Convolutional Neural Network (R-CNN) [49]. The R-CNN is composed of three modules. First, a region selector employs "selective search," an algorithm that

finds regions of pixels in an image that may represent objects, also known as "regions of interest" (RoI). For each image, the region selector generates approximately 2,000 regions of interest. The RoIs are then warped to a predetermined size and fed into a convolutional neural network. The CNN extracts feature from each region separately using a series of convolution operations. In the end, a classifier machine learning model maps the encoded CNN features to the output classes. The classifier generates a separate output class for "background," which represents anything that isn't an object.

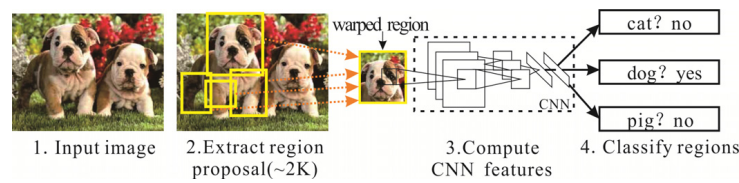


Figure 3.6: R-CNN architecture [30].

6.1.2 SPP-Net

In 2015, the Spatial Pyramid Pooling (SPP) model was proposed by He [50]. It solves the problems of low detection efficiency and the need for fixed input-size image blocks in R-CNN. This algorithm extracts the features of the region's proposal on the feature map after the original image has passed through the convolution layer, and all the convolution calculations are performed only once. At the same time, the spatial pyramid pooling layer is added after the last convolutional layer, and the feature of the region proposal is passed through the spatial pyramid pooling layer to extract the feature vector of fixed size. Compared to the R-CNN, Spp-Net performs feature extraction on the entire image only once, avoiding repeated calculations. However, it still has the same shortcomings as R-CNN: 1) Multi-step training steps are complicated. 2) Separate SVM classifiers need to be trained and additional regressors are required [30].

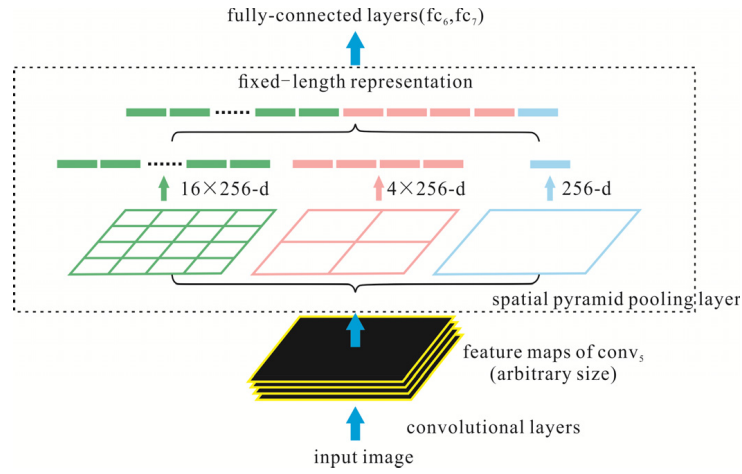


Figure 3.7: SPP-Net architecture [30].

6.1.3 Fast R-CNN

Given R-CNN success, Ross Girshick, then at Microsoft Research, proposed an extension to it in a 2015 paper called "Fast R-CNN" [51]. In this model, we generate region proposals based on the network's last feature map rather than the original image. As a result, we only need to train one CNN for the entire image. In addition, it replaced the SVM used in R-CNN with a softmax function that outputs the class probabilities directly. Now, instead of one neural net and many SVMs, we only have one neural net to train. In terms of speed, fast R-CNN performed significantly better. There was only one major bottleneck left: the selective search algorithm used to generate region proposals.

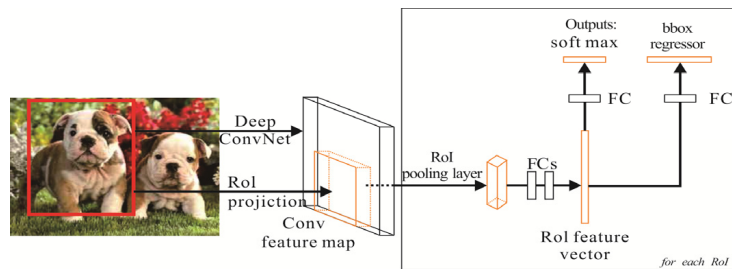


Figure 3.8: Fast R-CNN architecture [30].

6.1.4 Faster R-CNN

Faster R-CNN, which was introduced in 2016 [52], consists of incorporating the region extraction mechanism into the object-detection network. It uses an image as input and

generates a list of object classes and the bounding boxes for each. The model is divided into two modules, one of which module is called the region proposal network (RPN) a component that takes a CNN for proposing regions and the type of object to consider in the region. The other module is the Fast R-CNN detection algorithm, a CNN for extracting features from the proposed regions and outputting the bounding box and class labels. Faster R-CNN achieves near-real-time object detection speed by incorporating region detection into the main neural network architecture.

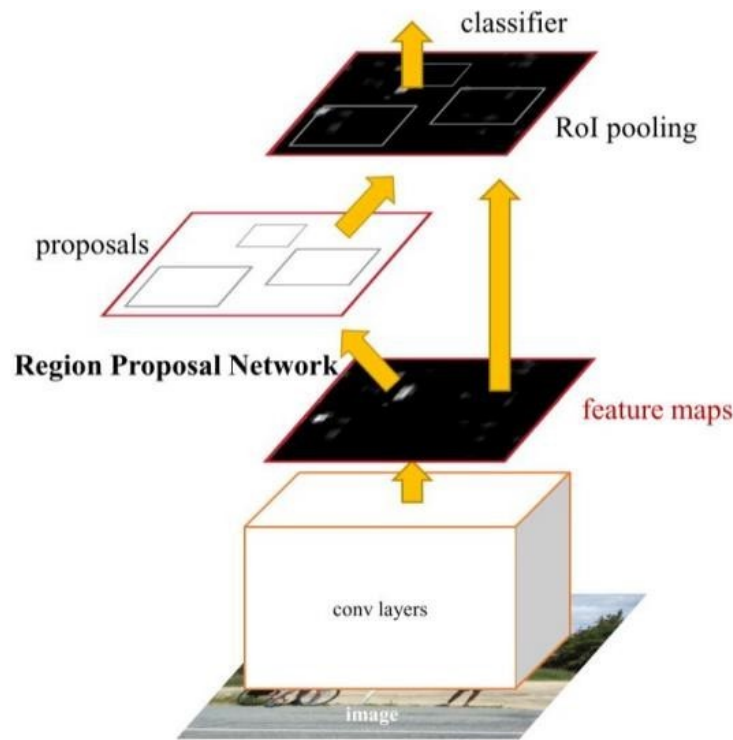


Figure 3.9: Faster R-CNN architecture[31] .

6.2 ONE-STAGE TARGET DETECTION ALGORITHM

6.2.1 YOLOv1

YOLOv1 or You Only Look Once is a single-stage object detection model, which was proposed by Joseph Redmon in 2016 [53]. YOLO suggests having a unified network to perform all tasks at once. It is also possible to create an end-to-end training network. An $S \times S$ grid ($S=7$) is created from the input image. If an object's center falls into a grid cell, that grid cell is responsible for detecting it. YOLO produces fewer background errors but

performs poorly when dealing with objects in groups.

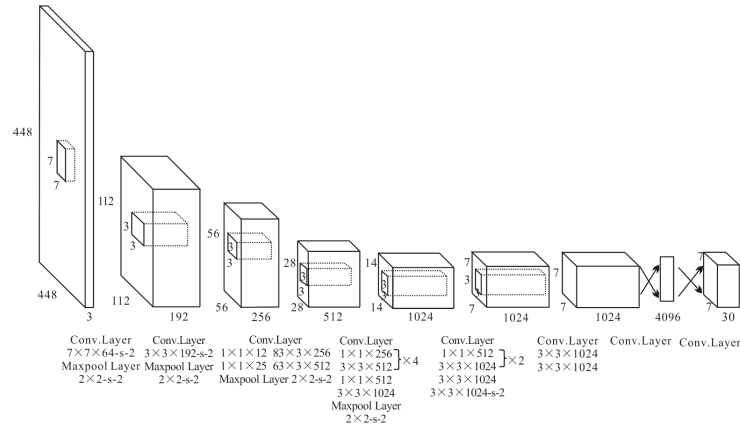


Figure 3.10: YOLOv1 architecture [30] .

6.2.2 YOLOv2

In 2016, Redmon proposed the YOLOv2 model. The main goal is to improve recall and localization while maintaining classification accuracy. YOLOv2 uses a new fully convolution feature extraction network Darknet-19, which contains a total of 19 convolutional layers and 5 maximum pooling layers. By adding a batch normalization layer to the convolutional layer and removing dropout, introducing an anchor box mechanism, using k-means clustering on the training set bounding box, and multi-scale training, the recall and accuracy are significantly improved. However, the detection of targets with high overlap and smatargetsget still needs to be improved [30].

6.2.3 YOLOv3

YOLOv3, in 2018 Redmon et al. further optimized the network structure and loss function, etc., based on YOLOv2 to make the model more accurate and robust. In contrast to YOLOv2, the backbone of YOLOv3 is DarkNet-53, which has a deeper network structure and uses a residual structure to prevent gradient disappearance. The Feature Pyramid Network (FPN) structure is also used to extract multi-scale features from images, which effectively improves the detection accuracy of small objects [54].

6.2.4 SSD

SSD or for Single-Shot Detector which was proposed by W. Liu et al in 2015[55]. With SSD, we only need to take one shot to detect multiple objects within the image. While

RPN-based strategies, such as faster R-CNN, requires two shots: one for producing region proposals and one for detecting the object of each proposal. The SSD detector is simple to learn and incorporate into software devices that need an object detection component. SSD has much good accuracy than other single-stage methods, even with smaller input image sizes.

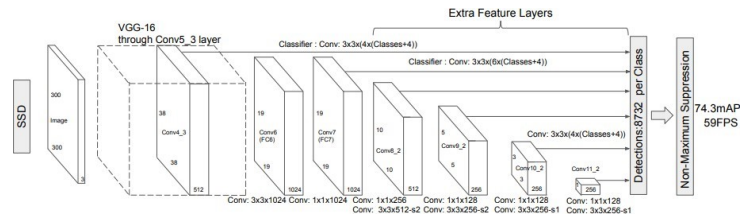


Figure 3.11: SSD architecture [32].

6.2.5 YOLOv4

Bochkovskiy et al. proposed YOLOv4 model in 2020. YOLOv4 was designed to achieve goals of higher speed and accuracy. Various network optimization solutions are used to improve network performance, such as increasing receptive fields, data augmentation, and changing the loss function. The backbone network was replaced with CSPDarkNet-53, which was experimentally found to be more suitable for object detection networks. Multichannel fusion using Path Aggregation Network (PANet) discards the previous FPN to improve the receptive field of the input images [54].

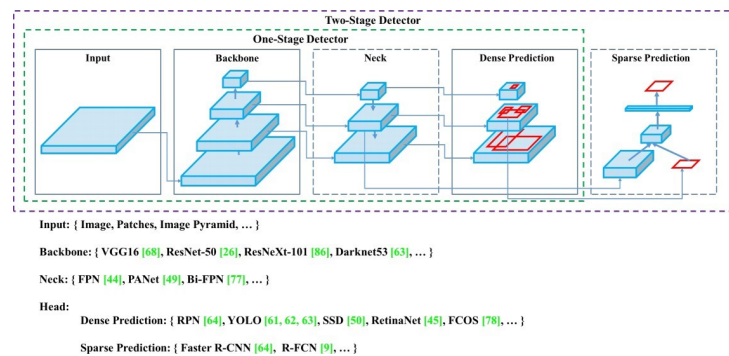


Figure 3.12: Architecture global YOLO v4 [33] .

7 Conclusion

In conclusion, object detection is a fundamental task in computer vision that has revolutionized the way we analyze and understand visual data. Throughout this chapter, we have explored the definition, principles, and applications of object detection. We have delved into important concepts that form the foundation of object detection. Concepts such as non-maximum suppression (NMS), region proposal networks (RPNs), and Mean Average Precision (mAP). Additionally, we have explored various object detection algorithms deep learning-based like R-CNN, Faster R-CNN, and YOLO. In the end, object detection continues to be a rapidly evolving field, with ongoing research and development aimed at further enhancing its accuracy, efficiency, and applicability. As technology progresses, object detection will play an increasingly vital role in various domains, shaping the way we interact with and understand visual data. The next chapter will be devoted to our realized work and the experimental results

Chapter 4

Implementation and Results Analysis

1 Introduction

In this chapter, we present the implementation details and provide an analysis of the obtained results for our approach applied to the Visual Question Answering dataset version 2 (Our approach is based on integrating object detection features into the VQA pipeline). We describe the steps involved in processing the data, building the vocabulary, encoding the questions, and integrating image presentation and question presentation. Additionally, we discuss the fusion process, MLP classifier, and output generation in our model. The chapter also includes the training and validation process, along with an overview of the development environments, hardware and software environments, choice of programming language, and the libraries and software tools utilized.

2 Dataset Overview

Our approach applied to visual question answering dataset version 2 (VQA V2 dataset).

The VQA V2 dataset is a widely used benchmark dataset for Visual Question Answering tasks. It consists of a large collection of images paired with human-annotated questions and corresponding answers. It is built on the Microsoft Common Objects in Context (MSCOCO) image dataset and divided into training, validation, and test sets.

The VQA v2 dataset contains a variety of question types, including counting questions (like “How many people are in the picture?”), yes or no questions (like “Is there a dog in the picture?”), and other questions (like “What color is the car?”). The questions in the dataset are designed to test a model’s ability to recognize entities and objects, reason about relationships between them, and understand natural language. The answers to questions are of a single-word type, and most answers in the dataset tend to cluster into a manageable set of a few thousand answers.

The VQA V2 dataset presents several challenges for researchers and developers working on visual question-answering systems. One of the main difficulties is the lack of image-text semantic alignments, as the VQA system consists of two distinct data streams (textual and visual data) that should be used and combined correctly to ensure robust performance. Additionally, most VQA algorithms consider the answer generation process as a classification problem, and each unique answer is treated as a distinct category, which limits the range of possible answers. The large size of the dataset, which contains more than 250K images and 1.1M questions, also makes it challenging to

process and analyze. Another challenge is the data imbalance problem, which was addressed in the VQA V2 dataset by collecting complementary images to balance the dataset, but there may still be some linguistic bias in the dataset.

Furthermore, the VQA V2 dataset also includes some noisy challenges and an absence of image caption challenges. The noisy challenge refers to the presence of noise in the dataset, such as incorrect annotations or irrelevant questions, which can affect the accuracy of VQA models. The absence of an image caption challenge refers to the fact that the VQA V2 dataset does not include image captions, which can provide additional contextual information to help answer questions. However, some researchers argue that the absence of image captions makes the VQA task more challenging and closer to real-world scenarios, where images are often encountered without captions. Despite these challenges, the VQA V2 dataset remains a valuable resource for developing and evaluating VQA systems, and it continues to inspire considerable research in the field.

Finally, VQA is a challenging task that involves capturing the high-level interactions between a question and an image to predict an accurate answer, which requires combining image and language understanding and guiding the attention of the model around the most informative regions of the image based on the asked question.

3 Our model

In our independent project, we took on the challenge of addressing the object detection aspect of Visual Question Answering (VQA) by developing a neural network model (ODNet model). The framework of the proposed model as shown in Figure 4.1 consists of five main phases namely: Image presentation, Question presentation, Fusion process, MLP Classifier, and Output generation.

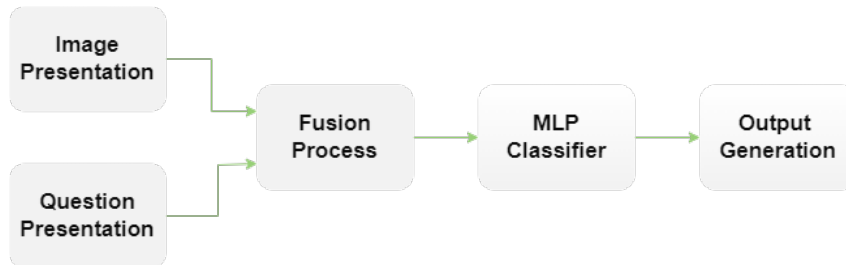


Figure 4.1: Our model.

3.1 Image presentation

In our research project, we used the grounded visual features (object detection features) generated by the bottom-up attention model [56] as image features. These grounded visual features are generated using the Faster R-CNN object detection model with a ResNet-101 backbone and the bottom-up attention mechanism.

Before using these grounded visual features as input to our VQA model, we perform pre-processing steps to ensure their suitability and compatibility. The purpose of pre-processing is to enhance the quality and usefulness of the visual features for our VQA task.

Firstly, we initialize an HDF5 file using `h5py.File()` and inside them, four datasets are created. These datasets will store the pre-processed image features, image IDs, bounding box coordinates, image widths, and image heights, respectively. The instructions below discuss this step.

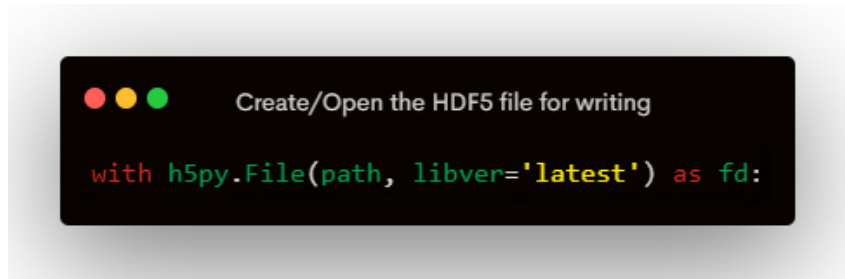


Figure 4.2: Accessing and Manipulating the H5 file.

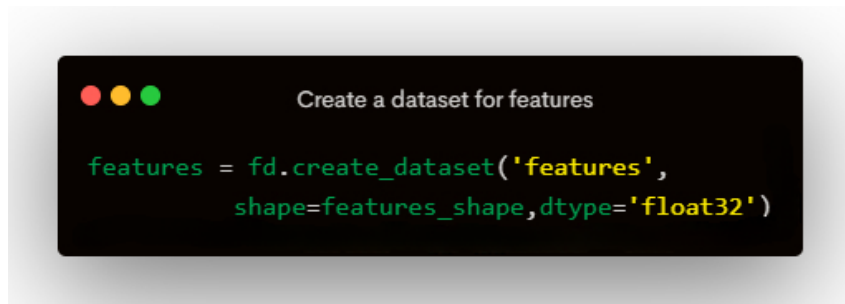


Figure 4.3: Creating a Dataset for Features using “fd.create_dataset”.

Then, we read the input TSV files (the file containing the grounded visual features and

their associated metadata such as image dimensions and box coordinates) using the CSV module and iterate over the rows of the TSV files.

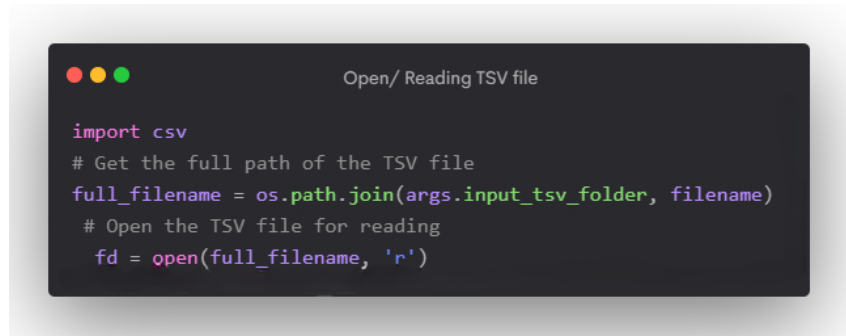


Figure 4.4: Reading and Accessing TSV File .

For each row, we extract the image ID, image width, image height, image features, and bounding box coordinates. The image ID, width, and height are stored in the `coco_ids`, `widths`, and `heights` datasets, respectively.



Figure 4.5: Extracting and storing the image Widths and Heights into their dataset .

The image features and bounding box coordinates are decoded from base64 and reshaped into the appropriate dimensions using the NumPy module. The reshaped arrays are then stored in the `features` and `boxes` datasets, respectively.



Figure 4.6: Encoded features to store them in the dataset .

Here is a table showcasing each dataset created in the HDF5 file and their respective properties:

Dataset Name	Description
features	The features dataset contains the image features. It has a data type of float32 and a shape of (123287, 2048, 36). Where the first dimension is the number of images in the dataset, the second dimension is the dimensionality of the image features, and the third dimension is the number of bounding boxes per image.
coco_ids	The coco_ids dataset contains the image ID. It has a data type of int32 and a shape of (123287,).
boxes	The boxes dataset contains the bounding boxes. It has a data type of float32 and a shape of (123287, 4, 36). Where the first dimension is the number of images in the dataset, the second dimension is the number of coordinates per bounding box, and the third dimension is the number of bounding boxes per image.
widths	The widths dataset contains the image width. It has a data type of int32 and a shape of (123287,).
heights	The heights dataset contains the image height. It has a data type of int32 and a shape of (123287,).

Table 4.1: Datasets of H5 file and their properties.

By executing this pre-processing step, we ensure that the visual input is transformed into a suitable format for our VQA model. The extracted features and bounding boxes will serve as valuable input during the training and validation phases, enabling our model to reason and answer questions based on the visual content of the images.

3.2 Question presentation

3.2.1 Preprocess

In the context of our VQA (Visual Question Answering) model, the pre-processing of questions and answers plays a vital role in enabling effective analysis and understanding of the input data. By performing several essential tasks such as data loading, vocabulary building, question indexing, and data writing, it ensures that the input data is in the appropriate format and ready for further processing.

- **Loading the data from JSON files:**

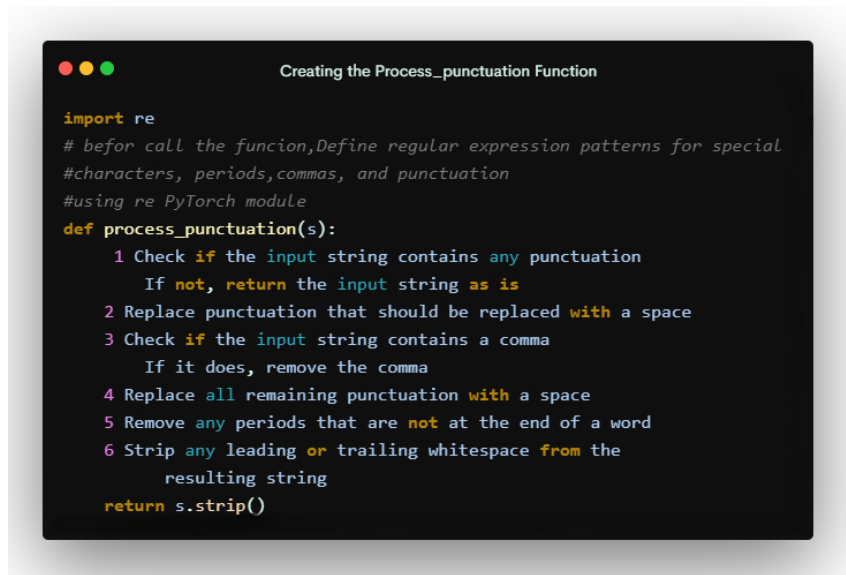
The first step involves loading the data from JSON files containing questions and annotations. These files are parsed using the `json.load()` function, enabling extraction of the questions and annotations into separate lists. This initial data acquisition sets the foundation for subsequent operations.



Figure 4.7: Loading the annotations and questions from multiple JSON files .

- **Building the vocabulary:**

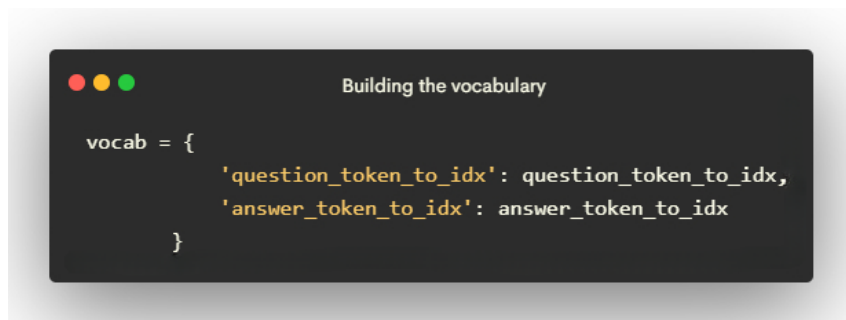
This step focuses on building the vocabulary required for question and answer processing. The answers are processed by removing special characters and punctuation marks using the `process_punctuation()` function. Each of these selected answer tokens is then assigned a unique index.



```
import re
# before call the function, Define regular expression patterns for special
# characters, periods, commas, and punctuation
# using re PyTorch module
def process_punctuation(s):
    1 Check if the input string contains any punctuation
      If not, return the input string as is
    2 Replace punctuation that should be replaced with a space
    3 Check if the input string contains a comma
      If it does, remove the comma
    4 Replace all remaining punctuation with a space
    5 Remove any periods that are not at the end of a word
    6 Strip any leading or trailing whitespace from the
      resulting string
    return s.strip()
```

Figure 4.8: Creating the function of Punctuation for handling punctuation and stripping Marks .

Similarly, the questions are processed by converting them to lowercase, removing special characters, and splitting them into tokens. Each unique question token is assigned a unique index.



```
vocab = {
    'question_token_to_idx': question_token_to_idx,
    'answer_token_to_idx': answer_token_to_idx
}
```

Figure 4.9: Creating a vocabulary dictionary .

The generated vocabulary is then written to a JSON file, which will be used for subsequent steps. This vocabulary-building step ensures that the textual data is represented in a structured numerical format, enabling efficient processing by the VQA model.

- **Questions Indexing:**

The questions indexing involves converting the questions into sequences of indices based on their respective token mappings from the vocabulary. The instructions below define the encode function of the questions indexing process:

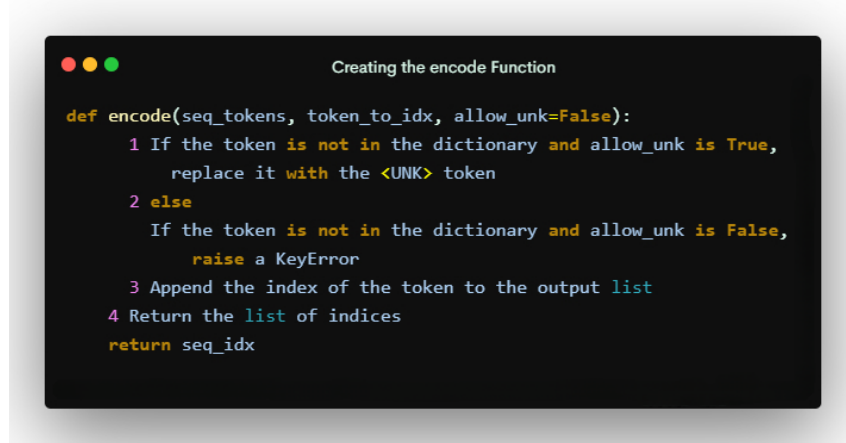


Figure 4.10: define the encode function for indexing questions .

For training and validation modes, the answers are also indexed by mapping them to their respective indices from the $answer_{token} \rightarrow o_idx$ mapping. To ensure uniformity, the indexed questions are padded to have the same length as the longest question. This step enables consistent input data for the VQA model.

- **Word Embeddings:**

In our work, we leverage the power of word embedding to represent questions as vectors. To achieve this, we use pre-trained GloVe 300d embeddings, which are trained on a large corpus of text to generate word embeddings of 300 dimensions. These embeddings offer a robust solution for transforming textual input into numerical representations. By mapping words to dense vectors in a continuous space, GloVe embeddings encode both the syntactic and semantic information of the words.

This enables our model to understand the context and nuances of the questions more effectively. By utilizing pre-trained GloVe embeddings, we can save time and resources required for training word embeddings from scratch, and leverage the extensive knowledge acquired from training on large text corpora.

By loading these embeddings from the specified file path and mapping them to our VQA model's vocabulary, we create a *glove_matrix*. The *glove_matrix* serves as a pivotal component by storing the GloVe embeddings for each token in the vocabulary.

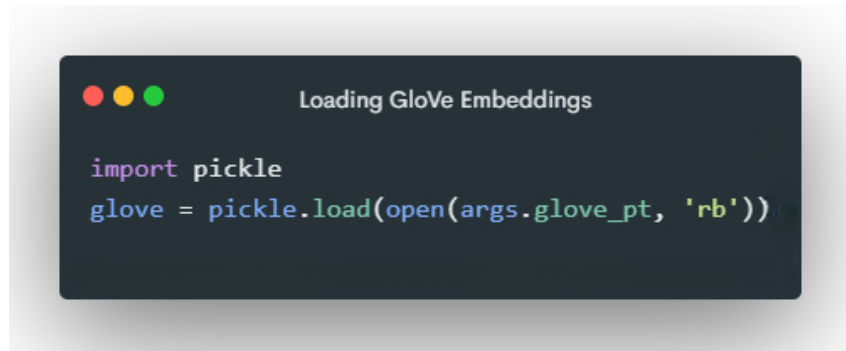


Figure 4.11: Loading GloVe Embeddings from Pickle file .

The *glove_matrix* is converted into a NumPy array of type `np.float32` using `np.asarray()` to facilitate their integration into the VQA model.

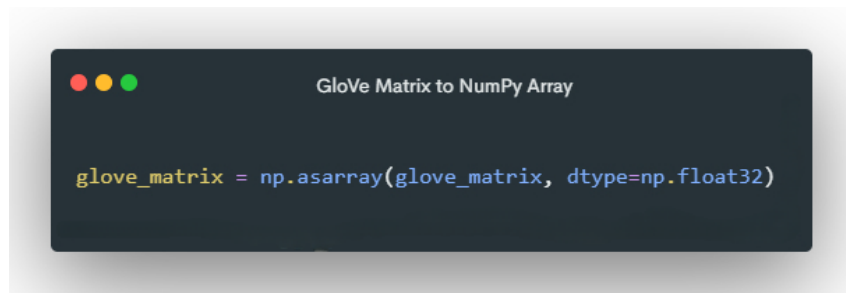


Figure 4.12: Converting GloVe matrix to NumPy array .

- **Save data:**

The processed data, including the indexed questions, question lengths, image IDs, answers (for training and validation modes), and GloVe embeddings, are written to a pickle file specified by *output_p.t*. This file serves as the preprocessed data input for the subsequent stages of the VQA model pipeline.

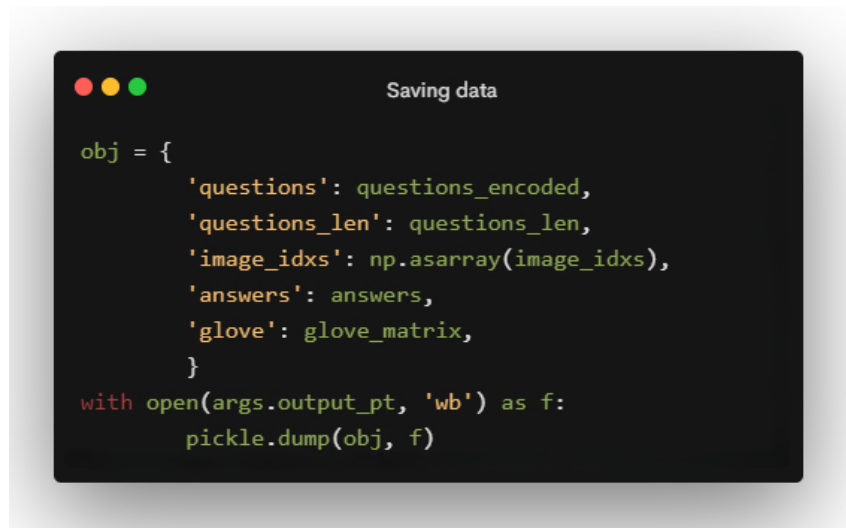
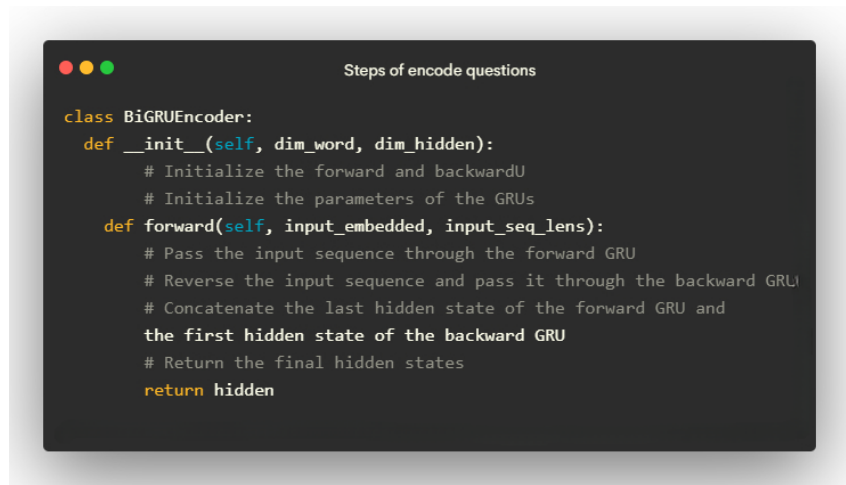


Figure 4.13: Saving processed data and GloVe embeds to Pickle file .

3.2.2 Question Encoded

To encode the questions, we use a representation used in the XNMs model [57] based on the BiGRU encoder. The BiGRUEncoder is a bidirectional GRU encoder that takes in the questions and their embeddings and outputs the hidden states of the encoder at each time. This allows the ODNet model to understand the meaning of the question and generate an accurate answer. The forward and backward of this encoder are initialized using “torch.nn.GRU layer”.

The nn.GRU is a type of recurrent neural network (RNN) layer in PyTorch that is used for processing sequential data. It takes in a sequence of input tensors and produces a sequence of output tensors. The figure below discusses the main steps of encoding questions:



```
class BiGRUEncoder:
    def __init__(self, dim_word, dim_hidden):
        # Initialize the forward and backward GRUs
        # Initialize the parameters of the GRUs
    def forward(self, input_embedded, input_seq_lens):
        # Pass the input sequence through the forward GRU
        # Reverse the input sequence and pass it through the backward GRU
        # Concatenate the last hidden state of the forward GRU and
        # the first hidden state of the backward GRU
        # Return the final hidden states
        return hidden
```

Figure 4.14: BiGRUEncoder for questions.

3.3 Fusion process

The fusion process in our model plays a crucial role in combining visual and textual information to generate accurate answers. For this, we combined the visual features and hidden states of the question encoder using element-wise multiplication. Then, calculates the mean of the fused features.

It is important to note that element-wise multiplication is different from matrix multiplication, which involves multiplying rows and columns of matrices to obtain a new matrix. In matrix multiplication, the dimensions of the matrices must be compatible, whereas, in element-wise multiplication, the dimensions must match exactly.

This fusion allows the information from both sources to interact and influence each other. The fused tensor contains combined information from the image and the question, which can provide a more comprehensive representation. The following figure explains this process:

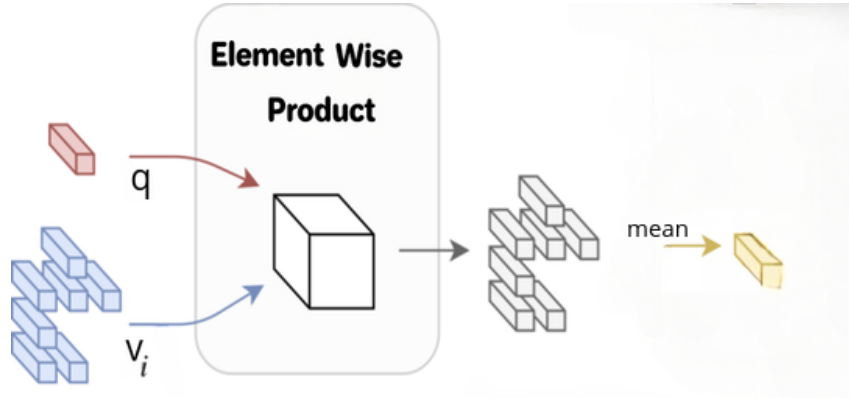


Figure 4.15: Fusion Process, The element-wise product represents the multiplication of the features of the image box V_i with the values of the question's hidden state q (each image includes 36 boxes) .

3.4 MLP Classifier

The fused feature representation is passed through a classifier perceptron before obtaining the predicted logits for the answer tokens. The classifier consists of a simple multi-layer perceptron that on the first passes fused features through batch normalization to normalize the features. Then, it applies a dropout layer to prevent overfitting by randomly dropping out some of the features. Finally, it passes the features through a linear layer allowing the model to learn a mapping from the features to the predicted logits for the answer tokens.

This classifier contributes to better generalization, improved training stability, and enhanced feature representation, which can lead to improved performance in predicting the answer logits for the VQA task.

3.5 Output Generation

The answer logits produced by the previous classifier are passed to the `log_softmax` function to output a probability distribution over the possible answers. This function is a PyTorch function from the `torch.nn.functional` module that applies the logarithm of the softmax function to an input tensor.

4 Training and validation of the proposed model

The training process for the "ODNet" model is a crucial step in developing an accurate and reliable model. The model is initialized with the specified model hyperparameters to control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The following table discusses the hyperparameters of the model:

The hyperparameter	Value	Description
<code>dim_word</code>	300	This hyperparameter represents the dimensionality of the word embeddings (to represent the input text).
<code>dim_hidden</code>	1024	It denotes the dimensionality of the hidden state in the sequence-to-sequence parser.
<code>dim_v</code>	1024	This hyperparameter represents the dimensionality of the node embeddings.
<code>dim_vision</code>	2048	This hyperparameter specifies the dimensionality of the visual features used in our model (the visual features extracted from the input image).
<code>cls_fc_dim</code>	1024	It denotes the dimensionality of the fully connected layer in the classifier.
<code>glimpses</code>	2	This hyperparameter determines the number of glimpses used in our model.
<code>dropout_prob</code>	0.5	It represents the probability of an element being zeroed out in the dropout layers of our model.

Table 4.2: The hyperparameters of the model.

For training, we selected the most frequent 3000 answers from the training set and predicted the target answer from these candidates. We used the Adam optimizer of PyTorch with an initial learning rate of 0.0008. Adam is an optimization algorithm used in deep learning and machine learning. It is a stochastic gradient descent method that is used to update network weights iterative based on training data.

The following instruction is used to implement the Adam algorithm in PyTorch.



Figure 4.16: Initializing Adam optimizer with learning rate and weight decay in PyTorch

The learning rate of the optimizer is gradually reduced using the `ExponentialLR scheduler` PyTorch over time, allowing the model to converge to a good solution. These components work together to optimize the model's performance during training. This scheduler implements using the following class:

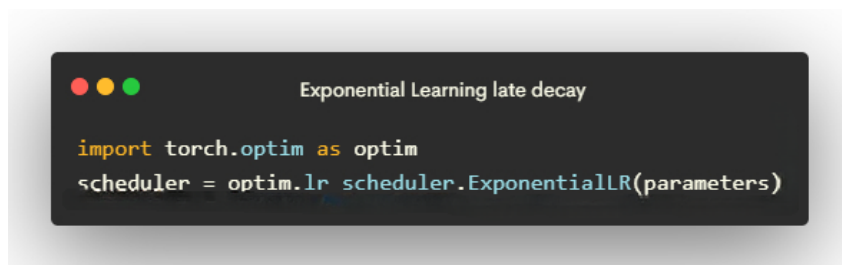


Figure 4.17: Implementation of Exponential Learning Rate Decay using the `lr_scheduler module`.

The training loop starts, iterating over multiple epochs and batches of data (a batch size of 16 and a total of 100 epochs). For each batch, the model processes the input data and computes the output logits. The logits represent the model's predictions for the answer to the given question. The loss is then calculated by comparing the logits with the ground truth answers.

After computing the loss, backpropagation is performed to calculate the gradients of the model's parameters with respect to the loss. These gradients are then used by the optimizer to update the parameters, adjusting them in the direction that minimizes the loss.

During training, checkpoints of the model and optimizer are saved at regular intervals.

These checkpoints allow for resuming training from the same point in case of interruptions or for later evaluation. The model’s state dictionary, optimizer state dictionary, and other relevant information are stored in the checkpoints.

The model’s performance is evaluated on a validation set after each epoch. This provides insights into how well the model is generalizing to unseen data. This information helps in monitoring the model’s progress and making adjustments if needed.

5 Experimental Results

To assess the effectiveness of our model, we conducted four experiments, each targeting a specific type of question in the training and validation process.

- In the first experiment, we focused on counting questions. These types of questions require the model to accurately determine the number of objects present in the image.
- The second experiment involved yes/no questions. These questions typically require the model to determine whether a specific object or attribute is present in the image or not.
- For the third experiment, we considered other types of questions that do not fall into the counting or yes/no categories. These questions may require the model to provide descriptive answers or explanations based on the visual information in the image.
- Lastly, we conducted an experiment that encompassed all question types.

By conducting several experiments and organizing them to include counting, yes/no, and other types of questions, we evaluate the general applicability and robustness of our model in handling various question types encountered in VQA tasks.

Here’s a consolidated table with the number of training and validation questions for each experiment:

Experiments	Train Questions	Validation Questions
Counting	51194	24538
Yes or No	171910	83226
Other	220653	106590
Overall	443757	214354

Table 4.3: Experiments questions.

Our experimental design allowed us to isolate the impact of different question types on object detection performance. The table below represents the results obtained.

Experiments	Accuracy	Number of epochs
Counting	43%	100 epochs
Yes or No	76%	100 epochs
Other	47%	95 epochs
Overall	41%	3 epochs

Table 4.4: Experiments results.

The experimental results table shows that our model's accuracy varies depending on the type of question being asked. The model achieved an accuracy of 76% for Yes/No questions after 100 epochs. for counting questions, it achieved an accuracy of 43% after 100 epochs and 47% accuracy for Other types of questions after 95 epochs. For overall, the model achieved an accuracy of 41% after only 3 epochs

To evaluate the impact of the proposed approach, we compare our VQA model with several ResNet baselines on the VQA v2.0 validation set of the paper "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering" [56].

The paper proposes a combined bottom-up and top-down visual attention mechanism for image captioning and visual question-answering (VQA) tasks. The ResNet baselines use a ResNet CNN pre-trained on ImageNet to encode each image in place of the bottom-up attention mechanism. The table below represents our obtained results and the results from the paper:

The models		Yes/No	Counting
Ours: Faster R-CNN		76%	43%
Paper_results (1×1)	ResNet	76%	36%
Paper_results (14×14)	ResNet	76%	36%
Paper_results (7×7)	ResNet	77%	37%

Table 4.5: Our obtained results and the results from the paper.

For the counting questions, our research achieved an accuracy of 43% using our proposed model. In comparison, the paper reports that ResNet (1×1) achieved a similar accuracy of 36% for counting questions. Similarly, ResNet (14×14) and ResNet also achieved an accuracy of 36% and 37% respectively for counting questions. From this comparison, it can be observed that our model’s performance is slightly better than ResNet (1×1) and comparable to ResNet (14×14) and ResNet for counting questions.

Regarding the yes/no questions, our research achieved an accuracy of 76%. In contrast, the paper indicates that ResNet (1×1) and ResNet (14×14) both achieved an identical accuracy of 76% for yes/no questions. On the other hand, ResNet achieved a slightly higher accuracy of 77% for the same type of questions. Based on this comparison, it can be inferred that our model achieves comparable results to the paper’s models, with accuracies of around 76%.

6 Overview of the development environments used

6.1 Hardware environment

We used a computer with the following features to do our work for the VQA object detection task:

- **Processor:** Intel Core i7-10750H.
- **RAM:** 32.00 GB
- **GPU:** NVIDIA GeForce RTX 2070 super (8GB).
- **Operating system:** Ubuntu 20.4.0

6.2 Software environment

6.2.1 Choice of programming language

When you choose a programming language that specializes in machine learning and deep learning, you should consider the skills listed in current job offers as well as the libraries available in different languages that can be used for deep learning processes. We decided to use Python programming language, which holds the highest demand and uses within the realms of machine learning and deep learning.

Python is a powerful, free, and interpreted open-source programming language. It makes it simple to learn and comprehend since it places an emphasis on code readability through the use of indentation and clear grammar.



Figure 4.18: Python [34] .

The reasons for this choice were:

- Python offers several powerful libraries and frameworks specifically designed for machine learning and deep learning projects, such as Tensorflow, PyTorch, NumPy, and SciPy.
- Python-based deep learning frameworks offer flexibility and scalability, allowing you to train models on various hardware configurations, including CPUs, GPUs, and distributed systems.
- It is a simple, productive, and usable language in almost all domains and systems.

6.2.2 Libraries

- **Numpy:** (Numerical Python) is an open-source library for the Python programming language used for numerical computing and scientific data analysis. It offers support for sizable, multidimensional arrays and matrices, as well as a range of mathematical operations for effectively using these arrays.

- **Torch:** The Torch library, is a powerful open-source machine learning library extensively used for building and training deep learning models. It provides a flexible and efficient framework for tensor computations and automatic differentiation.
- **Re:** is a tool for matching patterns and text manipulation in strings. It offers a clear and adaptable syntax for describing patterns that may be used to look up, match, and extract data from text.
- **Logging:** is a built-in module that provides a flexible and configurable framework for recording log messages from an application. It allows the incorporation of logging functionality into Python programs, making it easier to track and debug issues, monitor application behavior, and generate informative log output.

6.2.3 Software and Tools

For this work, we chose to utilize the following tools: Anaconda Navigator with Jupyter Notebook, and the Kaggle data science community due to their relevance and importance in the field of deep learning.

Anaconda Navigator, a popular package manager and environment manager, simplifies the setup of data science projects. Jupyter Notebook, on the other hand, provides an interactive computing environment for developing and presenting code. Anaconda Navigator is a graphical user interface (GUI) that comes bundled with the Anaconda distribution, which is a popular Python and R programming language distribution for scientific computing and data science. Anaconda Navigator provides an easy way to manage and launch applications, environments, and packages that are part of the Anaconda ecosystem.



Figure 4.19: Anaconda Navigator [35].

Anaconda Navigator offered a range of features that proved invaluable for our deep learning project. With its user-friendly interface, we could easily manage different Python environments, install packages, and handle dependencies. This streamlined the setup process and ensured the compatibility of various libraries and packages required for deep learning.

Jupyter Notebook is an open-source web program that enables us to create and share documents with real-time code, equations, visuals, and text. It's a popular tool among data scientists and researchers for interactive computing and data analysis.



Figure 4.20: Jupyter Notebook [36] .

Jupyter Notebook played a pivotal role in the development and experimentation phase of our deep learning project. Its ability to combine code, documentation, and visualizations within a single interface provided a powerful platform for interactive and reproducible research. Since Jupyter Notebook supports multiple programming languages, including Python (we use Python version 3.9.12 for our project), it was well-suited for implementing and testing deep learning algorithms.

On the other hand, we installed the deep learning framework PyTorch version 1.11.0 using Anaconda Navigator's package management system. PyTorch's extensive set of functionalities and modules empowered us to develop, train, and evaluate deep neural networks efficiently. With Anaconda Navigator ensuring compatibility and easy installation of PyTorch, we could seamlessly integrate this powerful framework into our deep learning workflow.



Figure 4.21: PyTorch [11] .

To harness the immense computational power of GPUs, we utilized CUDA version 11.3.1 within the Anaconda Navigator environment. CUDA, a parallel computing platform, and API, enables efficient GPU acceleration for deep learning tasks. With Anaconda Navigator's package management system, we installed CUDA alongside PyTorch seamlessly. This integration allowed us to leverage the parallel processing capabilities of GPUs, significantly accelerating the training and inference processes of our deep learning models. By utilizing CUDA within the Anaconda Navigator environment, we could easily configure GPU acceleration for our PyTorch models and maximize their performance.

The combination of Jupyter Notebook, PyTorch, and CUDA within the Anaconda Navigator environment created a powerful synergy for our deep learning project. Jupyter Notebook provided an interactive and user-friendly interface for code development, experimentation, and documentation. Anaconda Navigator ensured the smooth integration and compatibility of PyTorch, allowing us to utilize its comprehensive functionalities for building, training, and evaluating deep neural networks. By incorporating CUDA within the Anaconda Navigator environment, we could seamlessly harness the parallel processing capabilities of GPUs, accelerating the computations and improving the efficiency of our deep learning models.

On the other hand, Kaggle is a popular platform that provides access to a wide range of datasets, as well as tools and resources for data analysis and machine learning. One of the main benefits of using Kaggle is the availability of high-quality datasets that are preprocessed and ready for use. This saved us a significant amount of time and effort in data cleaning and preparation, allowing us to focus on building and training our model. Kaggle also offers a range of machine learning algorithms and frameworks, such as TensorFlow and PyTorch, which we used to build and train our neural network model. It offers access to 16GB of RAM or 13GB with a GPU connected to a single Kaggle

notebook. Moreover, Kaggle provides a user-friendly interface that allows users to easily upload their datasets, making it simple for data scientists and machine learning enthusiasts to access and analyze a wide range of data.



Figure 4.22: Kaggle [37].

Overall, by using Kaggle, we were able to train a high-performing neural network model that achieved our project goals. We believe that our use of Kaggle demonstrates our expertise in machine learning and our ability to leverage the latest tools and technologies to achieve our project goals.

7 Conclusion

In conclusion, this chapter presented the implementation details and analysis of the obtained results for our approach applied to the VQA V2 dataset. The preprocessing steps, including loading the data from JSON files, building the vocabulary, and indexing the questions, laid the foundation for further processing. The fusion process, MLP classifier, and output generation contributed to the successful integration of object detection features within the VQA pipeline. The experimental results demonstrated the effectiveness of our model in enhancing VQA performance. The choice of programming language, libraries, and software tools, combined with the hardware and software environments, facilitated the development and evaluation process. Overall, this chapter provides valuable insights into the realization of our work and its impact on the field of Visual Question Answering.

General Conclusion

In conclusion, this research introduces a novel approach to Visual Question Answering (VQA) that incorporates object detection features as image representations to enhance performance. Through specialized experiments targeting various question types, including Yes/No, counting, and others, valuable insights were gained. Promising results were achieved for Yes/No questions (76% accuracy), resulting in an accuracy of 43%. Other question types yielded an accuracy of 47%.

In our future work, we will focus on several key aspects. Firstly, it is essential to address the limitations imposed by the current software and hardware infrastructure. By improving these aspects, we will be able to conduct more experiments and obtain a comprehensive evaluation of our approach across all question types.

Secondly, we aim to enhance the model's ability to accurately estimate numerical values in various question types. This can be achieved through the fine-tuning of object detection features and the exploration of different architectures or fusion methods.

Furthermore, we plan to investigate strategies that will enable the model to handle more challenging question types, such as complex reasoning or temporal reasoning questions. By improving the model's performance in diverse VQA scenarios, we can ensure its effectiveness in a wide range of applications. Finally, conducting experiments on other VQA datasets or exploring transfer learning approaches to generalize the proposed approach's effectiveness would be valuable.

Bibliography

- [1] Yassmina Saadna. Cours 1, cours big data et deep learning. http://staff.univ-batna2.dz/sites/default/files/saadna_yassmina/files/cours_101.pdf, 2019. Visited on February 26, 2023.
- [2] Iqbal Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications, and research directions. *SN Computer Science*, 2, 08 2021.
- [3] Sagar Sharma. Activation functions in neural networks. <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>, 2017. visited on February 27, 2023.
- [4] Dishashree26 Gupta. Fundamentals of deep learning – activation functions and when to use them? <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>, 2020. visited on February 27, 2023.
- [5] Christian Versloot. An overview of the activation functions used in neural networks. <https://github.com/christianversloot/machine-learning-articles/blob/main/overview-of-activation-functions-for-neural-networks.md>, 2017. visited on February 27, 2023.
- [6] Ajitesh Kumar. Different types of cnn architectures explained: Examples. <https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/>, 2022. Visited on 02 March 2023.
- [7] Debasish Kalita. A brief overview of recurrent neural networks (rnn). <https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/>, 2022. Visited on March 02, 2023.

-
- [8] Jonte Dancker. A brief introduction to recurrent neural networks. <https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4>, 2022. Visited on March 02,2023.
- [9] Tensorflow. <https://www.tensorflow.org/>. Visited on March 15,2023.
- [10] Keras. <https://keras.io/>. Visited on March 15,2023.
- [11] Giovanni. Awesome pytorch list. <https://github.com/>, 2023. Visited on March 15,2023.
- [12] Tafadzwa Sigauke. *Design and Implementation of a mobile-based chatbot system for small-scale miners using artificial intelligence*. PhD thesis, 11 2019.
- [13] Editorial Team. Natural language processing (nlp) with python tutorial, 2020. Visited on March 12, 2023.
- [14] Anthony Técher. Pourquoi le tf*idf ne permet pas d’optimiser vos contenus. <https://www.seoquantum.com/billet/outil-tf-idf>, 2019. Visited on March 12,2023.
- [15] Ria Kulshrestha. Nlp 101: Word2vec skip-gram and cbow. <https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314>, 2019. Visited on March 12,2023.
- [16] Team rédac. Opencv : tout savoir sur le principal outil de computer vision. <https://datascientest.com/opencv>, 2022. Visited on March 12,2023.
- [17] Simplecv. <http://simplecv.org/>. Visited on March 15,2023.
- [18] Scikit image. <https://scikit-image.org/>. Visited on March 15,2023.
- [19] Himanshu Sharma and Anand Jalal. A survey of methods, datasets, and evaluation metrics for visual question answering. *Image and Vision Computing*, 116:104327, 10 2021.
- [20] Silvio Barra, Carmen Bisogni, Maria De Marsico, and Stefano Ricciardi. Visual question answering: which investigated applications? *Pattern Recognition Letters*, 151, 09 2021.
- [21] Javier Couto. Introduction to visual question answering: Datasets, approaches, and evaluation. <https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering>, 2018. Visited on February 06,2023.

- [22] Rufai Yusuf, Jim Owusu, Hailin Wang, Ke Qin, Zaharaddeen Lawal, and Yuezhou Dong. Vqa and visual reasoning: An overview of recent datasets, methods and challenges, 12 2022.
- [23] franky. Deep learning and visual question answering. <https://towardsdatascience.com/deep-learning-and-visual-question-answering-c8c8093941bc>, 2018. visited on March 23,2023.
- [24] Sanket Shah, Naganand Yadati, and Partha Talukdar. Kvqa: Knowledge-aware visual question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8876–8884, 07 2019.
- [25] Vincent Perrin. Une image vaut mieux qu’un long discours. <https://fr.linkedin.com/pulse/une-image-vaut-mieux-quun-long-discours-vincent-perrin>, 2019. Visited on March 30,2023.
- [26] Trad Houssein Eddine. La détection d’objet avec opencv et deep learning, 2020.
- [27] Kemal Erdem. Understanding region of interest (roi pooling). <https://erdem.pl/2020/02/understanding-region-of-interest-ro-i-pooling>, 2020. Visited on March 30,2023.
- [28] Kukil. Mean average precision (map) in object detection. <https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/>, 2022. Visited on March 30,2023.
- [29] Aishwarya Singh. Selecting the right bounding box using non-max suppression (with implementation). <https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/>, 2020. Visited on on March 31,2023.
- [30] Jun Deng, Xiaojing Xuan, Weifeng Wang, Zhao Li, Hanwen Yao, and Zhiqiang Wang. A review of research on object detection based on deep learning. *Journal of Physics: Conference Series*, 1684(1):012028, nov 2020.
- [31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

-
- [32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
- [33] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-yuan Liao. Yolov4: Optimal speed and accuracy of object detection, 04 2020.
- [34] Python. <https://www.python.org/community/logos/>. Visited on April 30,2023.
- [35] Anaconda. <https://geekflare.com/fr/install-anaconda-on-linux/>. Visited on April 30,2023.
- [36] <https://jupyter.org/>. Visited on April 30,2023.
- [37] Kaggle. <https://www.kaggle.com/brand-guidelines>. Visited on April 30,2023.
- [38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.
- [39] Benlahmar. Les réseaux de neurones convolutifs. <https://datasciencetoday.net/index.php/en-us/deep-learning/173-les-reseaux-de-neurones-convolutifs>, 2018. Visited on March 02,2023.
- [40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [42] Palash Goyal and Karan Jain. *Deep Learning for Natural Language Processing*. 01 2018.
- [43] Gaurav Rajesh Sahani. Understanding tf-idf in nlp. <https://medium.com/analytics-vidhya/understanding-tf-idf-in-nlp-4a28eebdee6a>, 2020. Visited on March 12, 2023.
- [44] Opencv. <https://opencv.org/>. visited on March 15,2023.
- [45] Akshay Gupta. Survey of visual question answering: Datasets and techniques. 05 2017.
- [46] Yash Srivastava, Vaishnav Murali, Shiv Ram Dubey, and Snehasis Mukherjee. Visual question answering using deep learning: A survey and performance analysis, 08 2019.

-
- [47] Dongxiang Zhang, Rui Cao, and Sai Wu. Information fusion in visual question answering: A survey. *Information Fusion*, 52, 03 2019.
- [48] Sruthy Manmadhan and Binsu Kooor. Visual question answering: a state-of-the-art review. *Artificial Intelligence Review*, 53, 12 2020.
- [49] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.
- [51] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.
- [52] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
- [53] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
- [54] MingYuan Wang and Watis Leelapatra. A review of object detection based on convolutional neural networks and deep learning. *International Scientific Journal of Engineering and Technology (ISJET)*, 6(1):1–7, Jun. 2022.
- [55] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
- [56] 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 VQA. *CoRR*, abs/1707.07998, 2017.
- [57] Jiaxin Shi, Hanwang Zhang, and Juanzi Li. Explainable and explicit visual reasoning over scene graphs. *CoRR*, abs/1812.01855, 2018.