الجمهورية الجزائرية الديمقراطية الشعبيــة
Democratic and Popular Algerian Republic
وزارة التـعليـم العالـي والبحـث العلمـــي
Ministry of Higher Education and Scientific Research
جامعة عين تموشنت بلحاج بوشعيب
University – Ain Temouchent - Belhadj Bouchaib
Faculty of Sciences and Technology
Department of Electronics and Telecommunications

End of Studies Project
For the obtaining of a Master's degree in: Telecommunications
Field: Science and Technology
Course: Telecommunications
Specialty: Networks and Telecommunications

Theme:

Arabic Image Captioning Using Neural Networks

Presented by:
1) EL HABIB DAHO Zakaria
2) HADDOUCHE Wissal

Before the jury members:

Dr SEKKAL Mansouria	MCA	UAT.B. B (Ain Temouchent)	President
Dr HACHEMI Belkacem	MCB	UAT.B. B (Ain Temouchent)	Examiner
Dr MEKAMI Hayat	MCB	UAT.B. B (Ain Temouchent)	Supervisor
Dr BOUTKHIL Malika	MAA	UAT.B. B (Ain Temouchent)	Co Supervisor

Academic year 2024/2025

Acknowledgments

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

*First, and foremost, we would like to express our deep gratitude to **Allah** for giving us the strength, patience, and perseverance to carry this work through to the end.*

*Our sincere thanks go to **Dr. Mekami Hayat**, our supervisor and co supervisor **Dr. BOUTKHIL**, for her support, guidance, and encouragement throughout this journey. Her dedication and thoughtful advice made a real difference, and we are truly thankful for her time and effort.*

*We also want to thank the **jury members** for taking the time to review our work and for their helpful feedback and suggestions.*

*A special thank you to all the professors in the **Telecommunications and Networks department**. Their teaching and commitment to our learning have been essential during our studies. We are also grateful to the administrative staff for always being helpful and supportive.*

*We are deeply thankful to our **families**, especially our **parents and siblings**, for standing by us through every stage — their support, love, and prayers meant everything.*

*Finally, to our **friends**, and to everyone who helped us along the way, directly or indirectly — thank you. Your encouragement kept us going.*

May Allah bless and reward each one of you.

Dedication

*I dedicate this work as a token of deep **respect**, **gratitude**, and **appreciation** to my **beloved parents**, for their unwavering dedication, patience, prayers, and the countless sacrifices they have made throughout my journey. May Almighty God watch over and protect them always.*

*To my **siblings**, **Rayen** and **Douaa**, for their love, support, and constant encouragement.*

*To my **dear friends**, **Charaf** and **Marouane**, for their presence, motivation, and support during both the challenging and rewarding moments of this journey.*

*To my **project partner, Haddouche**, for her patience, dedication, and the quality of work she brought to this collaboration.*

To all those who, directly or indirectly, contributed to the realization and success of this work.

*In addition, with deep respect and sincere thanks to **Ms. Mekami**, my supervisor, for her valuable guidance and meaningful contribution to this thesis.*

EL HABIB DAHO Zakaria

Dedication

I dedicate this modest work to my beloved parents for their endless sacrifices, support, and unwavering love, your presence have shaped the person I become.

To my dear sisters, Sabah, Feriel, and Ines — my first friends, and my long lasting companions your love and encouragement have always been a spring of joy and a mark on my full heart.

To my sister abroad, your courage and heartfelt messages from afar have been a motivation and comfort to my soul.

To my grandmothers, my aunt Rafika and my uncle Mohamed, your wisdom, prayers, and warmth have left a blessing in every step in my life.

To my friends, my girls — may we achieve more, dream bigger, and lift each other higher every step of the way.

Lastly, with a wounded heart I dedicate this thesis in loving memory of my grandfathers, HADDOUCHE Mohamed, who passed away on January 6th, 2023, and KACIMI Oukacha, June 11th, 2025, who is in the mercy of Allah.

May Allah ﷻ swaddle them in his mercy, forgive their sins, and reward them with the highest ranks of Jannah.

اللهم اغفر لهما وارحمهما، واجعل قبرهما روضة من رياض الجنة.

HADDOUCHE Wissal

Abstract

Image captioning, which aims to generate natural language descriptions for images, represents a major challenge at the intersection of computer vision and natural language processing. While significant advances have been made for English, Arabic image captioning remains underexplored, largely due to the lack of dedicated datasets and the complexity of the Arabic language.

In this thesis, we present an Arabic image captioning model based on an encoder-decoder architecture, combining the VGG16 convolutional neural network for image feature extraction with a Long Short-Term Memory (LSTM) network for sentence generation. To address the lack of Arabic resources, we constructed a new dataset by translating and refining existing English image caption datasets of flickr8K.

The proposed model was trained and evaluated on this dataset. By indicating its ability to generate relevant and coherent Arabic captions. These results highlight the potential of deep learning approaches to advance Arabic image captioning and contribute to bridging the resource gap for Arabic in the field of image understanding.

Keywords: Image Captioning, CV, NLP, CNN, RNN, LSTM, VGG16, AI, Deep learning, Machine Learning, Evaluation, BLUE, AraBERT

Résumé

La légende d'image, qui vise à générer des descriptions en langage naturel pour les images, représente un défi majeur à l'intersection de la vision par ordinateur et du traitement du langage naturel. Bien que des avancées significatives aient été réalisées pour l'anglais, la légende d'images en arabe reste peu explorée, principalement en raison du manque de jeux de données dédiés et de la complexité de la langue arabe.

Dans cette thèse, nous présentons un modèle de légende d'image en arabe basé sur une architecture d'encodeur-décodeur, combinant le réseau de neurones convolutionnels VGG16 pour l'extraction des caractéristiques d'image avec un réseau de mémoire à long et court terme (LSTM) pour la génération de phrases. Pour remédier au manque de ressources arabes, nous avons construit un nouveau jeu de données en traduisant et en affinant des jeux de données de légendes d'images en anglais existants de flickr8K.

Le modèle proposé a été entraîné et évalué sur cet ensemble de données. En indiquant sa capacité à générer des légendes en arabe pertinentes et cohérentes. Ces résultats soulignent le potentiel des approches d'apprentissage profond pour faire avancer la légende d'images en arabe et contribuer à combler le fossé des ressources pour l'arabe dans le domaine de la compréhension des images.

Mots-clés: Légende d'image, CV, NLP, CNN, RNN, LSTM, VGG16, IA, Apprentissage profond, Apprentissage automatique, Évaluation, BLUE,
AraBERT

الملخص

توليد وصف لصور، الذي يهدف إلى إنشاء أوصاف بلغة طبيعية للصور، يمثل تحدياً كبيراً عند تقاطع رؤية الكمبيوتر ومعالجة اللغة الطبيعية. على الرغم من القيام بتقدمات كبيرة في اللغة الإنجليزية، إلا أن توليد الوصف للصور باللغة العربية يبقى غير مستكشف إلى حد كبير، وذلك إلى حد كبير بسبب نقص مجموعات البيانات المخصصة وتعقيد اللغة العربية.

في هذه المذكرة، نقدم نموذجاً لتوليد وصف الصور باللغة العربية يعتمد على بنية ترميز-فك، يجمع بين الشبكة العصبية التلافيفية VGG16 لاستخراج ميزات الصورة مع شبكة الذاكرة الطويلة والقصيرة (LSTM) لتوليد الجمل. لمعالجة نقص الموارد العربية، قمنا ببناء مجموعة بيانات جديدة من خلال ترجمة وتنقيح مجموعات بيانات الوصف للصور الموجودة باللغة الإنجليزية من flickr8K.

تم تدريب النموذج المقترح وتقييمه على هذه المجموعة من البيانات. من خلال الإشارة إلى قدرته على توليد تسميات عربية ذات صلة ومتناسقة. تسلط هذه النتائج الضوء على إمكانيات أساليب التعلم العميق لتعزيز وصف الصور باللغة العربية والمساهمة في سد الفجوة في الموارد للغة العربية في مجال فهم الصور.

الكلمات الرئيسية: صورة معبرة، CV, NLP, CNN, RNN, LSTM, VGG16, IA, الذكاء الاصطناعي، التعلم العميق، التعلم الآلي، تقييم، BLUE, AraBERT

Abbreviation

2D = Two-Dimensional

3D = Three-Dimensional

AI = Artificial Intelligence

AI (Illustrator) = Adobe Illustrator

ANN = Artificial Neural Network

BLEU = Bilingual Evaluation Understudy

BMP= Bitmap Image File

CDR = CorelDRAW

CMYK = Cyan, Magenta, Yellow, Black

CNN = Convolutional Neural Network

CV = Computer Vision

DL = Deep Learning

EPS = Encapsulated PostScript

FDD = Feedforward Deep Network

GAN = Generative Adversarial Network

GIF = Graphics Interchange Format

GPS = Global Positioning System

GRU = Gated Recurrent Unit

GPU = Graphics Processing Unit

IA = Artificial Intelligence

ILSVRC = ImageNet Large Scale Visual Recognition Challenge

JPEG = Joint Photographic Experts Group

LSTM = Long Short-Term Memory

ML = Machine Learning

MSA = Modern Standard Arabic

MSCOCO = Microsoft Common Objects in Context

NLP = Natural Language Processing

PDF = Portable Document Format

PNG = Portable Network Graphics

RGB = Red, Green, Blue

RNN = Recurrent Neural Network

VGG16 = Visual Geometry Group 16

SVM = Support Vector Machines

Contents

List of Figures

List of Tables

List of Equations

General Introduction

General Introduction

In recent years, the integration of computer vision and natural language processing has led to significant advances in the field of image captioning, where machines automatically generate descriptive sentences for images. This technology plays a crucial role in various applications, including aiding visually impaired individuals, enhancing image retrieval systems, and supporting automated content generation.

Although image captioning has been extensively explored for languages like English, research in this area for the Arabic language remains limited. Arabic, despite being one of the most widely spoken languages globally, presents unique challenges for natural language generation. Its rich morphology, complex grammatical structures, and the presence of numerous dialects make the task even more complex. One of the main obstacles facing Arabic image captioning is the lack of large, high-quality datasets comparable to those available for English, such as MS COCO or Flickr30k. To overcome this limitation, in this work, we created our own Arabic image captioning dataset by applying automatic translation to an existing English dataset, followed by human verification and correction to ensure linguistic quality and accuracy.

This thesis focuses on building and evaluating deep learning models capable of generating accurate and meaningful Arabic descriptions for images, thus contributing to bridging the gap in resources and research for Arabic image captioning.

We organised this thesis in four chapters:

- The first chapter introduces the fundamental concepts of image processing, along with its main techniques and applications.
- The second chapter focuses on artificial intelligence, specifically deep learning and neural networks, with particular emphasis on CNN and RNN architectures used in computer vision and natural language processing tasks.
- The third chapter provides a detailed overview of the image captioning task, presenting existing approaches, commonly used models, and the specific challenges related to the Arabic language.

The fourth chapter presents the experimental results obtained, discusses the performance of the proposed model, and analyses its limitations and possible improvements.

Chapter I:

Introduction to Image Processing

I.1. Introduction

In the digital era, images have become essential tools for communication and analysis across various fields, including artificial intelligence, medical imaging, and multimedia applications. A digital image is a numerical representation of visual information, analysed and stored in a format that computers can understand. The fundamental concepts of a digital image are crucial for various fields, including deep learning and computer vision, where machines are left to process and analyse visual data.

In the field of image captioning, where artificial intelligence generates descriptions of images, the structure, processing and image factors such as types, formats, and pixel structure of digital images form the backbone of accuracy and quality of machine generated captions.

This chapter delves into the importance of digital images, covering their structure, representation, and essential processing techniques. Understanding these concepts enables a thorough insight into how humans and machines perceive and interpret visual information.

I.2 Digital images

I.2.1 Digital Image

An image is a two-dimensional (2D) representation of a visual element that is stored and processed in a digital format. It consists of a structured array of pixels, abbreviation for Picture Element, where the total number of pixels defines the image's resolution. The pixels represent colour, greyscale, or intensity level, with each element appointed to a specific (x, y) coordination. The mathematical representation of a digital image is a matrix, where each pixel numerically corresponds to a visual property. [1]

I.2.2 Types of digital image

In digital image processing there are many types of images available. Each one serves a very different purpose. We distinguish four types of digital images:

Figure I. 1: An image represented in Binary, Grayscale and colour image *[2]*

I. 2.2.1 Binary images

A binary image, also known as a Bi-level Image, is a digital image that uses two distinct pixel values and is displayed in black and white. In mathematical modelling and neural networks, each pixel has two possible numerical values: zero (black) and one (white). In some cases, like image processing, the pixels corresponding to the white colour are often represented as 255 instead of one. [3] [4]

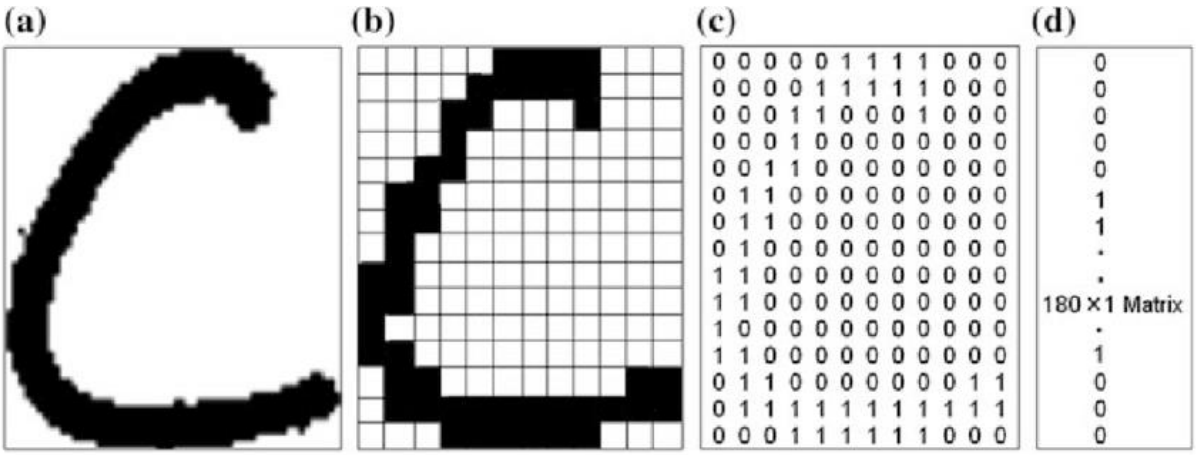


Figure I.2: (a) Binary image of character 'c', (b) resized binary image of character 'c', (c) binary matrix representation, and (d) reshaped binary matrix or feature vector of character 'c' *[5]*

I. 2.2.2 Grayscale images

A grayscale image is a type of image that contains no colours, only shades of grey. It ranges between 256 different levels of grey, where each pixel represents a single intensity value, ranging from zero (black) to 255 (white) in a 1-byte (8-bit) representation. Grayscale images use a single channel and are commonly used in object recognition and image processing. [3] [4]

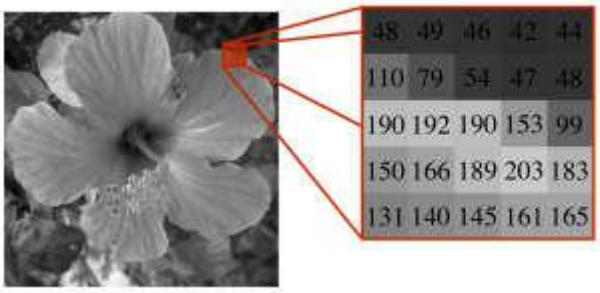


Figure I. 3: Grayscale Image *[6]*

I. 2.2.3 Colour images (RGB)

A coloured image is a digital or printed image made up of pixels that contain three intensity levels corresponding to the primary colours for mixing light: red, green, and blue (sometimes referred to as RGB), in digital displays, while subtractive primary colours used for mixing paints: cyan, magenta, yellow, and black (also referred to as CMYK), in printed images. Each colour pixel is typically stored using 3-bytes (24-bits)

of memory. Unlike grayscale images that use one channel, colour images are a combination of three separate images captured at selected ranges of the visible spectrum representing blue, red and green tones, making them more visually realistic. [3] [4]

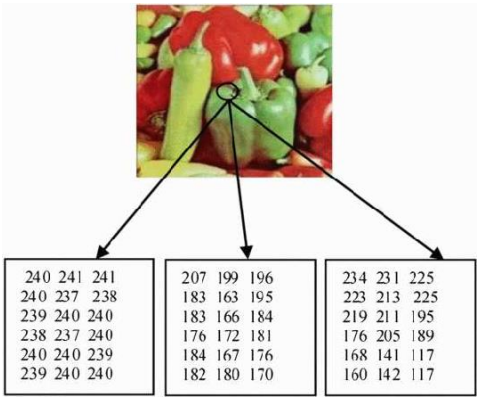


Figure I.4: RGB matrix representation of an image *[7]*

I. 2.2.4. Multispectral Images

Multispectral images is a group of image layers of the same spatial region, collected in one image using satellite sensors or specialised cameras, each image layer obtained at a certain wavelength band across the electromagnetic spectrum (ex. X-Ray, Infrared, and ultraviolet…), these data are typically outside the human perceptual reach, outside the visible spectrum (400 nm – 700 nm). [8] [9]

The high-resolution visible sensor uses the 3-band wavelengths:

- Blue: 450–520 nm
- Green: 520–600 nm
- Red: 630–690 nm
- Near Infrared (NIR): 750–900 nm

Multispectral data is typically stored as stacked grayscale images, where each layer represents a specific spectral band.

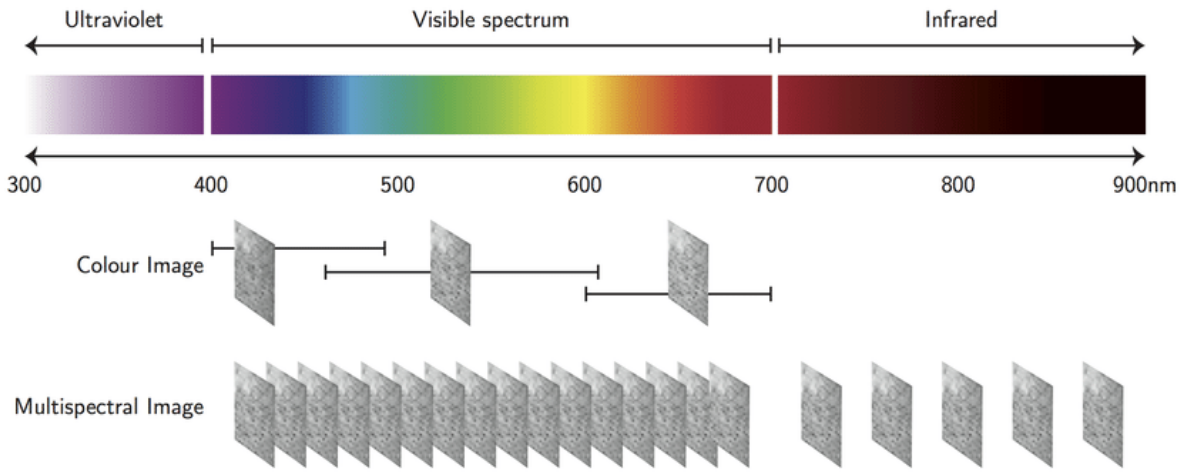


Figure I.5: The Process of Capturing a Multispectral Image *[10]*

I. 2.3 Pixel (Picture Element)

A pixel, short for Picture Element, is the smallest unit in a digital image or screen display. A grid of pixels arranged in a pattern forms an image, with each pixel holding information about colour (in colour images), intensity (in grayscale images), and brightness. The combination of these units determine the appearance of the image. [3]

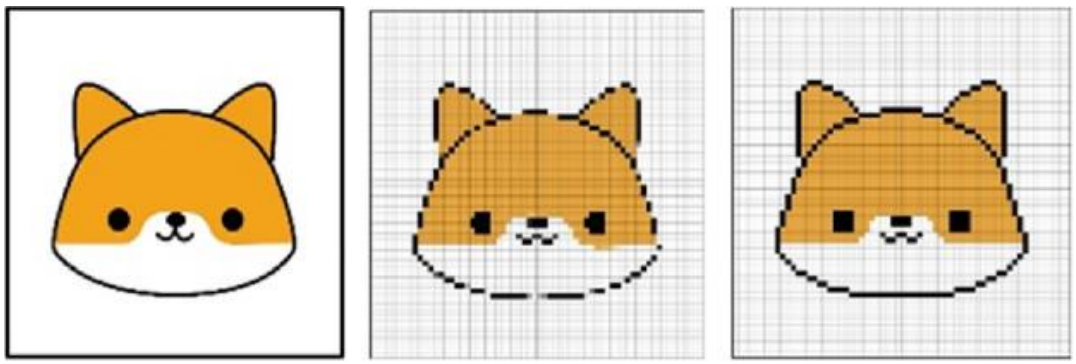


Figure I.6: Artist pixel image creation process: (a) original image, (b) image after processing by "pixilation" function, and (c) final pixel image *[11]*

I. 3. Pixel Distribution: Histograms

An image histogram describes the relative frequency level of each of the pixel values that forms an image. A histogram is a graph, a plot, used to visualise pixel distribution's value of an image in X-axis (for 8-bit grayscales images: 0 to 255) and their corresponding number of pixels (frequency) on Y-axis for each intensity value. [12] [13]

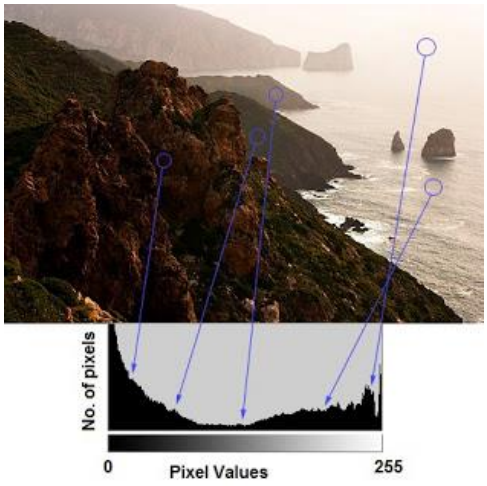


Figure I.7: Histogram Sample [10]

I. 3.1 Types of histograms

A histogram helps visualising pixel distribution of an image throughout a continual interval to make the data comprehensible. There are two variations of histograms based on images:

I. 3.1.1 Grayscale Image Histograms

A grayscale histogram is the portrayal of pixel intensities disposition in a greyscale image to aid in visualising the overall brightness, contrast, and distribution intensity. It uses only one channel (from black to white: 0-255) [14]

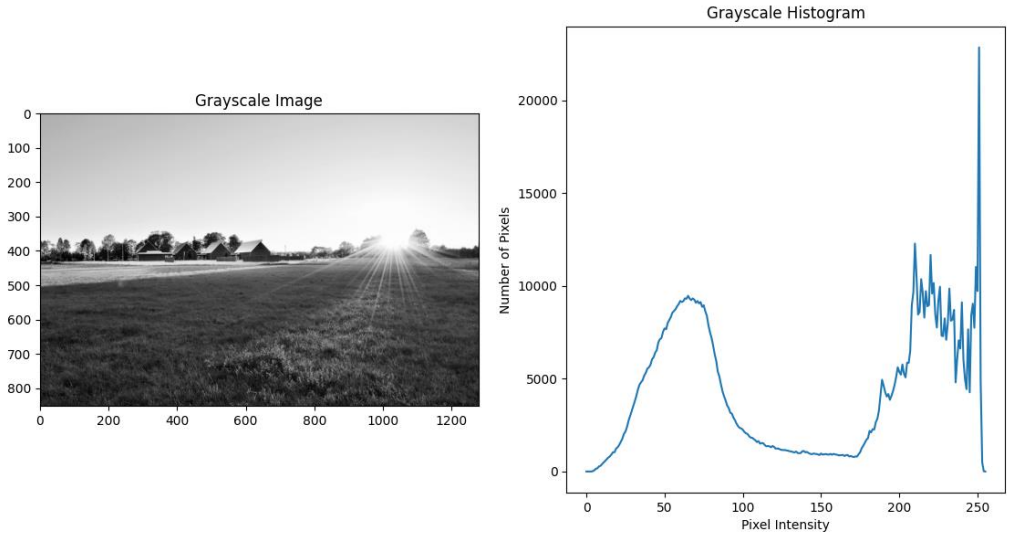


Figure I.8: Grayscale Image Histogram *[14]*

I. 3.1.2 Colour Image Histograms

A colour image consists of various colours (RGB combination); one pixel is stored using 3-bytes (24-bits) of memory. In this case, histograms are used to visualise the amount of RGB used to display an image, where each colour has a separate histograms for each channel. Colour image histograms are useful in analysing colour balance and dominance. [15]

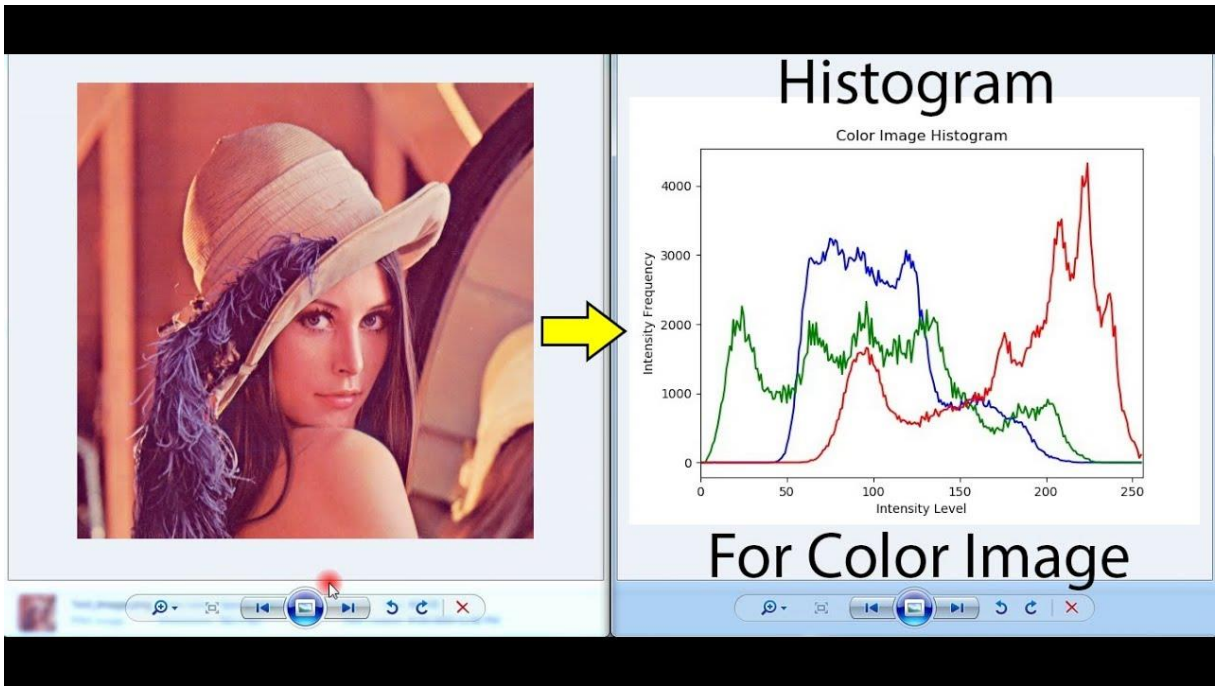


Figure I.9: Colour Image Histogram *[16]*

I. 3.2. Histogram Equalization

Histogram equalisation is a technique in image processing used to enhance contrasts level by spreading out the frequent intensity level across the available range. This task transforms pixel distribution values so that the output's histogram is steady. Enhancing low-contrast images, improving visual details are one of the case use of histogram equalisation, relying on pixel redistribution intensities. [12] [13]

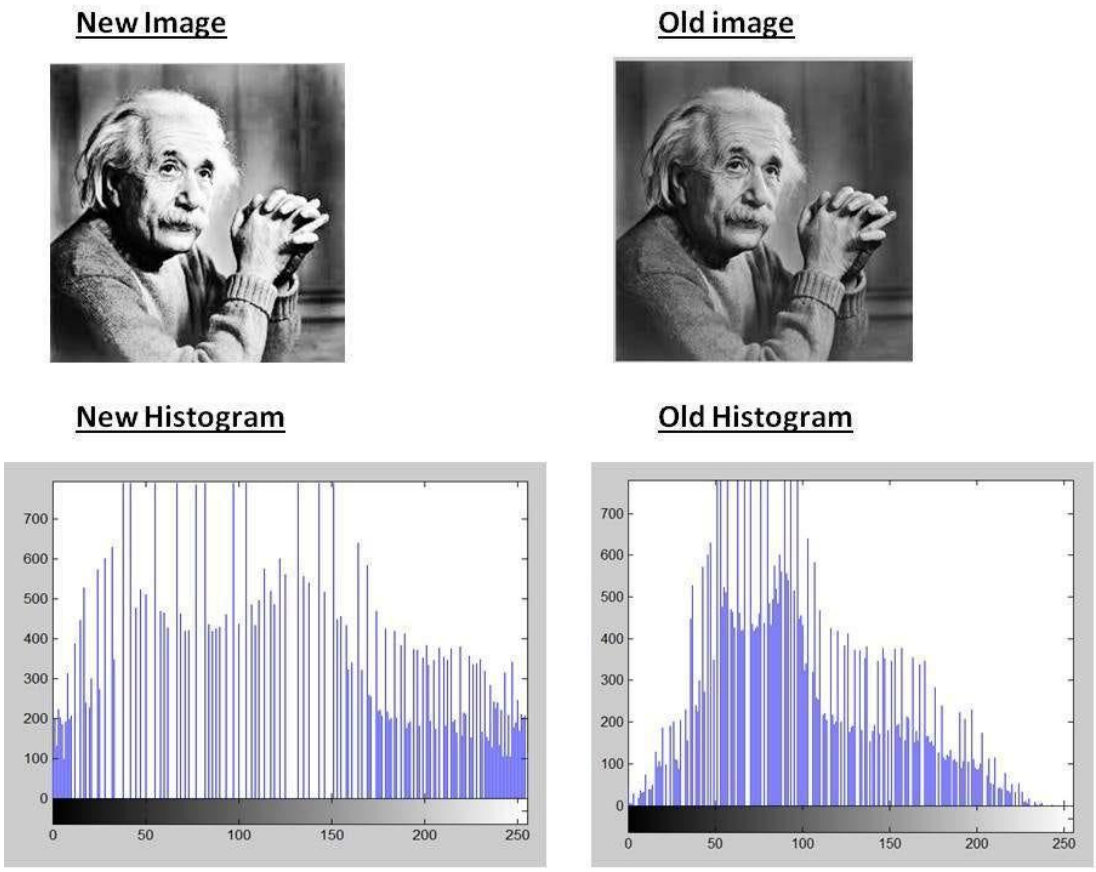


Figure I.10: an example of an image enhanced using histogram equalisation *[17]*

I. 4. Image Formats

I. 4.1 Common Image Formats

Every graphic seen online is an image file. These image files come in a variety of formats, and each file is optimised for a specific use. Most image files fit into two categories in general; vector files and bitmap files, and each category has its own specific uses.

Figure I.11: Bitmap and vector graphic comparison *[18]*

I. 4.1.1 Bitmap images

A bitmap image, also known as a raster image, is an image defined by a grid of pixels, where each pixel has a specific colour or intensity level. The total number of pixels (image resolution) in a photograph determines the level of detail a picture can represent. A pixel density (PPI/DPI) defines how sharp an image appears on a screen or print. [19]

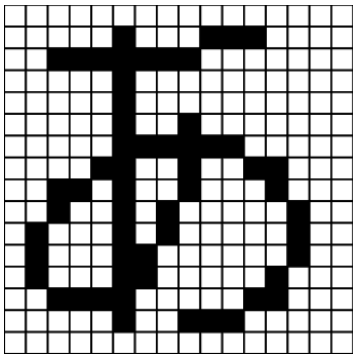


Figure I.12: an example of the bitmap image of a character *[20]*

Bitmap images comes in different formats used widely online, the table below lists the most commonly used formats:

Acronym	Name	Properties
BMP	Bitmap Image File	Contains bitmap graphics data

		Lossless compression algorithm Requires storage space Stores coloured or monochrome 2D image
GIF	Graphics Interchange Formats	A solution for electronic image storage Uses web-based graphic file format Lossless compression algorithm Provides a rough preview of an image before it fully loads
JPEG	Joint Photographic Experts Group	Created for the storage and transmission of photographic images Uses bit-mapped information stored in pixel Compress images for faster digital transfer Lossy compression algorithm
PNG	Portable Network Graphics	Supports transparent and semi-transparent backgrounds Lossless compression algorithm Supports 16 million colours Reconstructed image after decompression is the same as the original file

Table I.1: Bitmap file formats *[19] [21]*

These formats are commonly used in digital photography, computer graphics, and screen displays. [21] [19]

I. 4.1.2 Vector images

Vector images are composed of mathematical formulas to capture the characteristics of an image (size, border, shape…). They consist of lines, and curves rather than pixels, which make them resizable infinitely without losing resolution. Logos, icons, and graphic designs are the best examples. AI, CDR, EPS, and PDF are some of the commonly used formats of vector images, while Illustrator and CorelDraw are the commonly used vector programs. [19]

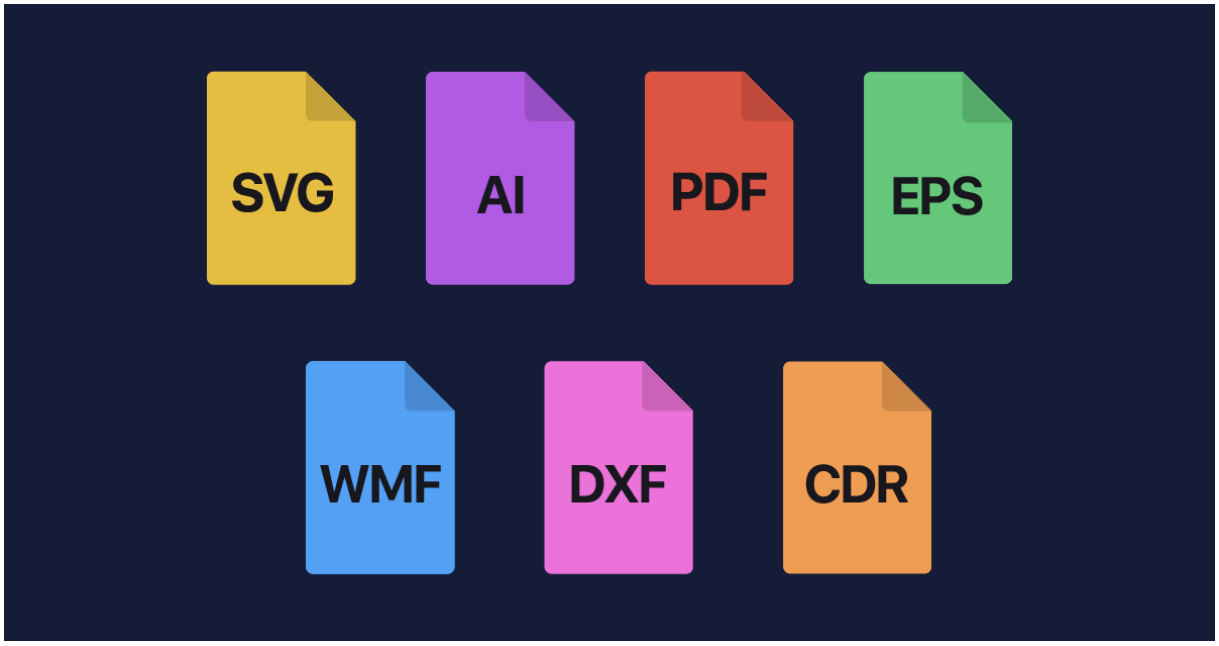


Figure I.13: Common Vector File Types *[22]*

I. 5. Image Processing

Image processing is a technique used to transfer a simple image into a digital image to extract useful data after performing certain operations. In general, image processing refers to processing a two-dimensional picture and data by a digital computer. This task broadly centres on two vital functions: enhancement of visual quality for human interpretation and processing for machine perception. [12] [23]

I. 5.1 Types of Image Processing

Image processing is a core area of computer vision and involves transforming images to extract information, enhance quality, or enable human analysis or machines'. [12] There are five main types of Image processing, each serving a distinct purpose:

Type of Image Processing	Purpose
Visualisation	Identify/highlight objects not visible clearly in unrefined image
Recognition	Detect/classify objects in an image
Sharpening and restoration	Enhance image quality by improving details or correcting distorted parts
Pattern recognition	Analyse patterns and interpret structures
Retrieval	Browse an image from a sizeable database based on similarity

Table I.2: Main Types of Image Processing *[24]*

I. 5.2 Evolution of Image Processing

The evolution of image processing stretches over several decades, from fundamental analogue methods processing to deep learning techniques. The figure below details the timeline:

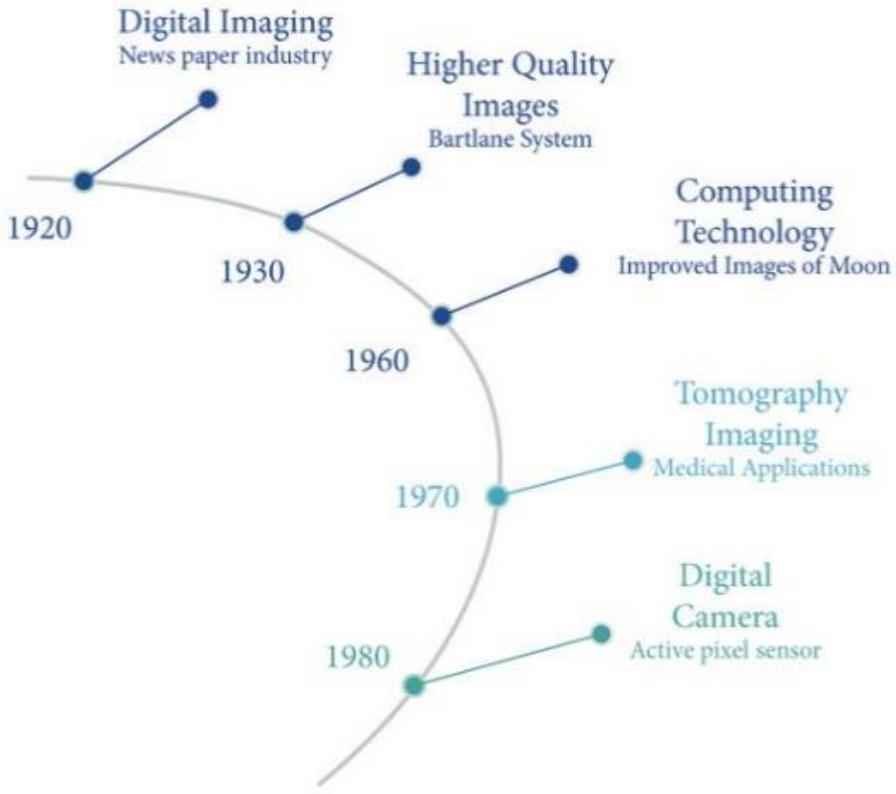


Figure I.14: Evolution of Image Processing *[25]*

I. 5.3 Image Processing Techniques

Image processing techniques can be categorised in general into several methods based on their functions and complexities. Below in an organised figure displaying the key techniques used for digital image processing.

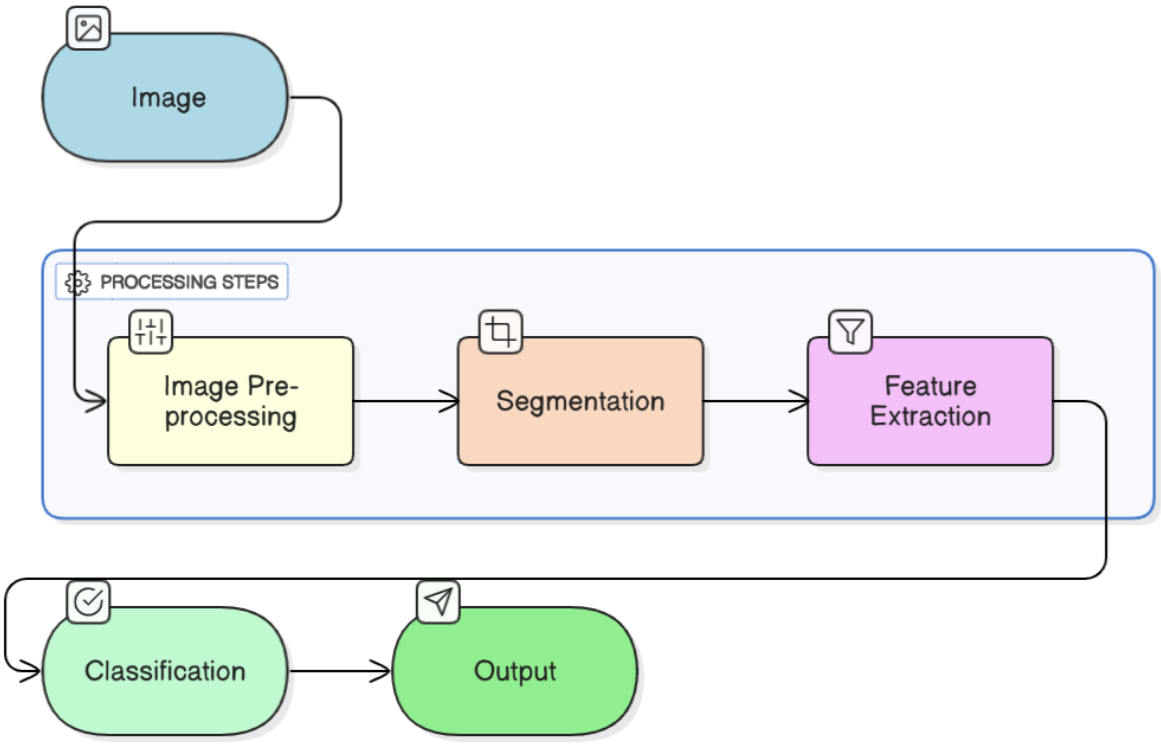


Figure I.15: Techniques of Image Processing

- **Image Enhancement**

Image enhancement changes pixel brightness values for an improved visual quality. The main issues addressed are low contrast, poor brightness, and noise from image limitations. It does not add new data, but it does highlight key features for display and analysis. [26]

Figure I.16: Image enhancement, before (Left) and after (Right)

The table below displays the different techniques for image enhancement.

Techniques	Description

Contrast stretching	Enhances images with low contrast by stretching a narrow range of levels to the full dynamic range, improving interpretability.
Noise filtering	Removes unwanted noise from images using filters such as low-pass, high-pass, mean, and median. Often interactive.
Histogram modification	Adjusts image characteristics by modifying the histogram, like Histogram Equalization redistributed pixel values to enhance contrast.

Table I.3: Different Enhancement Techniques *[26]*

- **Image Segmentation**

Segmentation image is a process that divides an image into significant parts. Its goal is to stop the divided parts are isolated. In the case of detecting vehicles, first, the environment is segmented, and then vehicles are identified. Thresholding techniques are commonly used for image segmentation. [26]

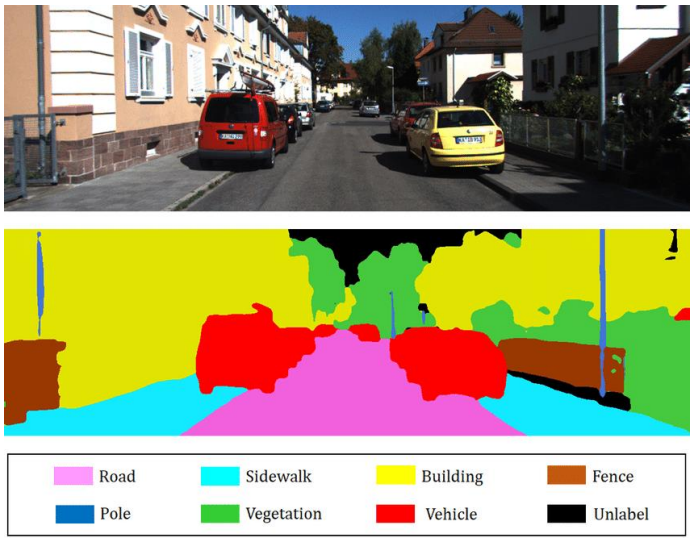


Figure I.17: Example of 2D semantic segmentation: (Top) input image (Bottom) prediction.

Thresholding is a technique that converts a digital image into a binary image, generally, the object pixels are black and the background is white. Mapping object pixel to black is the best threshold. [26]

$$S(x.y) = \begin{cases} 0, if\ g(x.\ y) < T(x, & y) \\ 1, if\ g(x.\ y) \geq T(x, & y) \end{cases}$$

Equation I.1: Thresholding: Mapping Grayscale Values to a Binary Set {0,1} *[26]*

S (x, y) is the segmented image value, g (x, y) is the pixel (x, y)'s grey level, and T (x, y) is the threshold value at (x, y). Segmentation of images sometimes involves the separation between different regions. [26]

- **Feature Extraction**

Feature extraction is a core Technique in image processing developed to retrieve useful key visual components such as shapes, textures, and colours from an image. These components are referred to as features, and they serve as a representation of the important parts of an image. [27]

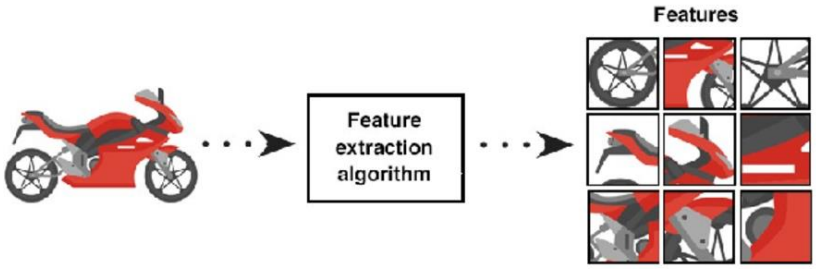


Figure I.18: Example of Feature Extraction Algorithm *[28]*

The image above shows a simple example of how feature extraction functions. First, the machine is given a raw image input of a simple object, a motorbike in this case. Then, the machine analyse the object, identify and isolate relevant attributes or patterns into a set of meaningful descriptions that can be used in differ tasks, such as, classifications, object detection, and recognition.

- **Image Classification**

Image classification is a core task in image processing and computer vision that aims to comprehend and analyse a content of an image and categorise it under a certain label based on visual content. [29] Common architecture of image classification is Convolutional Neural Networks (CNNs), standard for image classification. [30]

Figure I.19: An Example on Emotion Classification *[31]*

Key components of image classification are:

- Assigning labels to automatically categorise images into predefined classes (e.g., figure above “neutral,” “angry,” “surprise” …).
- Feature extraction to identify colours and shapes. It uses layers of mathematical operations to extract meaningful features (e.g., edges, object shapes).
- Training models with labelled data such as ImageNet, to achieve an accurate image classification. [29]

I. 6. Conclusion

Digital images play a crucial role in modern computation, allowing applications across various fields notably artificial intelligence, object recognition, and image description. This chapter has examined various aspects of digital images, which includes their characteristics, formats, compression techniques, and types.

Chapter II:

Foundations of Artificial Intelligence and Neural Networks

II. 1. Introduction

"The brain has about 100 billion neurons. If we can understand how just a few thousand of them work together, we can make a lot of progress in artificial intelligence."

— Geoffrey Hinton

In the rapidly evolving technological landscape, deep learning has emerged as a pivotal innovation, revolutionising the methodologies used to solve complex problems and perform intricate tasks. This paradigm shift is based on the principles of machine learning, whereby computers can be trained to simulate human intelligence through experiential learning.
Training a computer to recognise felines exemplifies the underlying principle of deep learning. Rather than relying on pre-programmed rules to identify whiskers, ears, and tails, a vast repository of cat images is presented for processing.

The computer then autonomously discovers recurring patterns and characteristics, developing the capacity to identify felines. This process highlights the essence of deep learning as a methodology that enables machines to derive inferences and make decisions without explicit programming instructions.
Neural networks, computational models based on the structural and functional similarities between the human brain and artificial systems, inspire deep learning.

These networks comprise interconnected nodes, or 'neuronal' units, that execute information processing functions and make decisions like discrete brain regions responsible for distinct tasks. Deep learning is characterised by the intricate layering of its networks, with multiple layers between the input and output, enabling the extraction of complex features and enhancing predictive capabilities.

II. 2. Artificial Intelligence (AI)

Artificial intelligence serves as the foundation and parent domain of both machine learning and deep learning. It's a very large field of research in which machines exhibit cognitive abilities such as learning behaviour proactive interaction with the environment, inference and deduction, computer vision, these AI capabilities include speech recognition, problem-solving, knowledge representation, perception, and many others. [32] More colloquially, AI refers to any activity in which machines mimic intelligent behaviour typically exhibited by humans. Artificial intelligence is inspired by elements of computer science, mathematics, and statistics.

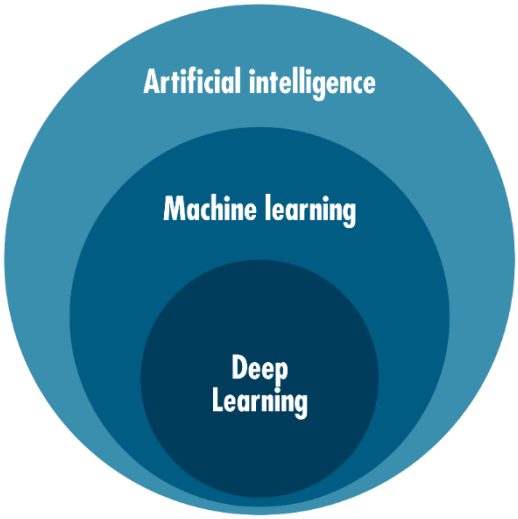


Figure II.1: From General to Specific: A visual Breakdown of AI technologies

II. 3. The Evolution of Machine Learning to Deep Learning

II. 3.1. Machine Learning

Machine learning, or ML for short, is a subfield of artificial intelligence (AI) that focuses on developing computers' algorithms that improve on their own with practice and data. In other words, machine learning enables computers to learn from data and make decisions or predictions without requiring special programming.

At its core, machine learning is the process of creating and implementing algorithms that facilitate these assessments and forecasts. These algorithms become more accurate and effective over time as they process more data.

In conventional programming, a computer executes a task by following a set of predetermined instructions. Meanwhile in machine learning, a task and a collection of examples (data) are provided to the computer, which then determines how to complete the task using the examples.

II. 3.2. An Overview of Machine Learning Processes

A machine-learning model is trained on a dataset to make predictions on new data. If the model's accuracy is acceptable, it is used for predictions. If not, the algorithm is retrained using an extended dataset to improve performance.

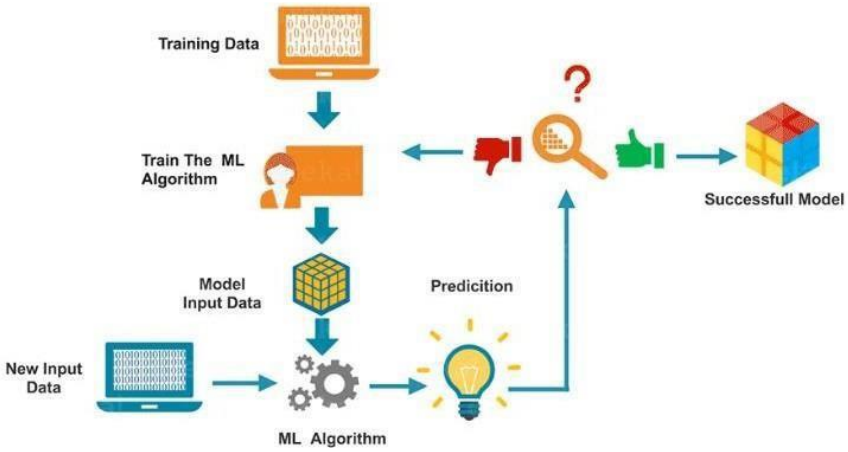


Figure II.2: Machine Learning Workflow: From Training Data to Model Deployment

II. 3.3. Types of Machine Learning

II. 3.3.1. Supervised Machine Learning Algorithms

This type of algorithm uses labelled data, or simply put data that contains both the input parameters and the desired output, as the dataset on which the machine is trained. For example, categorizing someone as male or female. We will use male and female as labels, with existing class assignments in our training dataset based on specific criteria, through which the machine will learn these features and patterns and classify some new input data based on learning from this training data.

Supervised learning algorithms can be broadly divided into two types of algorithms, classification and regression.

- **Regression:** This kind of problem requires us to predict a continuous-response value, such as the stock's value, which can range from -infinity to +infinity at the above number.

- **Classification:** This kind of problem requires us to predict the categorical response value, where the data can be divided into distinct "classes" (for example, we must predict one of a set of values). For instance, is this email spam or not?

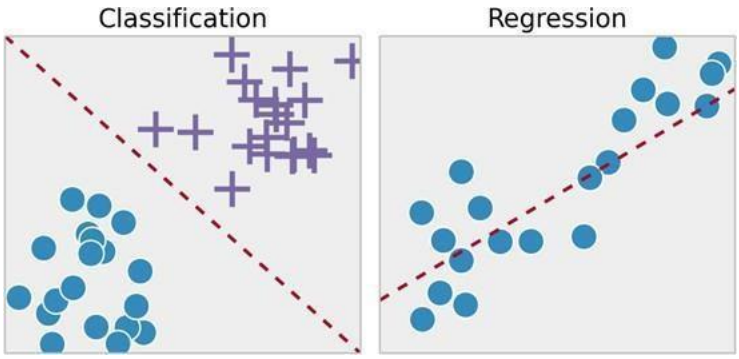


Figure II.3: Visual Comparison of Classification and Regression in Machine Learning

II. 3.3.2. Unsupervised Machine Learning Algorithms

Unsupervised Machine Learning Algorithms are used to find patterns or structures in data without using labelled outcomes. Unlike supervised learning, there is no target variable. These algorithms explore the data to identify hidden patterns, groupings, or features.

Common Types of Unsupervised Machine Learning Algorithms are Clustering Algorithms, Dimensionality Reduction Algorithms, and Association Rule Learning.

- **K-means clustering Machine Learning Algorithm**

This algorithm creates clusters, which are groups of data points arranged according to their commonalities.

K is the number of centroids considered for a given question, while 'means' refers to a centroid that is considered the centre of each cluster.

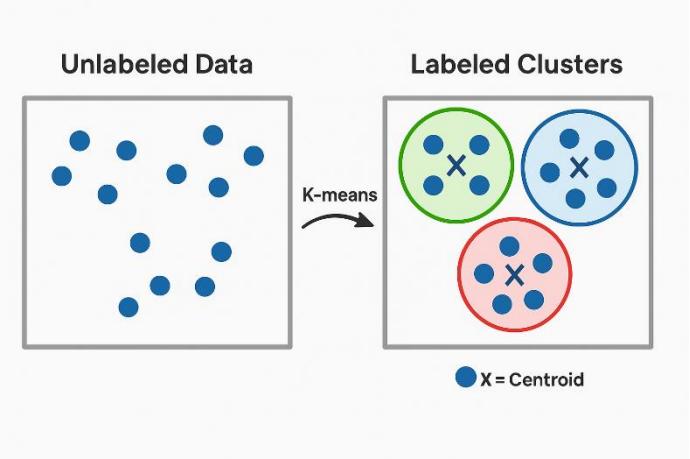


Figure II.4: Unlabelled data compared to labelled data by k-means

II. 3.3.3. Reinforcement Machine Learning Algorithms

Reinforcement learning is a type of machine learning where the goal is to maximize rewards by having the machine figure out the best behaviour in a given situation. Based on the principle of reward and punishment, a machine receives either a reward or a punishment for each decision it makes, allowing it to determine whether it is the right one or not. This teaches the machine to make the right decisions to optimize its long-term reward.

A machine using a reinforcement algorithm can be configured to prioritize either short-term or long-term rewards. When the machine is in a particular state but must take action for the next state to achieve this reward, this is called a Markov decision process.

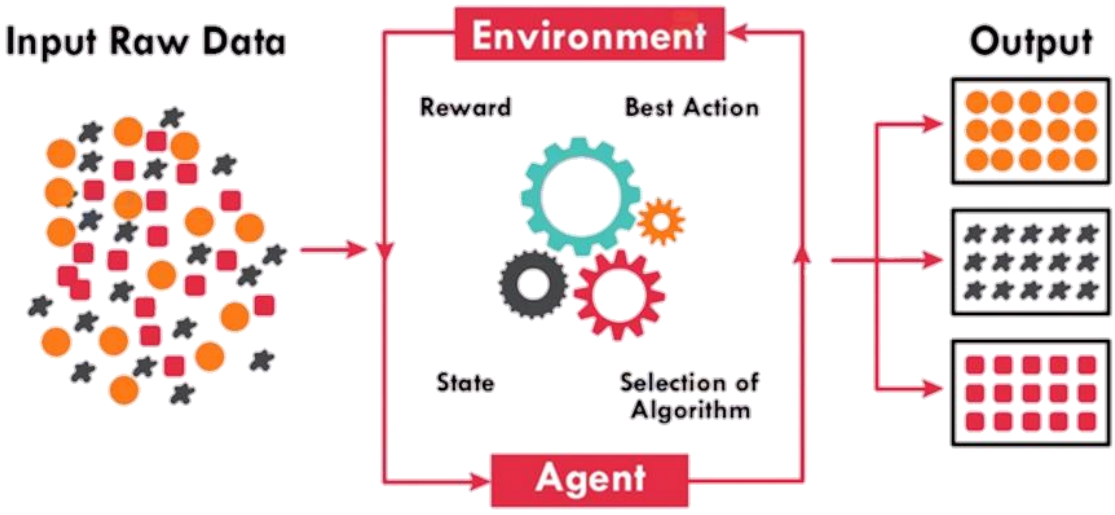


Figure II.5: Overview of the Reinforcement Learning Process

II. 4. Deep learning

A subset of machine learning called "deep learning" uses artificial neural networks to analyse data. The structure and operation of these networks should resemble that of the human brain. They are made up of various grouped neurons, or interconnected nodes. Complex data types such as text, speech, and images are particularly well suited to deep learning models. Human feature engineering is no longer necessary because the system can automatically extract relevant features from raw data.

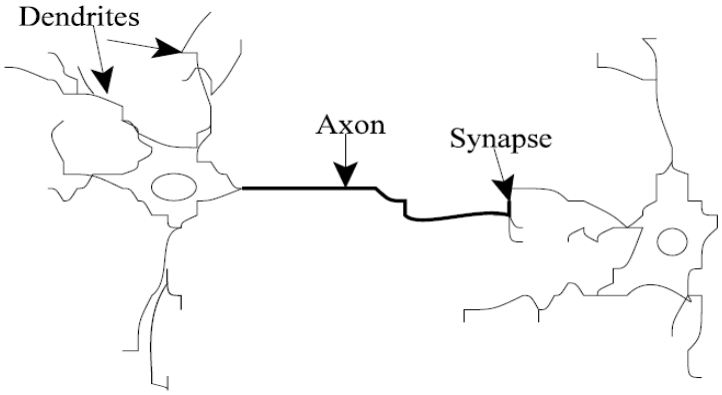


Figure II.6: A Conception of a Naturel Neuron

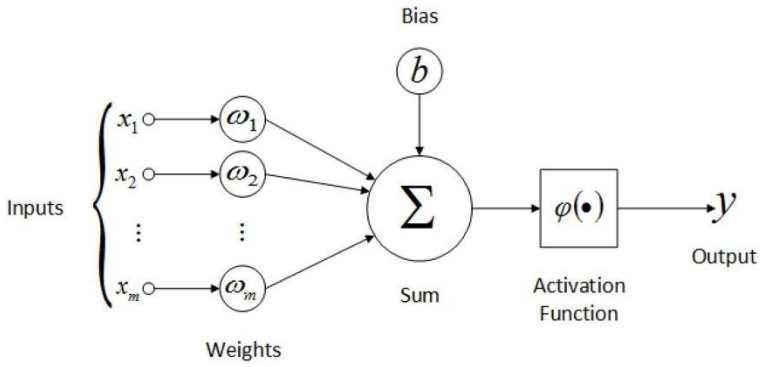


Figure II.7: An Artificial Neuron

II. 4.3. Artificial neural networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by biological neural networks, consisting of interconnected artificial neurons organized into layers (input, hidden, and output). Each connection has a weight, and neurons use activation functions to process data. ANNs learn by adjusting weights and biases through algorithms like backpropagation, enabling them to recognize complex patterns and perform tasks such as classification, regression, and anomaly detection. Their strength

lies in approximating nonlinear functions and uncovering hidden features in large datasets, making them essential for modern machine learning and AI.

II. 5. The Beginning of Deep Learning

As early as 1952, IBM's Arthur Samuel developed a programme for learning to play checkers. It could build new models by observing the moves of the pieces and use them to improve its playing skills. In 1959, Machine learning was presented as a field of research that could teach a machine a specific skill without the need for predetermined programming. As machine learning has evolved, various machine learning models have been proposed, including deep learning. Due to its complicated structure and the need for a large computation, the computational cost is very high, so it did not receive much attention in the beginning. However, with the great improvement in computing power, the excellent performance of deep learning has made it rise and become one of the hottest research areas. Main deep learning models will be briefly described, and potential prospects for deep learning development will be discussed.

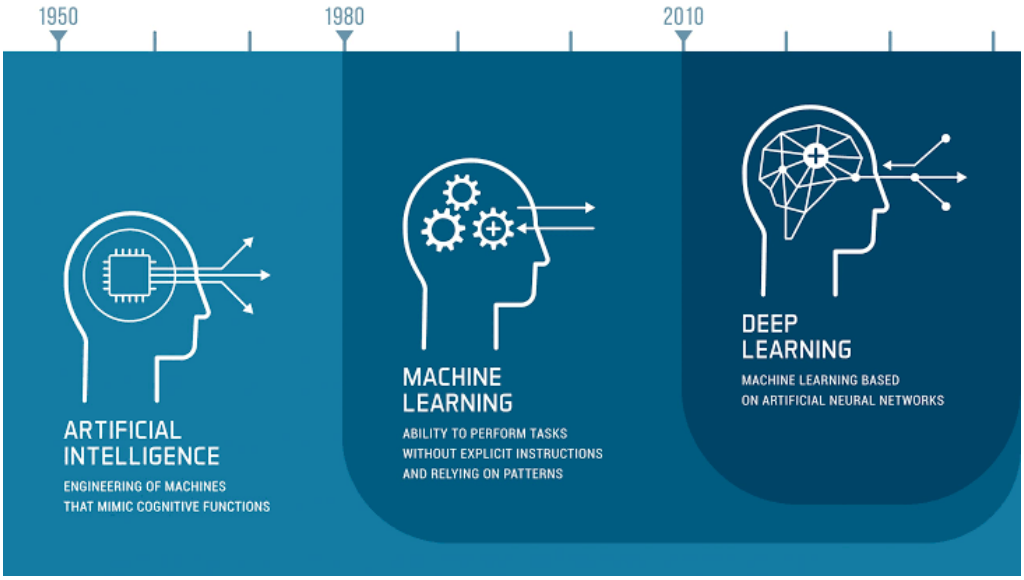


Figure II.20: Deep Learning Timeline

- **Before 1980: Deep Learning Origins**

Neurophysiologist Warren McCulloch and mathematician Walter Pitts used "threshold logic" to develop a neural network model in 1943.

Frank Rosenblatt introduces the Perceptron, an early neural network model that can learn and recognize patterns, in 1957.

Deep learning architectures see significant advancement in 1965 when Ivakhnenko and Lapa introduce multi-layered neural networks.

Kunihiko Fukushima first introduced convolutional layers in 1979 with the Neocognitron, a multi-layer hierarchical artificial neural network that served as the model for modern convolutional neural networks (CNNs) used in image recognition applications.

- **1980-2010: The Mid-journey 0f Deep Learning**

One of the earliest motivations for associative memory was the introduction by John Hopfield in 1982 of the Hopfield networks, a recurrent neural network that demonstrated how neural networks could be used for the storage and retrieval of patterns.

The Long Short-Term Memory (LSTM) network, introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997, enhances RNN training and resolves long-term dependency issues in sequence prediction tasks.

ImageNet, a massive dataset developed in 2009 by Fei-Fei Li and associates, would transform computer vision by serving as a standard for large-scale image classification and igniting advances in deep learning and computer vision models.

- **2010–Present: Deep Learning Revolution**

2012: Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton release AlexNet, a deep convolutional neural network (CNN), which demonstrates the power of deep learning for image recognition tasks by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

2014: The creation of realistic synthetic data because of Generative Adversarial Networks (GANs), a new generative modelling framework developed by Ian Goofellow and his colleagues.

2016: DeepMind’s AlphaGo defeated world Go champion Lee Sedol, demonstrating the power of deep reinforcement learning for difficult decisions.

2020s: Artificial intelligence began to evolve with the unprecedented natural language processing capabilities of large-scale AI models, including OpenAI's GPT-3, whose generated text was indistinguishable from that of humans.

II. 6. Deep Learning Architecture

II. 6.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specialized class of deep neural networks designed for processing structured grid data (or matrix format), such as images, video frames, or spectrograms, by leveraging spatially shared hierarchical feature learning. CNNs employ a series of convolutional layers that apply learnable filters (kernels) to extract local patterns through discrete convolution operations, followed by non-linear activation functions (e.g., ReLU) to introduce model capacity.

The human neural system has several layers, each responsible for performing a unique function. CNNs share a common architecture in which each layer extracts different features from the input image; it comes with three essential layers of neurons as demonstrated in figure II.29:

- Convolution
- Pooling
- Fully connected

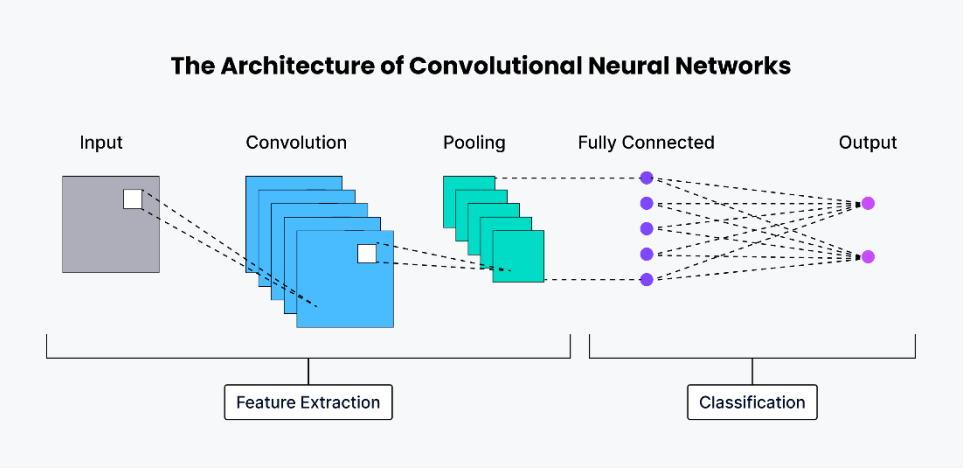


Figure II.21: The Architecture of convolutional Neural Network

- Convolution layers, which are the first few layers, are responsible for obtaining the basic features of the image, such as edges and shape.
- The pooling layers, which are the output layers responsible for reducing the size of the feature maps.
- The fully connected (FC) layer is the final layer and is responsible for assigning the image to one of the specified categories. [33]

Almost all modern pure convolutional architectures end with a fully connected layer after a single global pooling layer.

II. 6.1.1. Types of convolutional neural networks

Convolutional Neural Networks (CNNs) have evolved into diverse architectures tailored for specific tasks, computational constraints, and data modalities. Below is a taxonomy of prominent CNN variants, categorized by architectural innovation and application domain.

Here are some of the most well-known types:

1. LeNet-5 (Classic CNN)

- One of the first CNNs, created for recognizing handwritten digits.
- Simple and good for small images (like MNIST).
- **Layers**: Convolution → Pooling → Fully connected

2. AlexNet

- A deeper version of LeNet.
- Won a big image competition (ImageNet) in 2012.
- Helped CNNs become popular in computer vision.

3. VGGNet (VGG16 / VGG19)

- Uses many **small filters** (3x3) and is **very deep** (16 or 19 layers).
- Easy to understand and still used for transfer learning.

4. GoogLeNet (Inception)

- Uses a special block called **Inception Module**.
- Combines filters of different sizes in one layer (1x1, 3x3, and 5x5).
- More efficient and less expensive to compute.

5. ResNet (Residual Networks)

- Very deep (up to 100+ layers).
- Uses **"skip connections"** to solve the vanishing gradient problem.

- Makes training deep models easier and faster.

6. MobileNet

- Designed for **mobile devices** and low-power systems.
- Lightweight and fast.
- Great for real-time applications on phones or embedded systems.

Model	Key Features	Use Case
LeNet-5	Simple, early model	Handwriting recognition
AlexNet	Large, Deep model	General image classification
VGGNet	Many layers, Small filters	Transfer learning
GoogLeNet	Inception blocks	Efficient recognition
ResNet	Skip connections	Very deep models
MobileNet	Lightweight, fast	Mobile apps, edge devices

Table II.4: Comparison of Popular Deep Learning Models

II. 6.2. Recurrent Neural Networks (RNNs)

RNN is one of the most popular deep neural networks, widely used to deal with ordinal or temporal problems according to its internal memory, such as time series, speech recognition, image annotation, signal processing, or natural language processing (NLP). [34]

The architecture of a simple RNN consists of three layers, the input, recurrent hidden and output layers, as shown in Figure below. The basic idea behind the RNN is the feedback between hidden units over time, which provides a recurrence mechanism. This means that the output of recurrent neural networks depends on the previous elements within the sequence. In other words, the current input is the output of the previous time.

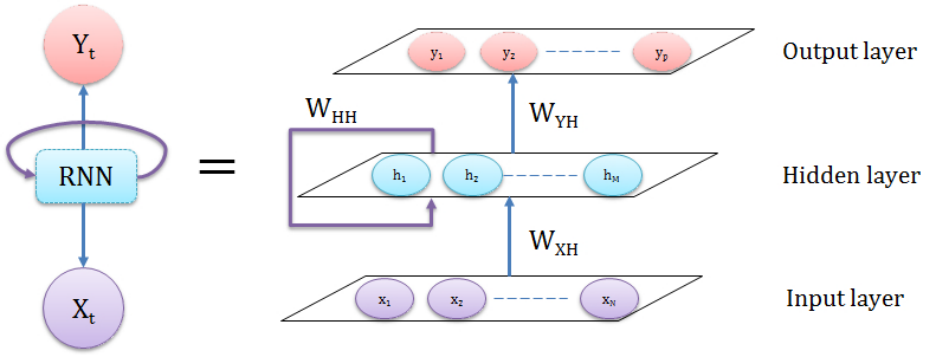


Figure II.10: An Artificial Neuron (RNN)

The input layer receives a sequence of vectors $\boldsymbol{X_t} = (\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n})$ and processes them one at a time. At each time step $\boldsymbol{t}$, the recurrent neural network is based on the input vector $\boldsymbol{X_t}$ and previous hidden state $\boldsymbol{h_{t-1}}$ to update its memory and produces a hidden state $\boldsymbol{h_t}$ as in equation II.2.

$$\boldsymbol{h_t = f\left(W_{ihx_t} + W_{hhh_{t-1}} + b_h\right)}$$

Equation II.2: Hidden State Update Equation For a Simple Recurrent Neural Network

Where $\boldsymbol{f}$ is the hidden layer activation function, $\boldsymbol{W_{ih}}$ is the weight matrix from the input-to-hidden, $\boldsymbol{W_{hh}}$ is the matrix of recurrent weights between the hidden layer and itself, and $\boldsymbol{b_h}$ is the bias vector of the hidden units. These weight matrices are known as parameters of RNN, which are calculated and optimized to obtain a powerful model.

The output layer is calculated according to the following equation:

$$\boldsymbol{O_t = g(W_{ho}h_t + b_o)}$$

Equation II.3: Output Equation of a Simple Recurrent Neural Network

Where $\boldsymbol{g}$ is the activation functions, $\boldsymbol{W_{ho}}$ is the weight matrix from the hidden-to output, and $\boldsymbol{b_o}$ is the bias vector in the output layer. [35]

II. 6.2.1. LSTM

The LSTM is a variant of RNN that is capable of learning long-term dependencies. LSTMs were first proposed by Hochreiter and Schmidhuber [36] and refined by many other researchers. They work well on a large variety of problems and are the most widely used type of RNN.

The LSTM architecture consists of a set of multiplicative gates, the Input (I), Output (O), and Forget (F) gates as illustrated in figure below.

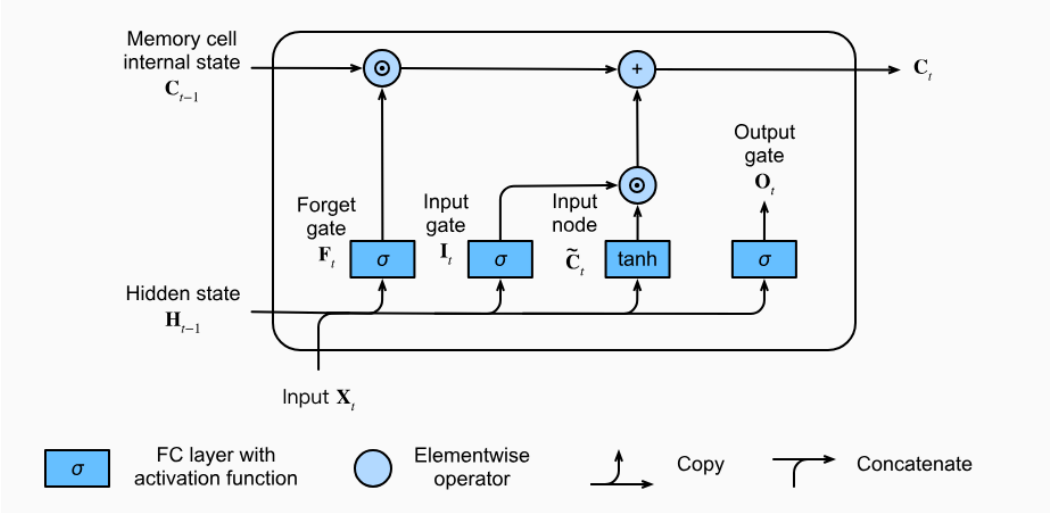


Figure II.22: Example of LSTM Memory Cell.

These gates control the circulation of the information that traverses the LSTM cell. They have the ability to keep, add, or remove information to the cell state, represented with vectors h_t, (short-term state) and C_t (long term state). The forget gate dropped a part of memory, which is from the previous long-term state C_{t-1}, short term state h_{t-1} and the input data X_t Then it adds to a new memory selected by the input gate to update the long-term state C_t. [35]

II. 6.2.2. Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN). GRU improves traditional RNNs by solving the problem of forgetting important information over time. GRUs manage information through two primary "gates." When the model receives fresh input, the reset gate assists it in determining how much of the past to forget, while the update gate determines how much of the past should be retained. [37]

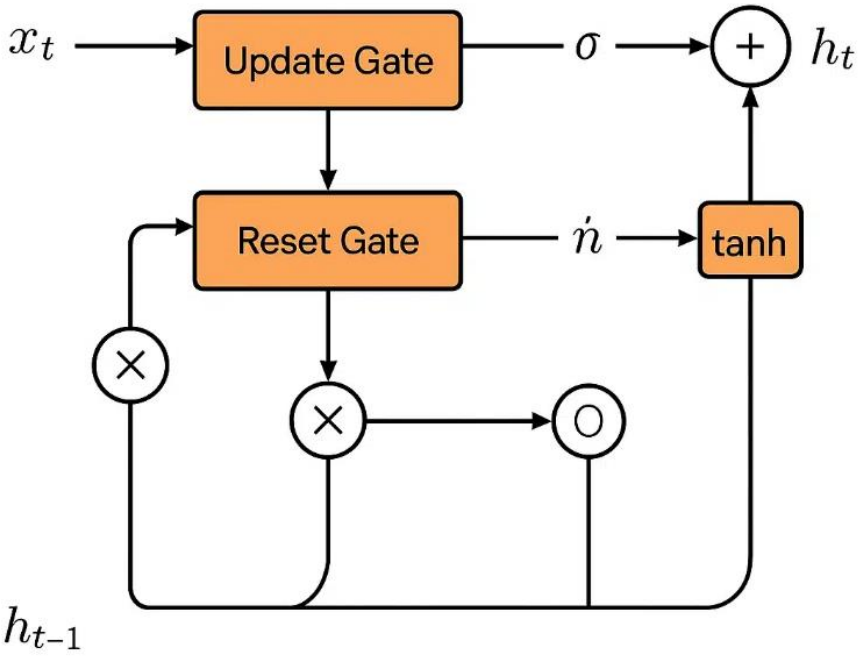


Figure II.23: Architecture of a Gated Recurrent Unit (GRU)

These gates help the GRU to remember useful information for a long time and ignore what is not important. As a result, it is good at understanding the context in sequences.

GRUs are faster and lighter than LSTMs (Long Short-Term Memory), which come with a more complicated memory structure and three gates. They typically train more quickly and require fewer parameters.

II. 6.3. Comparison of ANN, CNN, and RNN

	ANN (Artificial Neural Network)	**CNN (Convolutional Neural Network)**	**RNN (Recurrent Neural Network)**
Timeline	Back in the 1940s and 1950s	Were introduced in the **1980s**	Emerged in the **1980s** as well
Type of Data	Tabular Data, Text Data	Image and visual data	Sequential data like text, speech, or time series

Applications	Basic facial recognition, pattern recognition, classification	Advanced facial recognition, object detection, text digitization	Text-to-speech, language modelling, time series prediction
Strengths	Tolerates noisy/incomplete data, simple to implement	High accuracy for image tasks, captures spatial features, fewer parameters	Remembers previous inputs, great for context-aware tasks
Limitations	Hardware dependent, black-box behaviour	Requires large datasets, sensitive to object orientation/position	Suffers from vanishing/exploding gradients in long sequences
Core Structure	Fully connected layers (each neuron connects to all neurons in the next layer)	Uses convolution and pooling layers before final dense layers	Has loops and memory connections to retain information across time steps
Performance	ANN is considered to be less powerful than CNN, RNN	CNN is more powerful than ANN, and RNN.	RNN includes less feature compatibility when compared to CNN.

Table II.5: Comparison of ANN, CNN, and RNN

II. 8. What is Natural Language Generation

Natural Language Generation (NLG) systems use artificial intelligence and natural language processing techniques within software systems that generate texts in human languages such as English, Chinese, and Arabic. In other words, NLG is the science of AI systems that can write. As such it is related to (but not the same as) Natural Language Understanding (NLU), which is the science of AI systems that can read and extract meanings from human-written texts.

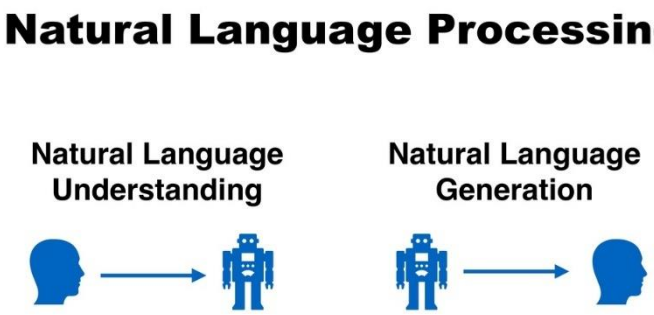


Figure II.24: Naturel Language Processing

Natural language generation has recently become more prominent because of the success of ChatGPT and other generative language models, but the field has been around for decades. AI and NLP techniques such as machine learning and language models are widely used, but they do not tell the whole story. [38]

II. 8.1. Neural networks in visual and language domains

Multiple approaches have been developed for representing images and words in higher-level representations. On the image side, Convolutional Neural Networks (CNNs) have recently emerged as a powerful class of models for image classification and object detection. On the sentence side, our work takes advantage of pre-trained word vectors to obtain low-dimensional representations of words. Finally, Recurrent Neural Networks have been previously used in language modelling, but we additionally condition these models on images. [39]

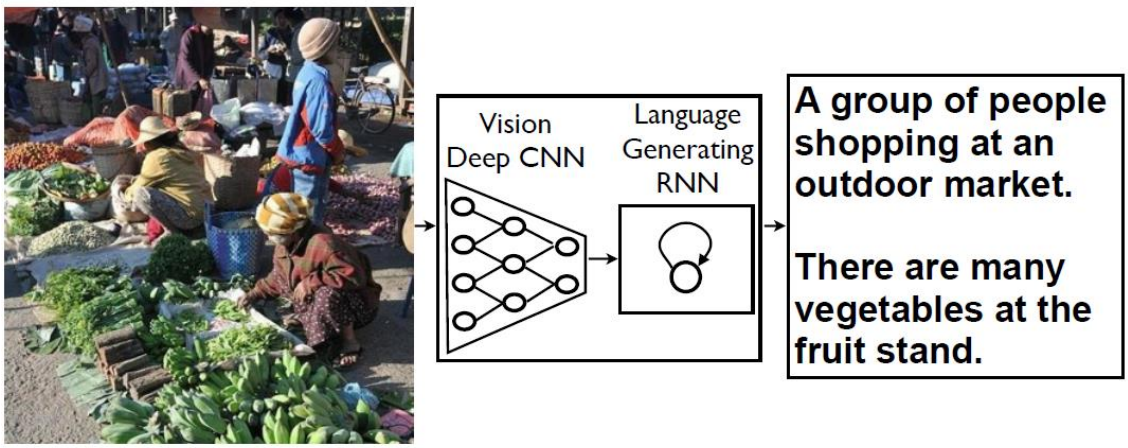


Figure II.25 : An Image Captioning Model That Combines a Deep CNN for Visual Feature Extraction and an RNN for Generating Natural Language Descriptions

II. 9. Conclusion

Artificial intelligence, machine learning, and deep learning are the foundations of modern intelligent systems, allowing machines to process data, learn patterns, and make informed decisions. This chapter has explored the basic concepts of these technologies, focusing on their different learning approaches including supervised, unsupervised, semi-supervised, and reinforcement learning as well as deep learning architectures such as convolutional neural networks and recurrent neural networks.

These models are extremely helpful when dealing with visual data, as understanding, analysing, and interpreting images is critical. The theoretical foundation in this chapter is critical to the next phase of our work. In the following chapter, we will look at image acquisition models, specifically how visual data is captured, represented, and prepared for further processing by intelligent systems.

Chapter III:

Image Captioning Models

III. 1. Introduction

Over the past two decades, the fields of natural language processing and computer vision have seen great advances in their respective goals of analysing and generating text, and generating a descriptive natural language sentence that accurately reflects the image content. While both fields share a similar set of methods rooted in artificial intelligence and machine learning, they have historically developed separately, and their scientific communities have typically interacted very little.

Recent years, however, have seen an upsurge of interest in problems that require a combination of linguistic and visual information. Many everyday tasks of this nature, e.g., following instructions in conjunction with a diagram or a map, understand slides while listening to a lecture… etc.

Image captioning is a fundamental task in computer vision and natural language processing (NLP) that involves generating a natural language description of an image. It requires a model to understand visual content (objects, actions, relationships) and express it in grammatically correct, contextually relevant text.

Despite considerable progress in English and other widely studied languages, research in Arabic remains limited. The lack of annotated datasets, combined with the linguistic complexity of Arabic, makes image captioning particularly challenging. This chapter examines the evolution of image captioning models, the unique challenges posed by the Arabic language, how modern deep learning techniques have evolved to address these issues.

III. 2. Computer Vision

Computer vision is a field that analyses digital images by combining individual pixel data to create a multidimensional representation of visual information; this field is not limited to facial recognition and autonomous vehicles but also applies to tasks like counting people in a crowded picture. The central objectives of computer vision encompass image categorization, object identification and complementary endeavours such as image division and environmental reconstitution.

Modern computer vision systems rely on deep learning networks, including Convolutional Neural Networks, which can learn patterns from raw image data.

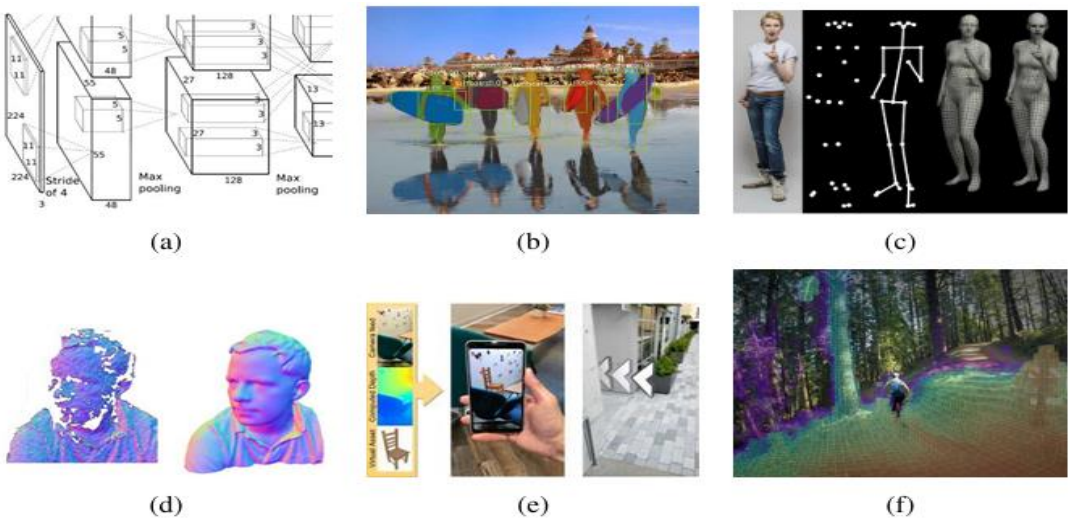


Figure III.1: **Examples of Computer Vision Algorithms from the 2010s: (a) the SuperVision deep neural network. (b) Object instance seg. (c) Whole body, expression, and gesture fitting from a single. (d) Fusing multiple colour depth images using the KinectFusion real-time system, (e) smartphone augmented reality with real-time depth occlusion effects, (f) 3D map computed in real-time on a fully autonomous Skydio R1 drone** [40]

III. 2.1. Computer vision applications

• **Motion capture (mocap):** employing vision-based methods, such as retro-reflective markers seen through multiple cameras, to record actors for computer animation;

• **Surveillance:** monitoring for intruders, analysing highway traffic and monitoring pools for drowning victims.

• **Fingerprint recognition and biometrics:** for both forensic applications and automated access authentication.

Alongside all of these commercial uses, there are countless consumer-level uses as well, like what you can do with your own images and videos. Among them are:

• **Stitching:** combining overlapping images to create a seamless panorama

• **Exposure bracketing:** creating a single, flawlessly exposed image by combining several exposures made in difficult lighting circumstances (strong sunlight and shadows).

• **Morphing:** turning a picture of one of your friends into another, using a seamless morph transition

• **3D modeling:** Utilises photogrammetry techniques, it is possible to transform one or multiple instances of photographic data, captured successively as snapshots, into a three-dimensional spatial model of the subject.

• **Video match move and stabilization:** inserting 2D pictures or 3D models into your videos by automatically tracking nearby reference points or using motion estimates to remove shake from your videos.

• **Photo-based walkthroughs:** flying between various images in 3D to explore a sizable collection of pictures, like the interior of your home.

• **Face detection:** Enhancements to camera focusing performance (automatic focus) and image search functionality.

• **Visual authentication:** When family members sit down in front of the webcam, your home computer will automatically log them in.

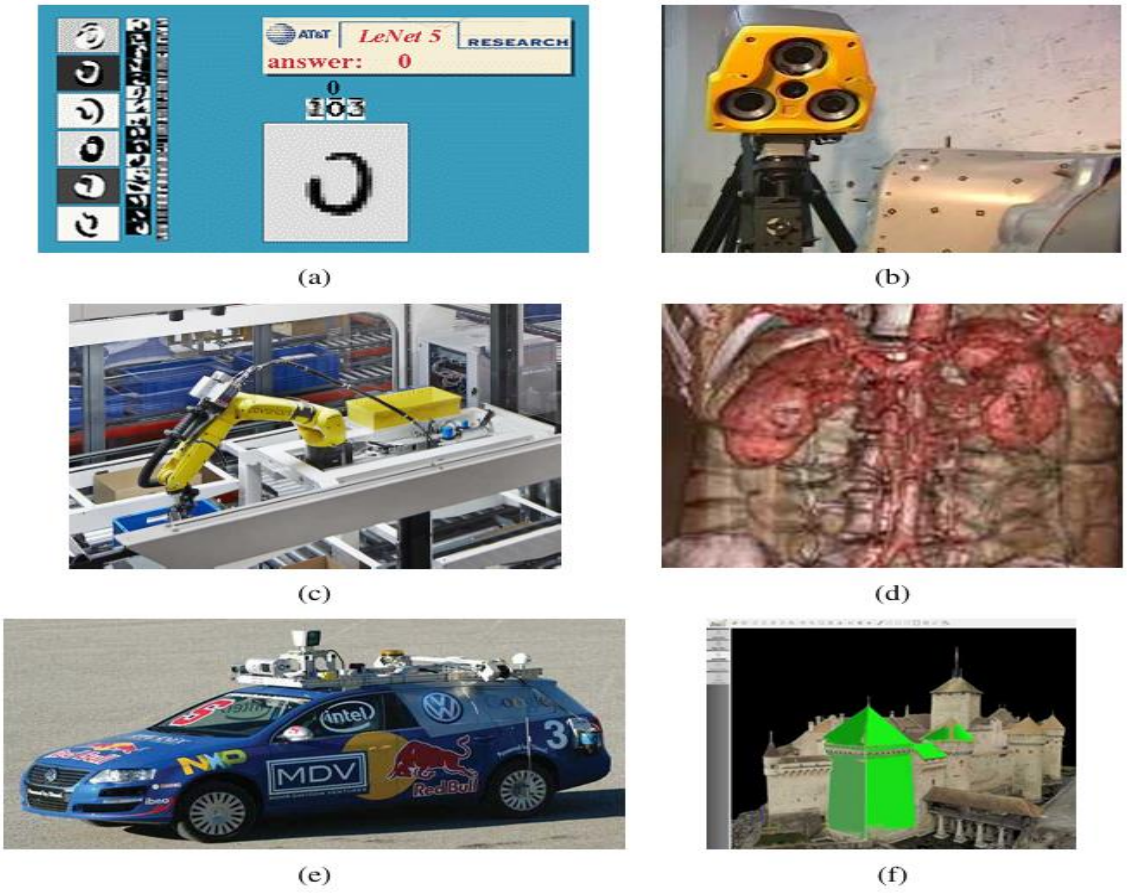


Figure III.2: Some industrial applications of computer vision: (a) optical character recognition (OCR). (b) Mechanical inspection. (c) Warehouse picking. (d) Medical imaging. (e) Self-driving cars. (f) Drone-based photogrammetry

III. 3. Naturel Language Processing (NLP)

A theoretically driven collection of computing methods for the autonomous analysis and representation of human language is known as natural language processing, or NLP. With the advent of Google and similar platforms, which can process millions of web pages in less than a second, NLP research has advanced from the days of punched cards and batch processing, when evaluating a sentence may take up to seven minutes. NLP makes it possible for computers to carry out a variety of tasks connected to natural language at all skill levels. [41]

III. 3.1. The NLP Definition

Natural language processing is a branch of computer science and AI that focuses on using machine learning to make machines able to comprehend and speak human language.

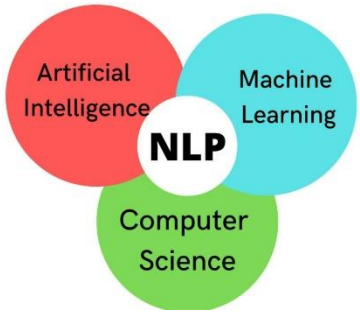


Figure III.3: Natural Language Processing: The Core of AI and Machine Learning and Computer Science

NLP combines statistical modelling, machine learning, deep learning, and computational linguistics the rule-based modelling of human language to allow computers and digital devices to recognize, comprehend, and produce text and speech.

NLP research has helped enable the era of generative AI, and its already part of everyday life for many, powering search engines, prompting chatbots for customer service with spoken commands, voice-operated GPS systems and question-answering digital assistants on smartphones such as Amazon's Alexa, Apple's Siri and Microsoft's Cortana.

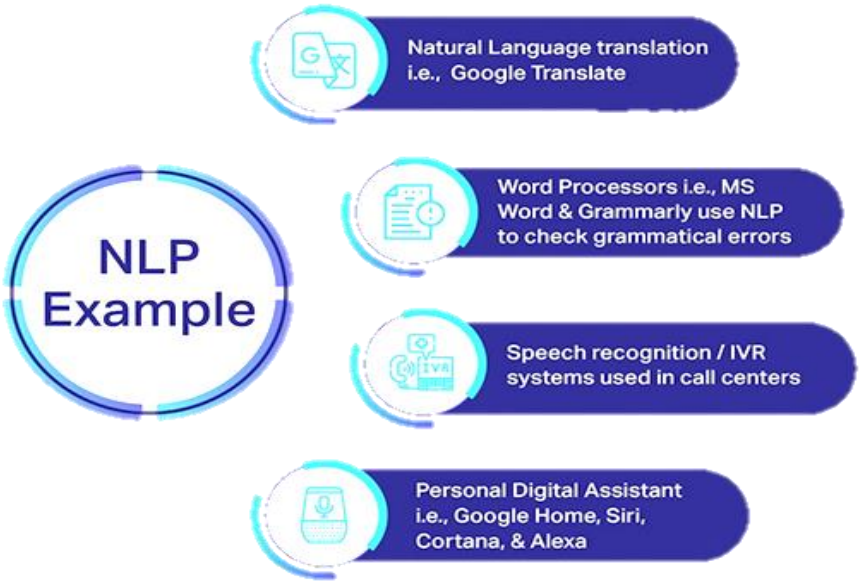


Figure III.4: Natural Language Processing: Key Applications and Examples [42]

III. 3.2. Understanding How NLP Works

NLP works by combining various computational techniques as shown in figure below to analyse, understand and generate human language in a way that machines can process,

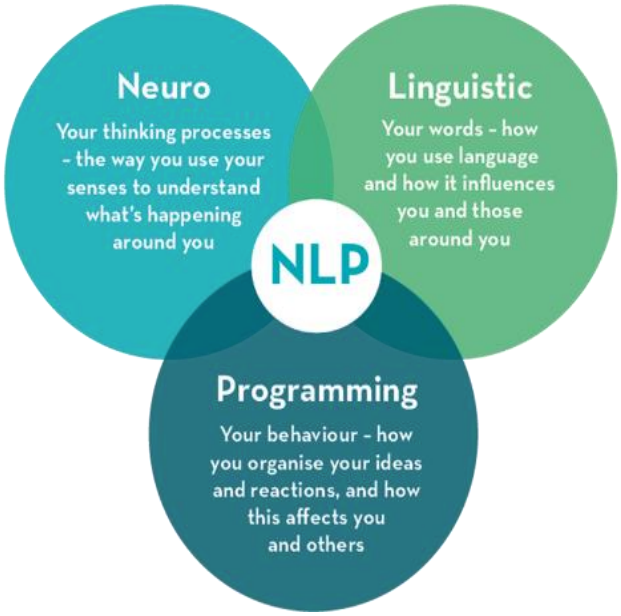


Figure III.5: NLP: Combining Linguistics, Programming, and Neuroscience

1. **Text Pre-processing**

NLP text pre-processing prepares raw text for analysis by transforming it into a format that machines can more easily understand. It begins with tokenization, which involves splitting the text into smaller units like words, sentences or phrases.

2. **Feature Extraction**

Feature extraction is the process of converting raw text into numerical representations that machines can analyse and interpret.

3. **Text Analysis**

Text analysis involves interpreting and extracting meaningful information from text data through various computational techniques. This process includes tasks such as part-of-speech (POS) tagging, which identifies grammatical roles of words and named entity recognition (NER), which detects specific entities like names, locations and dates.

4. **Model training**

Processed data is then used to train machine learning models, which learn patterns and relationships within the data. During training, the model adjusts its parameters to minimize errors and improve its performance. Once trained, the model can be used to make predictions or generate outputs on new, unseen data. The effectiveness of NLP modelling is continually refined through evaluation, validation and fine-tuning to enhance accuracy and relevance in real-world applications. [43]

III. 4. Image Captioning

In recent years, automatic image captioning has emerged as a critical research area at the intersection of CV and NLP. This task involves analysing an image's visual content such as objects, actions, and spatial relationships and generating a coherent, contextually relevant textual description.

Image captioning represents a data-to-text generation challenge, where the input is raw pixel data and the output is a natural language sentence that accurately summarizes the image's key elements, like shown in figure below. Unlike traditional computer vision tasks (e.g., object detection or classification), captioning requires multimodal reasoning, combining visual perception with linguistic fluency to produce human-like descriptions.

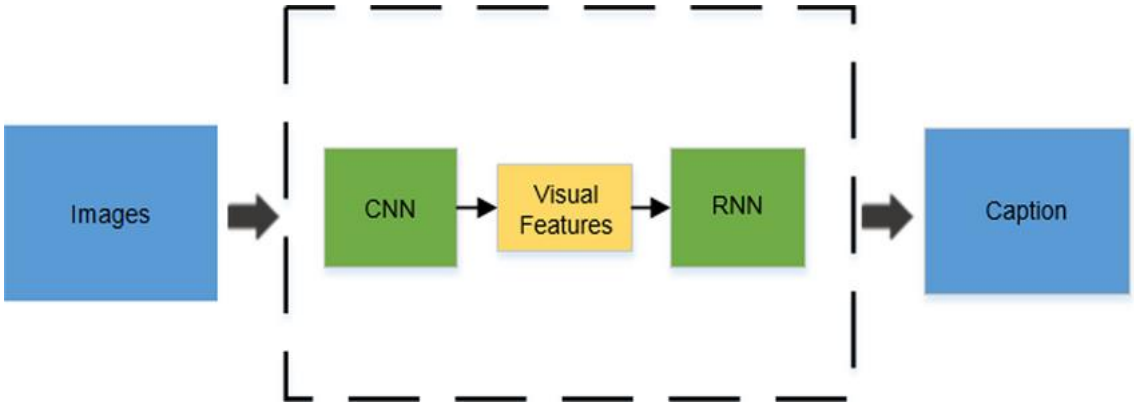


Figure III.6: **Illustration of the CNN–RNN-based Image Captioning Framework** [44]

III. 5. Historical Background

III. 5.1. Evolution of image captioning techniques

Early image captioning systems relied on handcrafted features and template-based methods. Features are extracted from input data. These data points are then submitted to a pre-existing classification model, utilising Support Vector Machines, in order to categorise the object. Feature extraction from a vast and heterogeneous dataset is unfeasible for task-specific attributes. Furthermore, real-world data like images and videos are complex and have multiple semantic interpretations. [45]

These systems generally followed one of the following approaches:

- **Template-Based Captioning**: These models detected objects, attributes, and actions using predefined rules or classifiers and filled these into sentence templates.

  Example: "A [person] is [riding] a [bicycle]".

- **Retrieval-Based Methods:** Captions were retrieved from a database of existing image-caption pairs by comparing visual features (e.g., SIFT, HOG). These methods could not generate new or creative captions and lacked generalization.
- **Pipeline Systems:** A modular approach involving object detection → scene understanding → sentence generation. However, errors at one stage propagated through the system, limiting accuracy.

III. 5.2. Deep Learning Approaches (Modern Era)

On the other hand, in deep machine learning based techniques, features are learned automatically from training data and they can handle a large and diverse set of images and videos.

The shift to deep learning introduced end-to-end trainable models that jointly learn visual and linguistic features. The most influential architecture is the encoder-decoder framework:

- **Encoder**: A CNN (e.g., ResNet, Inception) are widely used for feature learning, and a classifier such as Softmax is used for classification.
- **Decoder**: An RNN (e.g., LSTM, GRU) or Transformer generates a sentence based on those features.

- **Attention Mechanisms**: Allow models to focus on different parts of the image while generating each word (e.g., "Show, Attend and Tell", 2015).

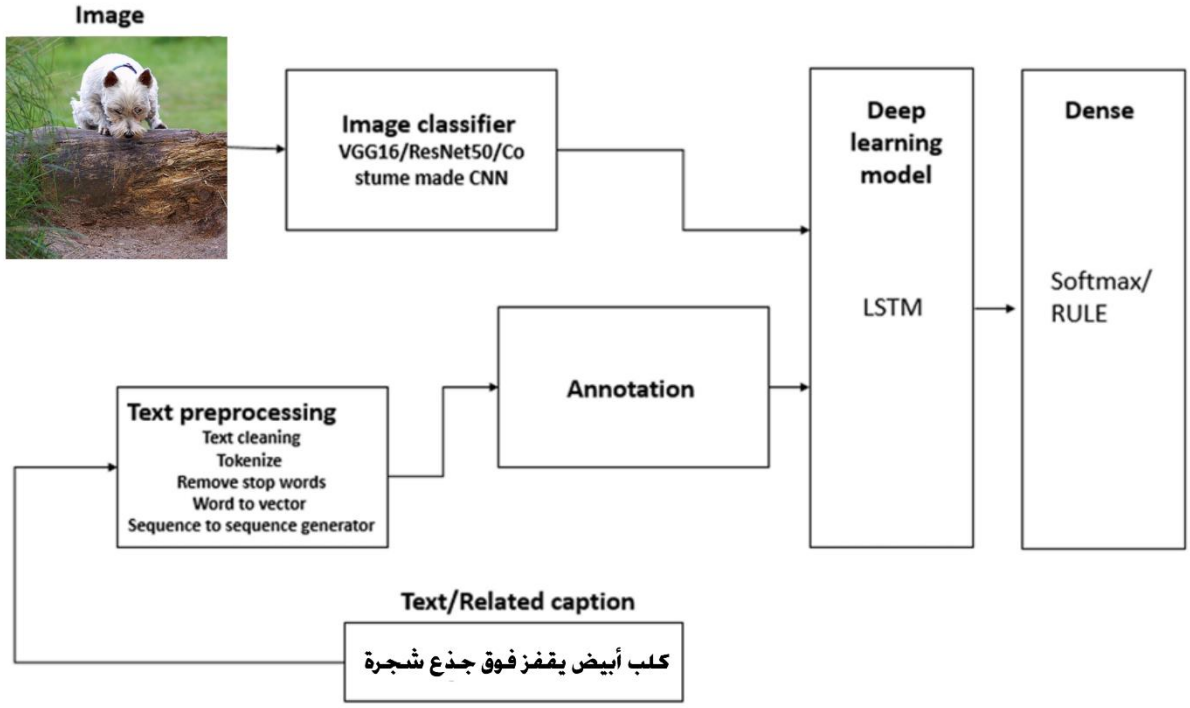


Figure III.7: Image Captioning Framework Combining CNN, LSTM, and Text Pre-processing

III. 6. Understanding How Image Captioning Works

Broadly speaking Image Captioning makes use of three primary components. First, the visual encoder is used to extract visual features from the input image. The extracted visual features are then passed to a sequence model (text decoder) in the final step sentence or captions will be generated (caption generation).

III. 6.1. Visual Encoder (Image Feature Encoder)

The process takes the original image as input and generates a compressed version that retains its key characteristics.

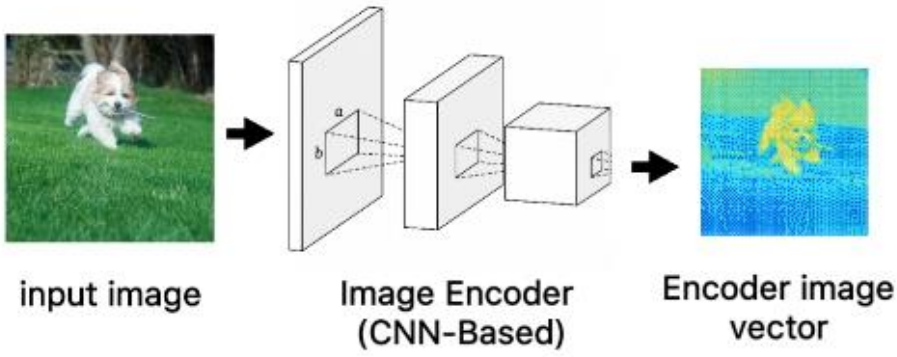


Figure III.8: Image Input to an 'Image Encoder' and Outputs an Encoded Vector

This uses a CNN architecture, and transfer learning commonly does it. It takes a pre-trained CNN model for image classification and deletes the final part known as the 'classifier'. As demonstrated in figure below.

The 'backbone' of this model is made up of many CNN blocks that gradually extract numerous aspects from the photo and build a compact feature map that captures the most relevant parts of the image.

For example, it begins by extracting simple geometric forms like curves and semi-circles in the earliest layers, then moves on to higher-level structures like noses, eyes, and hands, and finally recognizes components like faces and wheels.

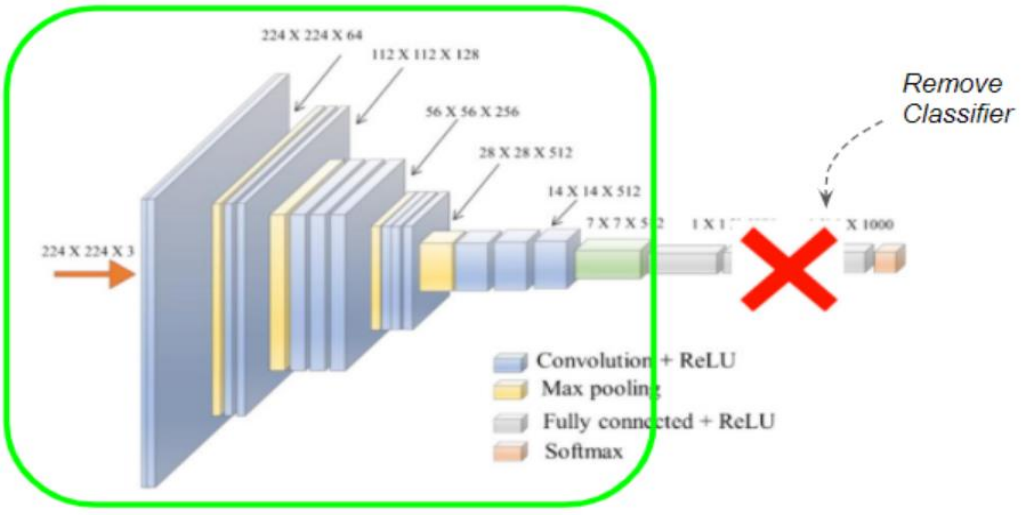


Figure III.9: Feature Extraction Using a Pre-trained CNN (Classifier Removed)

The last component of an image classification model, the classifier, receives this feature map and uses it to predict the class of the main item in the picture (e.g., vehicle or cat).

When using this model for image captioning, we are only interested in the feature map representation of the image and do not require classification prediction. So, we preserve the backbone while removing the classifier layers like the figure above.

III. 6.1.1. Vgg16

The University of Oxford's Visual Geometry Group developed the VGG-16 model. This is a convolutional neural network (CNN) architecture, with 16 layers, including 3 completely connected and 13 convolutional layers as illustrated in figure below. Its depth distinguishes it. When it comes to a variety of computer vision tasks, such as object recognition and image classification, VGG-16 is well known for its efficiency and simplicity.

The architecture of the model consists of a stack of convolutional layers with gradually increasing depth, followed by max-pooling layers. The model can learn complex hierarchical representations of visual characteristics thanks to this approach, producing predictions that are reliable and accurate. VGG-16 is still a popular option for many deep learning applications because of its great performance and versatility, even if it is simpler than designs that are more modern. [46]

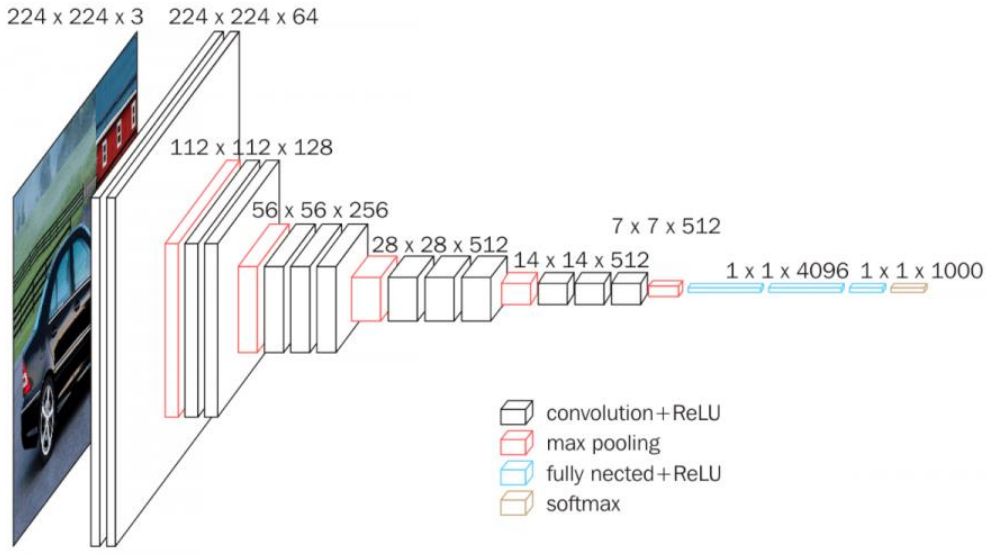


Figure III.10: A VGG16 Architecture [47]

III. 6.2. Text Decoder

The text decoder is the component responsible for generating a natural language description (caption) from the visual features extracted by the encoder.

Usually, an embedding layer feeds a stack of LSTM layers in a recurrent network model.

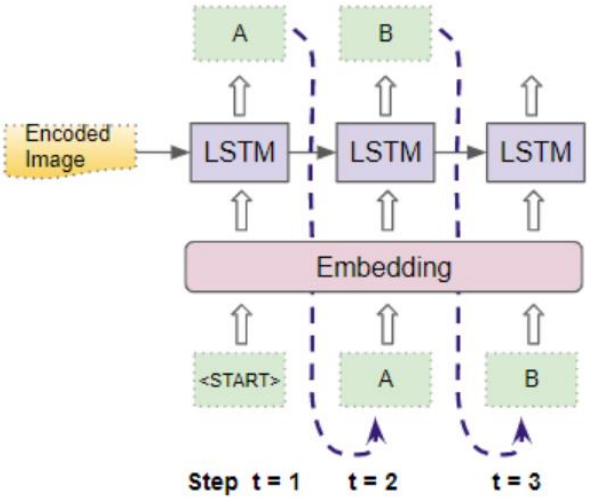


Figure III.11: LSTM Network with Initial State and Sequence so far as Input, Output Fed Back

The process begins with an image-encoded vector, seeded with a short input sequence comprising only the 'Start' token. The input image vector is 'decoded' and a sequence of tokens is output.

This prediction is generated in a loop, with one token produced at a time and fed back to the network as input for the next iteration. Consequently, at each step, it uses the previously anticipated token sequence to construct the next token in the series. Finally, an 'End' token is generated to conclude the sequence.

III. 6.2.1. Embedding in decoder sequence:

Refers to the process of converting discrete input data (like words or tokens) into dense, continuous-valued vectors (numbers) that capture semantic meaning that a neural network can understand and learn from. For example, if our vocabulary is ["cat",

"dog", "apple", "banana"]. For each word, an index is assigned: "cat" → 0, "dog" → 1, "apple" → 2, and “banana" → 3.

Instead of using these plain numbers, they are mapped to vectors:

"Cat" → [0.2, 0.1, 0.9], "dog" → [0.8, 0.5, 0.4], etc.

These vectors are learned during training and help models like LSTM understand the relationships between words. Therefore, a sentence as "cat eats apple" becomes [0, 4, 2] so it’s embedded into vectors and then fed into the model for processing.

III. 6.3. Sentence Generator

The Sentence Generator's task is to take the token sequence and produce a caption—a string of words that explains the image in the language of choice.

It is composed of a “softmax” layer after a linear layer. For each place in the sequence, this generates a probability for each word in the target language's lexicon. This probability represents the chance that the word will appear in the sentence at that particular location. Then, by selecting the word with the highest likelihood at each location, to generate the final sentence.

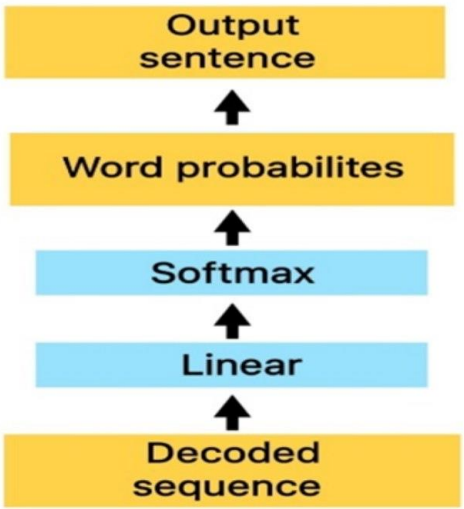


Figure III.12: Process of Generating an Output Sentence in a Sequence-to-Sequence Model from the Decoded Sequence through Linear Transformation, Softmax Probability Distribution, and Search Decoding to the Final Sentence.

III. 7. Image Captioning Architectures

III. 7.1. Encoder-Decoder architecture

The 'inject' architecture is commonly said to as the most popular deep learning architecture for picture captioning. As previously mentioned, it establishes a direct connection between the image feature encoder, the sequence decoder, and the sentence generator.

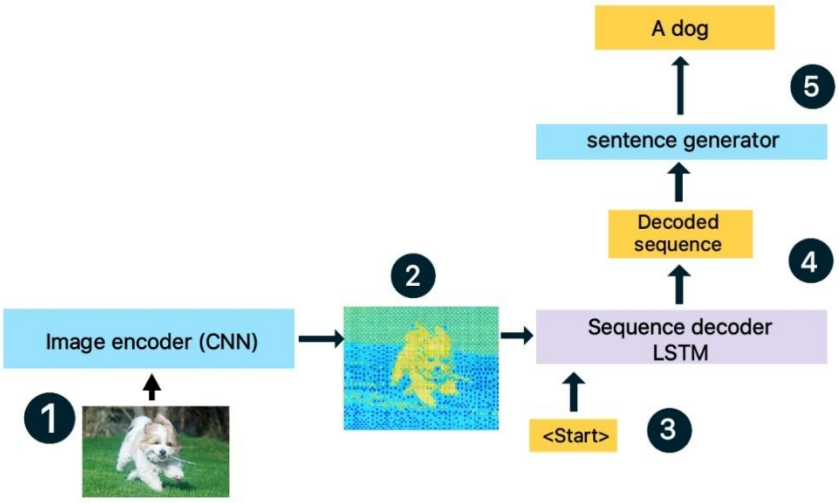


Figure III.13: Image Feature Encoder Connected to Text Generator

III. 7.2. Multi-Modal architecture

Image captioning was first implemented using the Inject architecture, which is still widely used today. Nevertheless, a different approach known as the "Merge" design has been discovered to yield superior outcomes.

Instead of progressively linking the Image Encoder as the Sequence Decoder's input, the two parts function separately. To put it another way, it does not combine the two modes—that is, text and graphics.

The LSTM network only works with the sequence that has already been created, while the CNN network just processes the picture.

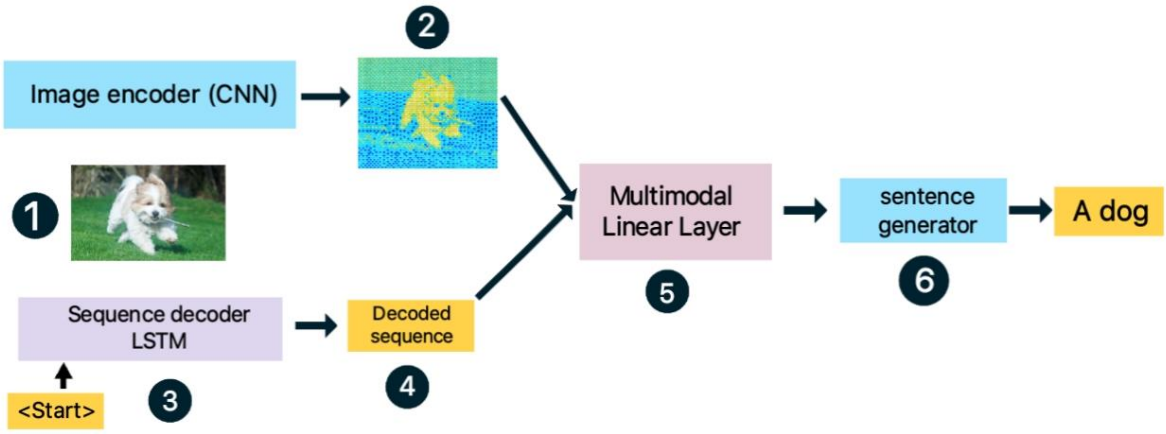


Figure III.14: A Multimodal Architecture

A Multimodal layer which may consist of a Linear and Softmax layer then combines the outputs of these two networks. It performs the task of understanding both outputs, after which the Sentence Generator generates the final caption prediction.

This method has the added benefit of enabling us to apply transfer learning to both the image encoder and the sequence decoder. For the Sequence Decoder, it may make use of a language model that has already been trained.

III. 7.3. Object Detection backbone architecture

We previously discussed utilizing an image encoder that uses the backbone of a pre-trained image classification model. Typically, this kind of model is trained to recognize a single class for the whole image.

Nevertheless, you will probably have more than one object of interest in most pictures. Why not use a pre-trained object detection backbone to extract features from the image instead of an image classification backbone?

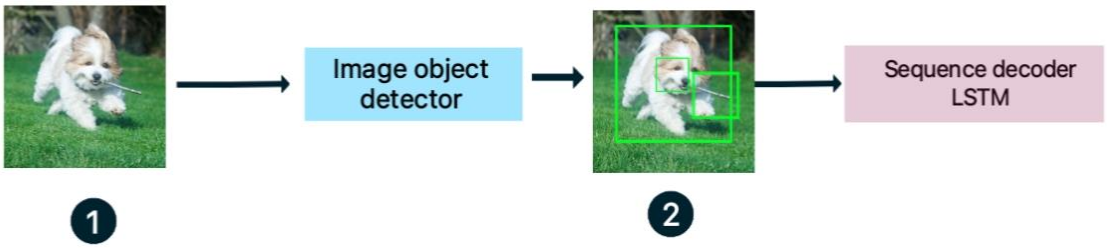


Figure III.15: Object Detection backbone

Bounding boxes are created around each of the scene's notable items using the Object Detection model. It detects the relative locations of various things in the image in addition to labelling them. As a result, it may offer a more detailed encoded representation of the picture, which the Sequence Decoder can utilize to add a description of each of those items.

III. 7.4. Encoder-Decoder with Attention

The use of attention to NLP models has been increasingly popular in recent years. It has been discovered to greatly enhance NLP applications' performance. Attention assists the model in concentrating on the words from the input sequence that are most pertinent to each output word as it creates each one.

Therefore, it should come as no surprise that attention has been used to achieve state-of-the-art achievements in image captioning as well.

Attention helps the Sequence Decoder focus on the area of the image that is most pertinent to the word it is creating as it generates each word of the caption.

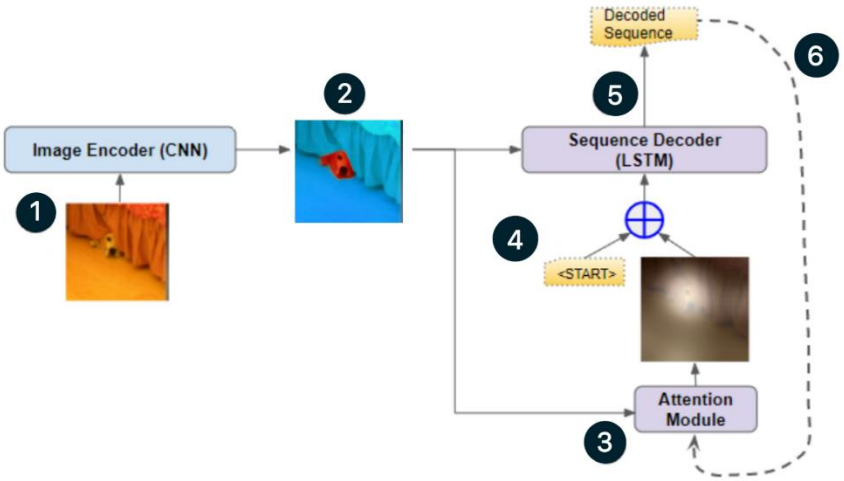


Figure III.16: Image Caption Architecture with Attention Module

The attention module receives the current output token from the LSTM and the encoded picture vector. A weighted Attention score is generated. The weight of the pixels that the LSTM should pay attention to when predicting the next token is increased when that score and the picture are combined.

For example, the model concentrates on the dog in the picture while it creates the word "dog" for the caption "The dog is behind the curtain." while you might imagine, it then turns its attention to the curtain when it reaches the word "curtain."

III. 7.5. Encoder-Decoder with Transformers

The Transformer is without a doubt the current titan when it comes to attention. Its major focus is attention, and it does not make use of the Recurrent Network, which has long been a mainstay of NLP. With the Transformer in place of the LSTM, the design is quite similar to the encoder-decoder.

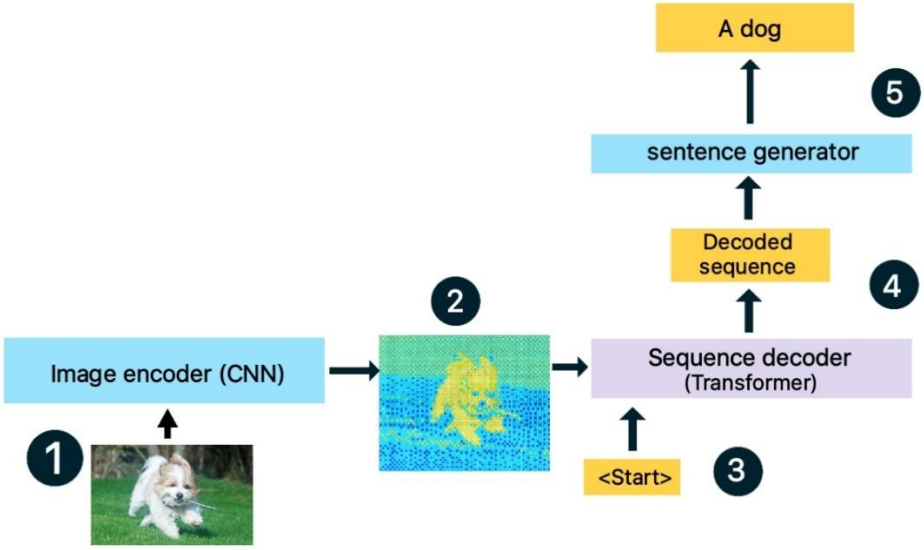


Figure III.17: Image Caption Architecture with Transformer

To solve the Image Captioning issue, many transformers architectural variations have been put forth. In order to comprehend the scene, one method is to encode not only the individual items in the picture but also their spatial connections. When creating a caption, for example, knowing if an object is next to, behind, or under another object offers helpful context.

III. 7.6. Dense Captioning architecture

Dense Captioning is another variation of the object detection technique. The premise is that a photograph frequently contains a diverse array of items and activities in several locations.

As a result, it can represent several captions for various areas of the image rather than simply one. It is able to catch every detail in the image thanks to this model.

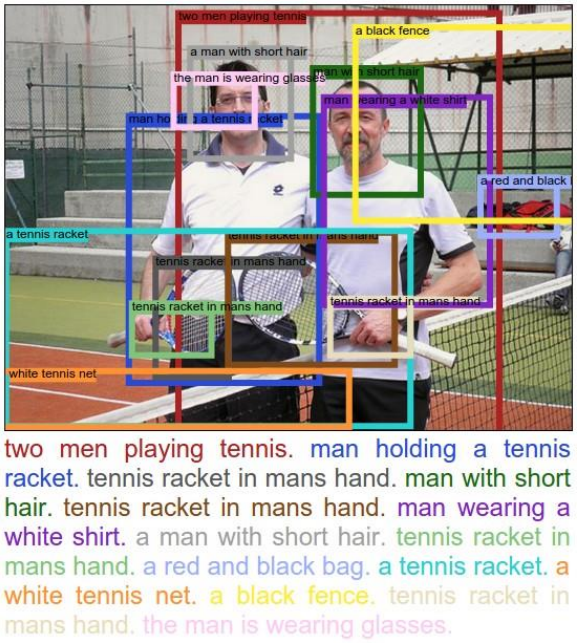


Figure III.18: Two Examples of Dense Captioning [48]

III. 8. Evaluation of Image Captioning Models

To evaluate captioning systems, both automatic metrics and human judgment are employed:

III. 8.1. Evaluation of Image Captioning using BLEU

The BLEU score evaluates the quality of machine-generated text (like image captions) by comparing them to human-written references. [49]

Bilingual assessment understudy is an algorithm for assessing the quality of content that has been machine-translated from one natural language to another.

The metric is built around the well-known precision measure. To calculate precision.

The core equation:

$$\boldsymbol{BLUE = BP \cdot exp\left(\sum_{n=1}^{N} w_n \ log\, p_n\right)}$$

Equation III.4: Calculating Precision of a BLEU Metric Equation

- BP = Brevity penalty
- pn= Precision of modified n-grams of different lengths (for 1-gram, 2-gram, ..., N-gram)

- wn = Weight of each n-gram precision (typically $1/N$)
- N = Max n-gram order (e.g., 2 in BLEU-2)

Precision is determined by dividing the total number of n-grams from a candidate that appears in the references by the total number of n-grams in the candidate. Overgeneration of the same n-gram, however, might result in great precision that is detrimental. To address this, the maximum number of times the n-gram appears in any reference is trimmed from the total match count of each candidate n-gram. For example, the count of repeated n-grams is pruned.

$$\boldsymbol{pn} = \frac{\sum \boldsymbol{ngram} \in \boldsymbol{C}^{\min(\boldsymbol{Count}_{\boldsymbol{clip}}, \boldsymbol{Count}_{\boldsymbol{ref}})}}{\sum \boldsymbol{ngram} \in \boldsymbol{C}^{\boldsymbol{Count}_{\boldsymbol{cand}}}}$$

Equation III.5: A Modified n-gram Precision Used in the BLEU Score

Brevity Penalty (BP**)** handles the issue of very short generated captions getting unfairly high scores:

$$\boldsymbol{BP} = \begin{cases} \boldsymbol{1}, & \boldsymbol{if\ c > r} \\ \boldsymbol{e}^{\boldsymbol{1-\frac{r}{c}}}, & \boldsymbol{if\ c \leq r} \end{cases}$$

Where c = length of candidate sentence, r = length of reference sentence.

- **Example:**

Targeted (or Reference) = "a person is riding a horse"

Predicted (or Candidate) = "a man riding a horse"

Step 1: Tokenize

We split each sentence into words.

- Candidate: ["a", "man", "riding", "a", "horse"]
- Reference: ["a", "person", "is", "riding", "a", "horse"]

Step 2: 1-gram precision (words)

Words in candidate: “a, man, riding, a, horse”

Count how many of them appear in any reference (clipped):

- a → appears (we only count it 1 time even if it's repeated)
- man → appears
- riding → appears
- horse→ appears
- **Matched 4 out of 5** (one of the two "a" was clipped)

1-gram precision= 4/5= 0.8

Step 3: 2-gram precision (pairs of words)

Candidate 2-grams:

- a man, man riding, riding a, a horse

Compare to 2-grams in references:

- Reference: a person, person is, is riding, riding a, a horse

Matches:

- riding a → found
- a horse → found

Two matched out of 4

2-gram precision= 2/4= 0.5

Step 4: Brevity Penalty (BP)

- Candidate length c=5
- reference length r=6

$BP = e^{(1-\frac{r}{c})} = e^{(1-\frac{6}{5})} = e^{(-0.2)} \approx 0.8187$

- **Step 5: BLEU Score**

We use only up to bigram (1-gram & 2-gram), so:

$$\boldsymbol{BLUE = BP \cdot exp\left(\frac{1}{2}\log 0.8 + \log 0.75\right)}$$

$$\boldsymbol{=0.8187 \cdot \exp\,(1/2(-0.2231-0.2877))}$$

$$\boldsymbol{= 0.8187 \cdot \exp\,(-0.2554) \approx 0.8187 \cdot 0.7746}$$

$$\boldsymbol{\approx 0.6342}$$

Therefore, the final BLEU Score is 63.4% that means the candidate is close to the human references.

III. 8.2. Evaluation of Image Captioning using AraBERT

AraBERT is a transformer-based model like BERT but trained specifically on Arabic text. It is used for various NLP tasks; like text classification, named Entity Recognition, even Sentiment analysis, and sometimes caption evaluation or generation.

In this method, we cannot use the BLEU traditional n-gram method. AraBERT is a deep contextual language model, so if we want to evaluate using AraBERT.

The most common and reliable way to use AraBERT for evaluating sentence similarity is:

Cosine Similarity between Sentence Embeddings

1. Take two Arabic sentences:
   - A candidate/predicted one
   - A reference/target one
2. AraBERT is use to turn each sentence into a vector (embedding).
3. Compare the two vectors using cosine similarity.
4. A result close to one means very similar; close to zero means very different.

For example:

- Candidate (predict): "رجل يركب حصاناً"
- Reference (target): "شخص يركب حصاناً"

When simulate AraBERT turns these Arabic sentences into embedding vectors with 768-dimensional. For more clarity, only three dimensions are used.

Step 1: Simulated Sentence Embeddings

Sentence	Embedding Vector (simulated)
"رجل يركب حصاناً"	$\vec{A}$= [0.3, 0.7, 0.5]
"شخص يركب حصاناً"	$\vec{B}$= [0.28, 0.72, 0.48]

Table III.1: An Example of Simulated Sentences Embeddings

Step 2: Cosine Similarity Formula

$$\mathbf{CosineSimilarity}(\vec{\mathbf{A}}, \vec{\mathbf{B}}) = \frac{\vec{A} \cdot \vec{B}}{\| \vec{A} \| \cdot \| \vec{B} \|}$$

Step 3: Dot Product $\vec{A} \cdot \vec{B}$

(0.3×0.28) + (0.7×0.72) + (0.5×0.48) = 0.504+0.24 = 0.828

Step 4: Vector Norms

Norm of $\vec{A}$:

$$\| \vec{A} \| = \sqrt{\mathbf{0.32^2 + 0.72^2 + 0.52^2}} = \sqrt{0.83} \approx 0.911$$

Norm of $\vec{B}$:

$\|\vec{B}\| \approx 0.9095$

Step 5: Apply the Formula

$$\mathbf{CosineSimilarity}(\vec{\mathbf{A}}, \vec{\mathbf{B}}) = \frac{0.911 \times 0.9095}{0.828} \approx 0.9993$$

Therefore, the final AraBert Score is **99.9%** so the predicted sentence and the reference are extremely semantically similar

Other methods:

- **METEOR**: Considers stemming and synonym matching, which makes it more suitable for languages like Arabic where multiple forms of a word can be valid [50]

- **ROUGE**: Focuses on recall of overlapping units. Commonly used in summarization tasks, but applicable to captioning as well.
- **CIDEr**: Designed specifically for image captioning. It weights n-gram matches using TF-IDF to emphasize informative words and downplay common ones [51]

Despite these tools, automatic metrics often fail to fully capture caption quality, especially in morphologically complex languages. Human evaluation remains critical to assess fluency, adequacy, and contextual relevance.

III. 9. Approaches for Morphologically Complex Languages

To address these linguistic challenges, several strategies have been developed:

- **Subword Tokenization**: Byte Pair Encoding (BPE) and SentencePiece divide words into more manageable units, improving generalization and reducing vocabulary size. This has proven useful in handling rare and compound words in morphologically rich languages [52]

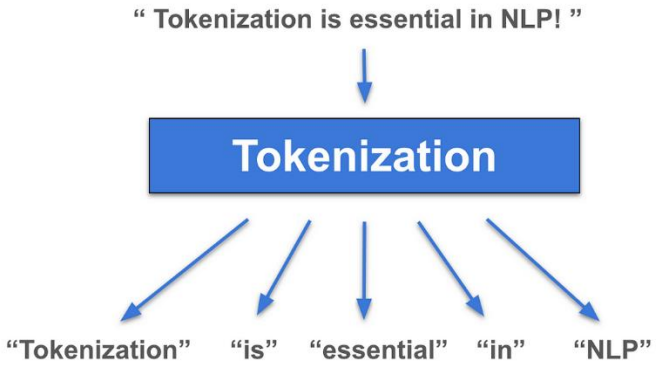


Figure III.19: Example of Tokanization

In some tokenizers (like BPE used in BERT/Arabert), each word might be broken further into subwords, but the idea remains the same <start> and <end> help define the boundaries.

<start> helps the model know when to **begin generating** a sentence.
<end> tells the model when to **stop generating**.

- **Character-Level Embedding**: Some models represent input at the character level instead of words, offering more control over morphology and spelling.
- **Morphological Analyzers**: Tools like MADAMIRA and Farasa analyze words into roots, stems, affixes, and POS tags, providing valuable features to the language model, especially when labeled data is limited
- **Transfer Learning**: Pretrained multilingual models such as mBERT, XLM-R, or mT5 allow for zero-shot or few-shot learning by transferring knowledge from high-resource languages to Arabic. These models can be fine-tuned on Arabic captions, even with smaller datasets

III. 10. Arabic Image Captioning

Arabic image captioning is an emerging field in computer vision (CV) and natural language processing (NLP) that focuses on automatically generating descriptive, contextually accurate captions for images in the Arabic language. This task involves complex challenges due to lack of labelled data in Arabic, Arabic's morphological richness, right-to-left (RTL) script, and dialectal variations.

The Arabic image captioning has many benefits such as:

- **Accessibility**: Helps visually impaired Arabic speakers understand visual content.
- **Content Indexing**: Improves Arabic image search and retrieval.
- **Cultural Relevance**: Captions must align with Arabic linguistic and social norms.

III. 10.1. Challenges in Arabic Image Captioning

The Arabic language presents unique challenges due to the complexity of the Arabic language and the limitations of existing vision-language models.

III. 10.1.1. Scarcity of Annotated Datasets

While datasets like MS-COCO and Flickr30K have fueled the progress of English captioning systems, equivalent Arabic datasets are scarce. The lack of large-scale, high-quality image-caption pairs in Arabic limits the ability to train deep models from scratch

and necessitates alternative strategies like translation or transfer learning, but we translated our dataset that's we going to use in the next chapter.

- Person in red and black parka climbing snow covered hill.
- A person in a black and red jacket walking up a snowy mountain.
- A person is walking up a snowy hill
- A person wearing a red jacket climbs a snowy hill.
- Mountain climber climbing snowy mountaintop.

- شخص يرتدي سترة سوداء وحمراء يمشي على جبل ثلجي.
- شخص يمشي على تلة ثلجية
- شخص يرتدي سترة حمراء يتسلق تلة ثلجية.
- متسلق الجبال يتسلق قمة الجبل الثلجي.
- شخص يرتدي سترة حمراء وسوداء يتسلق تلة مغطاة بالثلوج

- Two children play soccer in the park.
- Two children playing with a ball on the grass.
- Two children playing with a ball on the grass.
- Two boys in a field kicking a soccer ball.
- Two little boys are playing outside with their soccer ball on the green grass.

- طفلان يلعبان كرة القدم في الحديقة
- طفلان يلعبان بالكرة على العشب
- طفلان يلعبان بكرة القدم على العشب.
- صبيان في الملعب يركلان كرة القدم.
- يلعب طفلان صغيران بالخارج بكرة القدم على العشب الأخضر.

Figure III.20: Examples of Two Images with Their Caption while Using our Translation to Create the Arabic Dataset

III. 10.1.2. Morphological Complexity

One of the major challenges in Arabic image captioning is the language's rich morphology. Arabic words are typically derived from root-pattern structures that yield multiple forms through affixation and internal changes. This leads to a large vocabulary with sparse distribution, making it difficult for models to learn effective language representations.

For example, the root "k-t-b" can generate many words like "كتب" (he wrote), "كتاب" (book), and "مكتبة" (library) as shown in figure 36. Each form may vary based on gender, number, and case, further complicating learning in low-resource environments.

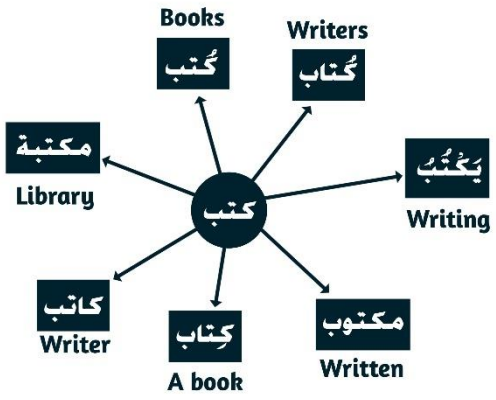


Figure III.21*:* An example of the Lexical/Morphological Sparsity

III. 10.1.3. Linguistic Variation and Dialects

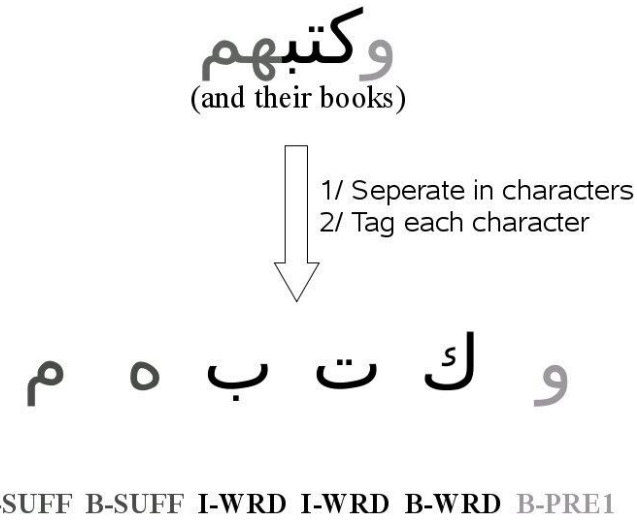
Arabic includes various dialects and Modern Standard Arabic, causing differences in grammar and vocabulary. This raises concerns about which variant a model should use with some formal systems relying on one variant and real-world applications containing mixed and inconsistent dialects. [53]

The Arabic language unlike English language deals with the male and females individually for example in Figure below our translate is the right one.

Figure III.22: A child in a pink dress is climbing up a set of stairs in an entryway. Automatic Translation: طفل يرتدي ثوبًا ورديًا يتسلق مجموعة من السلالم في مدخل المنزل. Native Speaker Captioning: طفلة ترتدي ثوبًا ورديًا تصعد مجموعة من السلالم في مدخل المنزل

III. 10.1.4. Tokenization and Embedding Issues

Traditionally, Arabic tokenization is hard due to Arabic its agglutinative nature and scripes when handling Arabic due to its script and characteristics. Models can misclassify words with the same roots unless segmented correctly. Tools for morphological analysis assist Arabic text preparation.



III. Figure III.23: **Character-Level Segmentation and Tagging of an Arabic Word in NLP**

Figure III.24: Positional Forms of Arabic Letters Based on Context

III. 11. Conclusion

This chapter looked at how image captioning models have evolved from simple rule-based systems to complex deep learning systems. It highlighted the difficulties of applying these models to Arabic due to its complex language, limited resources, and differences in dialect.

Current best practices use a combination of tools for breaking words into hunks, sub-words modelling and transferring knowledge from models that work on multiple languages. Evaluation metrics help compare results but human evaluation is still needed for a detailed understanding.

CHAPTER IV:

Arabic Image Captioning Model Implementation

IV.1. Introduction

The implementation and evaluation of deep learning models for image description is a crucial step in evaluating their practical virtue, particularly in morphologically complex languages such as Arabic. This chapter centres on the realisation of an image captioning system using the Arabic language, covering the technical pipeline from dataset preparation to final model evaluation. The main objective of this chapter is to explore how neural networks can be adapted to generate coherent and contextually appropriate Arabic captions for images.

In the domain of generating image descriptions, model evaluation plays an essential part. It not only provides insights into the precise results of the model but also underlines its challenges and sectors of improvement. Due to being morphologically rich and context-behaving, the Arabic language presents unique challenges in data representation and model training alike. Therefore, a thorough assessment is crucial to ensure that the model can handle the syntactic and semantic nuances of the language effectively.

This final chapter highlights the sequential process of implementing an Arabic image-captioning model, beginning with dataset preparation, providing details on the model's architecture, training processes, and hyperparameter tuning strategies used. Lastly, the model's performance is accurately analysed using established metrics such as BLEU along with a qualitative evaluation of generated descriptions. Throughout this broad evaluation, this project intends to clarify both the strengths and imperfections in the model, paving the way for future development and extensive applications in Arabic image understanding.

IV.2. Data Preparation

The quality, accuracy, and linguistic richness of training data were crucial to the success of any deep learning model, especially in image captioning tasks where illustrations must be effectively placed together with natural language. This work was carefully considerate of the selection and understanding of the dataset, its annotation process, and the associated challenges.

IV.2.1. Collecting and annotating training data

- **Dataset descriptions**

This project utilised a publicly available dataset provided by Obeida ElJundi Arabic image captioning repository [54], which was built on the globally acknowledged Flickr 8K dataset. The original Flickr 8K dataset contained over 8000 images, each annotated with five English captions describing objects and scenes in everyday life. [55]

- **Annotation process**

The process followed a semi-automated approach. Initially, captions from the Flickr dataset [55] were translated into Arabic using machine translation tools. However, to refine the linguistic accuracy and semantic naturalism, these primary translations were manually evaluated and improved by native Arabic speakers.

Image	English Caption	Machine Translation	Human Evaluation
	Theatre in the Roman ruins of Timgad, Algeria.	مسرح في الآثار الرومانية بتيمقاد، الجزائر.	آثار المسرح الروماني بالمدينة الأثرية الرومانية تيمقاد، الجزائر

Table IV.1: Image Description in English and Arabic with Machine Translation and Human Evaluation [56]

This hybrid method allowed the creators to scale up the annotation procedure while preserving a high degree of naturalness and cultural relevance in the Arabic descriptions. The final dataset mirrors the nuances of the Arabic language, including its rich morphology and syntactic form.

IV.2.2. Textual Data Preprocessing

To ensure the quality and improved training results, we performed comprehensive pre-processing of the Arabic captions— a crucial step to reduce noise and prevent errors that could hinder the learning process.

The figure below, demonstrates the caption cleaning procedure to ensure the consistency of the textual data:

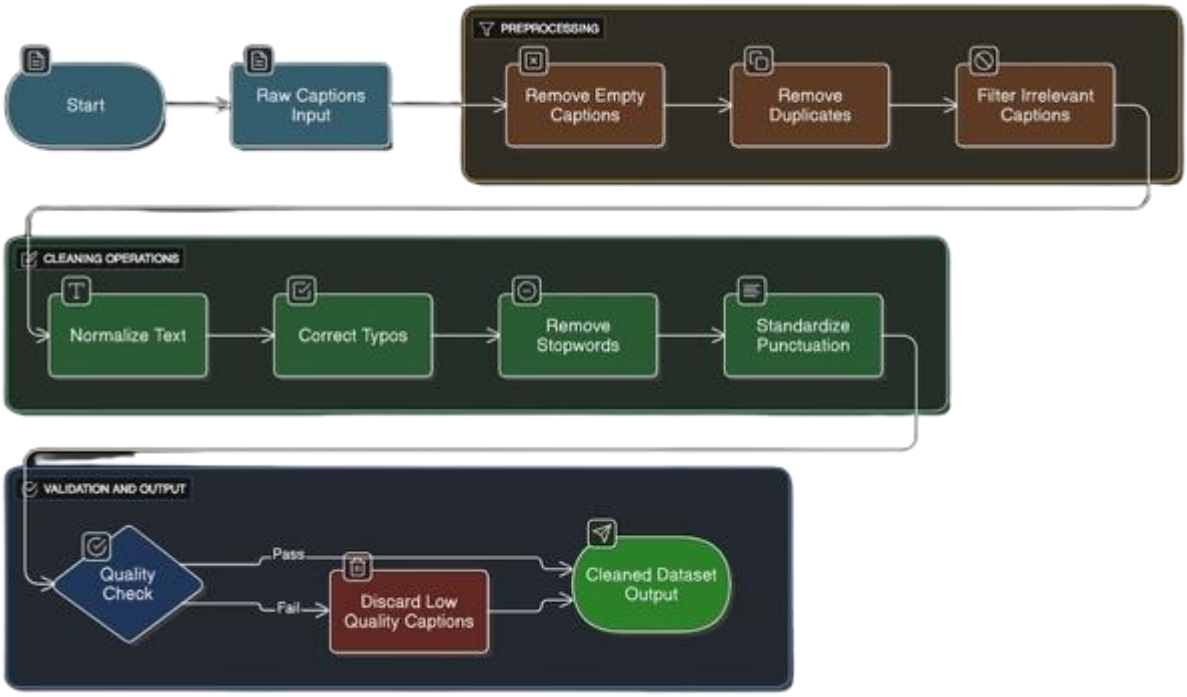


Figure IV.1: Caption Dataset Cleaning Pipeline

The Arabic captions were processed to correct errors and standardise the format across the dataset. The following steps were taken to ensure that:

Stage	Operation	Purpose
Preprocessing	Remove Empty Captions	We eliminated captions with no content
	Remove Duplicates	Repeated captions were rephrased\removed
	Filter Irrelevant Captions	Unrelated captions to the image context were excluded

Cleaning Operations	Normalise Text	Standardise text format
	Correct Typos	Fixed spelling\typographical errors
	Standardise Punctuation	Consistent punctuation usage

Table IV.2: Caption Cleaning Workflow

These steps aimed to reduce token variability and improve consistency in text processing, as a result, simplifying the data and minimising the risk of misinterpretation. Essentially, these measures contributed to improving the semantic quality of the dataset, enabling alignment between image features and their respective textual descriptions.

IV.3. Development Tools and Frameworks

This project was implemented using Python 3.11.9 and developed mostly in Visual Studio Code (VS Code) for local experimentation and script development, using a CPU (16 GB RAM, Base Speed processor: 1.80GHz) compatible TensorFlow (tensorflow_cpu==2.19.0).

For model training and evaluation, a paid Google Colab Pro was utilised to leverage its GPU (A100 GPU) resources and seamless integration with TensorFlow and Keras libraries. Our model used Keras (TensorFlow backend), along with standard libraries such as NumPy, Pandas, Matplotlib, and NLTK for natural language processing tasks. The collaborative and cloud-based environment of Colab facilitated rapid prototyping and debugging during model training.

IV.4. Model architecture

The architecture of our Arabic image captioning model followed the encoder-decoder paradigm, integrating CNNs and RNNs alongside an Arabic-specific embedding and preprocessing strategy. The model was designed to generate accurate and syntactically coherent Arabic captions for input images.

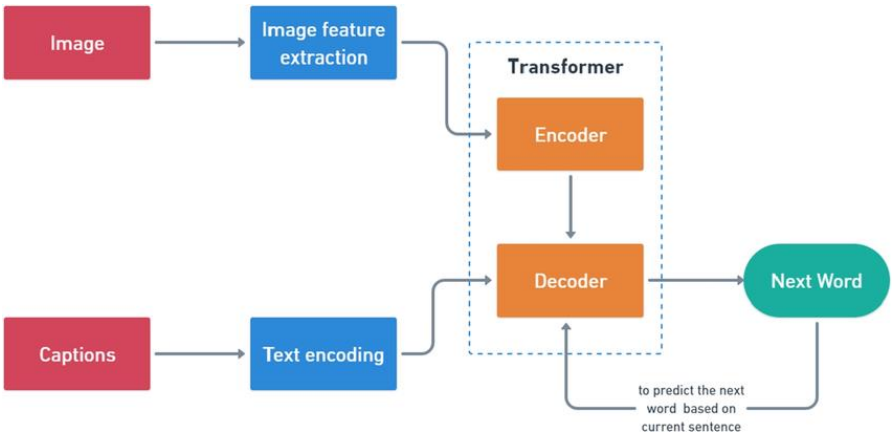


Figure IV.3: Architecture of a Simple Image Captioning Model *[57]*

As shown above in figure (name), our model was split in two blocks; the first block dedicated for image features and the second for encoded text.

- **Image Feature**

For the encoder part, we used the pre-trained VGG16 model shortened before the final classification layer. The extracted feature map from the fc2 layer (shape: 4096) is treated as a fixed-length image representation.

The dataset was resized to 224x224 pixels to match the input requirement of the encoder network. The pixel values were normalised using the ImageNet-specific preprocessing function from the library Keras (from keras.applications.vgg16 import preproces_input), which centres the pixel values based on the ImageNet training mean and std deviation.

- **Text Encoding**

For sequence modelling, we used a Long Short-Term Memory network. The LSTM architecture received two inputs: the embedded caption tokens and extracted image features. Input captions are first passed through a trainable embedding layer, followed by an LSTM. In parallel, the image features are projected through a dense layer.

The outputs of both branches are then concatenated and passed through a fully connected layer to predict the next word in the sequence.

- **Embedding Layers**

Given the complexity and morphology of the Arabic language, effective text representation is crucial. In the basic version of our model, we used trainable word embeddings initialised randomly and updated during training. Each Arabic token was appended to a 256-dimensional vector. To represent Arabic captions, we used a trainable embedding layer of 256 dimensions. Each token in the caption sequence was mapped to a vector, which was learnt jointly with the model parameters.

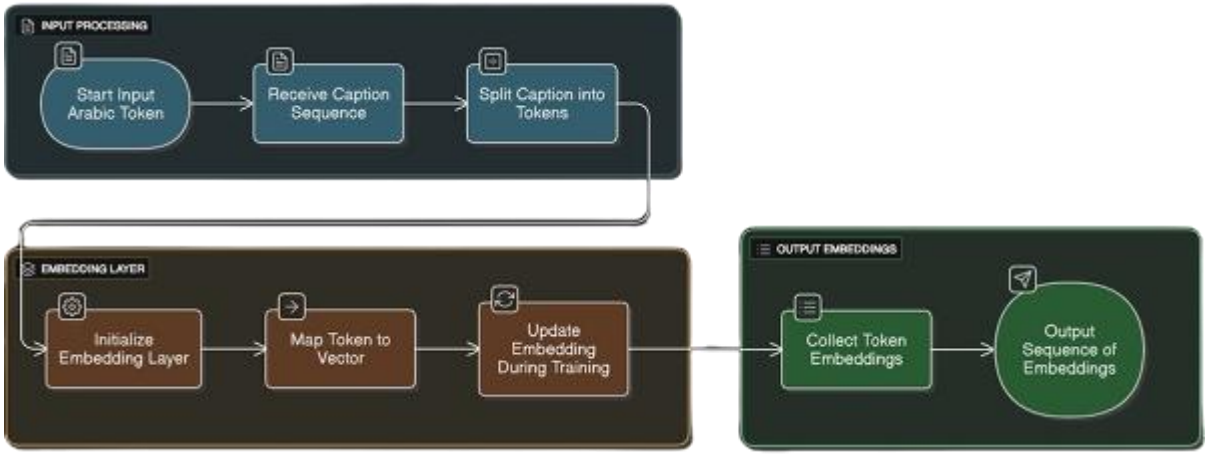


Figure IV.4: Embedding Layers for Arabic Description

IV.5. Training and Validation

This section outlines the training methodology used to optimise our Arabic image captioning model, including the dataset partitioning, model configuration, loss function, evaluation metrics, and monitoring training behaviour.

- **Loss Function**

The loss function governs how well the model predictions match the expected output. For this sequence generation task, the categorical cross-entropy was used as a loss function. Categorical cross-entropy was particularly suited for classification tasks where the target output is a one-hot encoded vector, as is the case in image captioning.

$$Loss = -\sum_{i=1}^{V} y_i \log(\hat{y}_i)$$

Equation IV.1: Mathematical Expression for the Categorical Cross-Entropy Loss Function *[58]*

- **Optimiser and Learning Rate**

Training was carried out using the Adam optimiser [59], chosen for its adaptive learning rate and convergence efficiency.

- **Initial learning rate:** learning_rate = 0.00003; controls how much the model updates its weight with each step. Smaller values mean slower, more stable learning.
- **Gradient clipping:** clipnorm = 1.0; prevents the gradients from getting too large, by limiting their total size to 1 — useful for RNNs like LSTM.

Adam is preferred in DL tasks involving large datasets and variable-length sequences due to its robustness in handling sparse gradients and noisy updates.

- **Evaluation Metrics**

To assess the quality of generated Arabic captions, both automatic and linguistic metrics were used:

Metric	Purpose
Bleu (1 and 2)	Measures n-gram overlap between candidate and reference captions

Table IV.3: overview of Automatic Evaluation Metric BLEUs Used in Arabic Image Captioning

These metrics collectively evaluate grammatical correctness, lexical accuracy, and semantic similarity.

- **Training Phases and Epochs**

A complete pass through an entire selected dataset by learning algorithm is called an epoch. It controls the times a model sees the dataset while training. [60]

In our program, the LSTM model we built was trained in three distinct phases, each involving different dataset sizes and epoch configuration. A batch size of 64 was used for the training process to balance computational efficiency and convergence stability.

- **In the first iteration**, a dataset of 3843 size was utilised, with the model trained twice, 15 epochs followed by 20 epochs.

- **The second phase**, the dataset size was dropped to 3074, intentionally to observe the model's learning behaviour over fast iteration cycles with extended epochs. The model was trained over 10, 20, 30, 40, and 50 epochs.
- **In the third and last phase**, we utilised a larger dataset size, going up to 3843 samples, with training conducted for 20 and 40 epochs.

Across all three phase, performance and validation performance consistently peaked around the 3rd epoch, specially validation accuracy reached its maximum values, stabilised up to the 8th epoch (<20 epochs), then the model's training performance declined beyond this point, indicating early overfitting.

IV.6. Model Execution and Results

This section details the execution of each phase in the image captioning pipeline, accompanied by sample code and the corresponding outputs. The model was divided into six phases:

IV.6.1. Phase1: Image Feature Extraction

As mentioned earlier, we used a VGG16 model in order to extract features from the Flickr8K images. The model successfully processed 8091 images from the Flickr8K dataset. Each image was resized to 224x244 pixels, passed through the model to extract 4096-dimensional feature vectors from the fc2 layer, which served as the fixed-length representation for caption generation.

Below are samples of the code used to display and inspect image feature extraction process:

```
# Show info about the smallest images
if min_height_file:
    print(f"Image with the Minimum Height: {min_height} - {min_height_file}")
    Image.open(min_height_file).show()

if min_width_file:
    print(f"Image with the Minimum Width: {min_width} - {min_width_file}")
    Image.open(min_width_file).show()

return features
```

Output Sample:

Image with the Minimum Height: 127 – 456512643_0aac2fa9ce.jpg

Image with the Minimum Width: 164 – 3039209547_81cc93fbec.jpg

```
# Print sample shapes
print(" Sample feature shapes:")
for k in list(features_check.keys())[:5]:
    print(k, features_check[k].shape)
```

Sample feature shapes:

1000268201_693b08cb0e (4096,)

1001773457_577c3a7d70 (4096,)

1002674143_1b742ab4b8 (4096,)

1003163366_44323f5815 (4096,)

1007129816_e794419615 (4096,)

```
print(f"Total features extracted: {len(features_check)}")
```

Total features extracted: 8091 (Flickr8K contain exactly 8091)

This confirms that the model successfully encoded all 8091 images into fixed-length feature vectors for future use in the decoder model.

IV.6.2. Phase2: Caption Loading and Analysis

In this phase, the goal was to load Arabic captions, preprocess them, and associate each description with its designated image. It was crucial for both training the model and understanding the linguistic structure.

- **Loading and Structuring Captions**

The captions file consisted of lines in the format: Image_id.jpg<TAB>caption in Arabic. Each caption was wrapped in <startseq> and <endseq> tokens to define sequence boundaries for the LSTM decoder.

```
def load_descriptions(filename):
    #open the captions file
```

```
file = open(filename, 'r', encoding='utf-8')


#read the entire content of the file

doc = file.read()


#create an empty dictionary to store image ID to captions mapping

mapping = {}


#go through each line in the file

for line in doc.strip().split('\n'):


    #each line is expected to have two parts: image filename and caption

    tokens = line.split('\t')


    #skip lines that don't match expected format

    if len(tokens) != 2:

        continue


    #split filename and caption

    image_id, caption = tokens


    #remove the file extension (.jpg, .jpeg...) to get the image ID

    image_id = image_id.split('.')[0]
```

```
        #add start and end tags to the captions
        caption = f'startseq {caption} endseq'


        #if the image ID is not already in the dictionary, initialise with an empty list
        if image_id not in mapping:
            mapping[image_id] = []


            #append the caption to the list for this image ID
        mapping[image_id].append(caption)


        #return the final dictionary: {image_id: [caption1, caption2,...]}
    return mapping
```

Output Sample:

103106960_e8a41d64f8:

طفل يرتدي سترة حمراء يلعب هوكي الشارع ويحرس المرمى.

طفل صغير يلعب دور حارس المرمى في حلبة الهوكي.

- **Visualisation of Captions with Images**

To verify the relativity between text and image data, we visualised a few samples. Arabic reshaping and bidirectional correction were applied for proper display in Matplotlib.

```
# Display 4 example images with captions
subset = {k: descriptions[k] for k in list(descriptions)[:4]}
visualize_images_with_captions(subset)
```

Results:

فتاة تدخل إلى مبنى خشبي.
فتاة صغيرة تتسلق إلى بيت اللعب الخشبي.
فتاة صغيرة تصعد الدرج إلى بيت اللعب الخاص بها.
فتاة صغيرة ترتدي فستانا ورديا تذهب إلى كوخ خشبي.

كلب أسود وكلب مرقط يتقاتلان
كلب أسود وكلب ثلاثي الألوان يلعبان مع بعضهما البعض على الطريق.
كلب أسود وكلب أبيض ببقع بنية ينظران إلى بعضهما البعض في الشارع.
كلبين من سلالات مختلفة ينظران إلى بعضهما البعض على الطريق.
كلبين على الرصيف يتحركان نحو بعضهما البعض.

فتاة صغيرة مغطاة بالطلاء تجلس أمام قوس قزح مرسوم بيديها في وعاء.
فتاة صغيرة تجلس أمام قوس قزح كبير مطلي.
فتاة صغيرة تلعب بألوان الأصابع على العشب أمام قماش أبيض عليه قوس قزح.
هناك فتاة ذات ضفائر تجلس أمام لوحة قوس قزح.
فتاة صغيرة ذات ضفائر ترسم في الخارج على العشب.

رجل مستلق على مقعد بينما يجلس كلبه بجانبه.
رجل مستلق على مقعد مربوط به أيضا كلب أبيض.
رجل نائم على مقعد بالخارج مع كلب أبيض وأسود يجلس بجانبه.
رجل عاري الصدر يرقد على مقعد في الحديقة مع كلبه.
رجل مستلق على مقعد ممسكا بسلسلة كلب يجلس على الأرض

Figure IV.5: Visualised Samples from Text and Image Data

We conducted a DataFrame and computed statistics such as caption length, vocabulary size, and word frequency.

- **Basic Statistics:**

```
# How many rows were loaded?
print(f"Total captions loaded: {len(data)}")


# How many unique image files?
print(f"Unique image files: {data['file'].nunique()}")
```

Output:

Total Caption Loaded: 40455

Unique Image Files: 8091

- **Maximum Caption Length :**

```
#calculate the maximum length of any caption
max_caption_length = calc_max_length(all_captions)
print(f"Maximum caption length (in words): {max_caption_length}")
```

Output:

Maximum caption length (in words): 31

```
#calculate the total number of words in all captions
total_words = data['caption'].apply(lambda x: len(x.split())).sum()
print(f"Total number of words: {total_words}")
```

Output:

Total number of words: 336394

```
# Total number of characters in all captions
total_chars = data['caption'].apply(lambda x: len(x)).sum()
print(f"Total number of characters: {total_chars}")
```

Output:

total number of characters: 1851771

```
#function to build the vocabulary frequency
def df_word(df_txt):
    vocabulary = []

#loop over each caption in the data frame
    for i in range(len(df_txt)):
        temp = df_txt.iloc[i, 2]  # Assumes captions are in column index 2
        vocabulary.extend(temp.split()) #split the caption into words and add them to the
vocabulary list

    print('Vocabulary Size:', len(set(vocabulary))) #print the size of the vocabulary

    ct = Counter(vocabulary) #count the frequency of each word in the vocabulary

    #create a data frame with the word and its frequency
    dfword = pd.DataFrame({
        "word": list(ct.keys()), #list of unique words
        "count": list(ct.values()) #frequency of each word
    })

#sort the data frame by frequency in descending order
    dfword = dfword.sort_values("count", ascending=False)
```

```
    #reset the index of the data frame
    dfword = dfword.reset_index(drop=True)
    return dfword
```

Output:

Vocabulary size: 19540

	Count	Word
0	15461	في
1	13933	على
2	7065	من
3	6541	كلب
4	6321	رجل

Table IV.4: Five Most Used Words in the Arabic Captions

```
# Plot the top 50 most frequent words
topn = 50

#function to plot the top n most frequent words
def plthist(dfsub, title="Top 50 Most Frequent Words"):
    plt.figure(figsize=(20, 3)) #set the figure size
    plt.bar(dfsub.index, dfsub["count"]) #plot the bar chart
    plt.yticks(fontsize=16) #set the y-axis tick size
    plt.xticks(dfsub.index, dfsub["word"], rotation=90, fontsize=14) #set the x-axis tick
size and rotate the labels
    plt.title(title, fontsize=18) #set the title of the plot
    plt.tight_layout() #adjust the layout so that the labels fit within the figure
    plt.show() #display the plot
#plot the top 50 most frequent words
plthist(dfword.iloc[:topn], title="Top 50 Most Frequent Words in Captions")
```

Output:

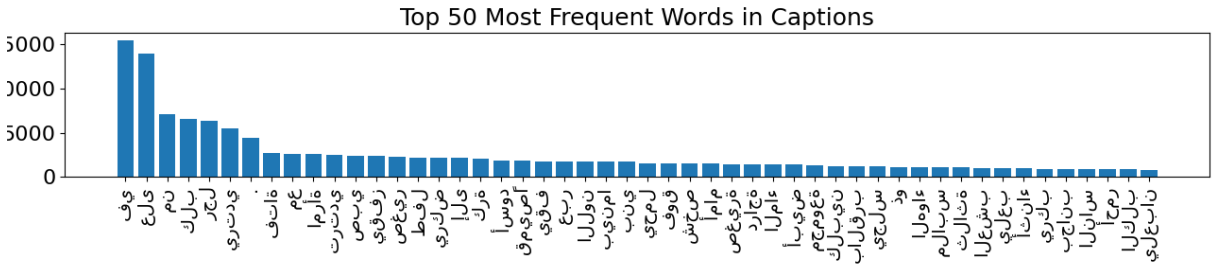


Figure IV.6: Top 50 Most Used Words in Arabic Captions

This stage provided a cleaned, structured corpus of image-caption pairs and crucial statistics for preparing the tokenizer and training pipeline.

IV.6.3. Phase3: Linguistic Cleaning and Caption Normalisation

Arabic, as a morphologically rich language, introduces additional preprocessing challenges for natural language processing. This particular phase centres on normalising and cleaning the caption text to reduce redundancy, handle orthographic variants, and improve token consistency before model training.

- **Text Cleaning Utilities**

Several Arabic-specific and general NLP utilities were used for preprocessing:

Cleaning Step	Command Used	Description	Purpose
Diacritic Removal	strip_diacritics strip_tashkee	Remove all ḥarakāt	Reduce token variability and improve consistency in text processing
Tatweel and Punctuation Removal	strip_tatweel	Eliminate elongation characters (-) and unnecessary punctuations	Simplify the captions and avoid misinterpretation during tokenization
Duplicate Removal	Done manually	Delete identical captions	Prevent overfiting and ensure diverse training examples

Incomplete or Malformed captions	Done manually	Removed very short, grammatically incorrect, or nonsensical captions using manual review	Improve the semantic quality of the training data

Table IV.5: Caption Cleaning Steps Applied to the Arabic Dataset

```
# Standard libraries
import pandas as pd #data manipulation and analysis
import string #for ponctuations and characters
import re #text pattern matching
import nltk #natural language processing

# Arabic libraries
from num2words import num2words
import arabic_reshaper #reshaping arabic text for better display
from pyarabic.araby import (
    strip_diacritics, #remove arabic diacritics
    strip_tashkeel, #remove tashkeel
    strip_tatweel, #remove tatweel
)
from bidi.algorithm import get_display
```

- **Loading and cleaning the dataset:**

```
# -------- Load captions --------
data_path = r'C:\master\Arabic Image Captioning\captions\captions.txt' #path to the captions file
data = pd.read_csv(data_path, sep='\t', names=['image_id', 'caption', 'extra'], encoding='utf-8') #load captions
print("Missing captions:", data['caption'].isna().sum()) #check for missing captions
data['caption'] = data['caption'].fillna('') #replace missing captions with an empty string
```

Output:

Missing Captions: 0

- **Clean all captions:**

```
cleaned_captions = [] #list to store cleaned captions
caption_lines = [] #list to store captions lines

for i, row in data.iterrows(): #loop through each row in the captions file
    image_id = row['image_id'].replace('.jpg', '').strip() #get the image ID
    caption = row['caption'] #get the caption
    cleaned = clean_text(caption) #clean the caption
    cleaned_captions.append(cleaned) #append the cleaned caption to the list
    caption_lines.append(f"{image_id} {cleaned}") #format the image ID and caption
as a line
```

Example:

Before: ألمَسْجِدٌ الأَقْصَى مِنَ المَسَاجِدِ الثَّلاَثَةُ الَّتِي لَهَا مَكَانَةٌ عَظِيمَةٌ فِي الإِسْلاَمِ.

After: مسجد اقصى من مساجد ثلاثة تي لها مكانة عظيمة في اسلام

Aspect Improved	Explanation
Tokenization accuracy	Cleaning removed noisy characters and standardized forms, allowing tokenizers to segment words more accurately.
Vocabulary consistency	Normalization reduced variations of the same word (e.g., "ئ" "ؤ" "ء" "إ" "آ" "ا" "أ"), which helped to reduce vocabulary size.
Model convergence	With cleaner, the model was able to learn mappings between images and captions more efficiently, resulting in faster convergence.

Table IV.6: Impact of Caption Cleaning on Model Performance

- **Vocabulary Analysis**

```
# ---------- Vocabulary Report ----------
```

```
clean_vocabulary = set() # Create a set to store unique words in the vocabulary
for txt in cleaned_captions:
    clean_vocabulary.update(txt.split()) # Add each word in the caption to the vocabulary set
print('Clean Vocabulary Size:', len(clean_vocabulary)) # Print the size of the vocabulary
```

Output:

Clean vocabulary size: 12762

- **Baseline Dataset comparison:**

Aspect	ElJundi's Baseline	Our Model
Dataset	Flickr8k-arabic	Flickr8k-Arabic
Captions per image	3	5
Language Cleaning	Profound	Advanced

Table IV.7: Baseline Comparison between our Dataset and Obeida ElJundi's

IV.6.4. Phase4: Memory-Efficient Data Loading and Preparation for Training

This phase was designated for loading only the required subset of data — image features and cleaned captions, into memory for the model training. The complete dataset was filtered down to only the image IDs used for training.

- **Loading Training Image IDs**

The training image IDs were obtained from a predefined split file, where each line contains an image filename. This emsure the model is trained only on a consistent subset.

```
def load_image_ids(filename):
    """Load image IDs from the train/test split file (e.g., Flickr_8k.trainImages.txt)"""
    with open(filename, 'r', encoding='utf-8') as file:
        lines = file.read().strip().split('\n')
    return set(line.split('.')[0] for line in lines if line.strip())

filetrain = r'C:\master\Arabic Image Captioning\captions\training data\train_Images.txt'
# Step 1: Load training image IDs
```

```
train_ids = load_image_ids(filetrain)
print(f'[Train] Number of image IDs: {len(train_ids)}')
```

Output:

[Train] Number of image IDs: 3646

- **Loading Image Features**

Image features extracted using VGG16 (look up phase 1) were saved in a pickle file, then were loaded into memory for future use.

```
def load_selected_photo_features(features_path, image_ids):
    """Load VGG16 image features for a specific subset of image IDs"""
    all_features = pickle.load(open(features_path, 'rb'))
    selected_features = {img_id: all_features[img_id] for img_id in image_ids if
img_id in all_features}
    return selected_features

features_path = r'C:\master\feature_extraction.pkl'
# Step 2: Load image features for training
train_features = load_selected_photo_features(features_path, train_ids)
print(f'[Train] Loaded features for: {len(train_features)} images')
```

Output:

[Train] Loaded features for: 4994 images

- **Loading Cleaned Captions**

This step loads the cleaned captions created in phase 3, and filters then down to only those relevant for the training.

```
def load_cleaned_captions(cleaned_caption_path, image_ids):
    """Load cleaned captions for selected image IDs."""
    descriptions = {}
    with open(cleaned_caption_path, 'r', encoding='utf-8') as file:
        for line in file:
            tokens = line.strip().split('\t')
            if len(tokens) < 2:
```

```
            continue
        image_id, caption = tokens[0].split('.')[0], tokens[1]
        if image_id in image_ids:
            if image_id not in descriptions:
                descriptions[image_id] = []
            descriptions[image_id].append(f'startseq {caption} endseq')
    return descriptions

cleaned_caption_path = r'C:\master\Arabic Image
Captioning\captions\cleaned_captions.txt'
train_descriptions = load_cleaned_captions(cleaned_caption_path, train_ids)
print(f'[Train] Loaded descriptions for: {len(train_descriptions)} images')
```

Output:

[Train] Loaded descriptions for: 3646 images

- **Verifying Data Integrity**

Checking for any image IDs without captions ensures consistency in downstream training steps.

```
missing_ids = train_ids - train_descriptions.keys()
print(f'Missing descriptions for: {len(missing_ids)} images')
print("IDs:", missing_ids)
```

Output:

Missing descriptions for: 0 images

IDs: none

This phase ensured only relevant data was loaded into memory—reducing resources usage while maintaining alignment between features and descriptions. The filtered datasets were now ready for tokenisation and model input formatting.

IV.6.5. Phase 5: Model Construction and Training

This phase details the implementation of the Arabic image captioning model's pipeline. Following the typical encoder-decoder pattern, a pre-trained CNN encodes the

visual data, and an LSTM encoder generates the corresponding Arabic caption word-by-word. The model was built and trained using the deep learning API library Keras (TensorFlow backend).

- **Libraries and Environment**

```
import numpy as np
import matplotlib.pyplot as plt
import pickle
import keras
import tensorflow as tf
from keras.applications.mobilenet_v2 import preprocess_input
from keras.models import Model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout, concatenate
from keras_preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint
from nltk.tokenize import word_tokenize
import re
import pyarabic.araby as ar

print(keras.__version__)
```

Output:

Environment: TensorFlow backend, Keras API, NLTK for tokenisation, PyArabic for handling Arabic-specific text patterns.

- **Loading and Verifying Data**

```
# Load VGG16 image features
features_path = r'C:\master\feature_extraction.pkl'
with open(features_path, 'rb') as f:
    raw_features = pickle.load(f)
    train_features = {k: np.squeeze(v) for k, v in raw_features.items()}
def load_captions_from_file(file_path):
    captions = dict()
    with open(file_path, 'r', encoding='utf-8') as file:
```

```
        for line in file:
            line = line.strip()
            if not line:
                continue
            parts = re.split(r'\s+', line.strip(), maxsplit=1)
            if len(parts) != 2:
                print(f"Skipping malformed line: {line}")
                continue
            key, caption = parts
            image_id = key.split('#')[0]
            captions.setdefault(image_id, []).append(caption)
    return captions

# Remove .jpg from keys so they match feature keys
captions = {k.replace('.jpg', ''): v for k, v in captions.items()}

# Check that keys in captions match feature keys
sample_caption_keys = list(captions.keys())[:5]
print("Sample caption keys:", sample_caption_keys)
print("Sample feature keys:", list(train_features.keys())[:5])

# Sanity check: find captions that don't have corresponding image features
missing = [k for k in captions if k not in train_features]
print(f"Missing image features for {len(missing)} images. Example: {missing[:5]}")
```

Output:

Sample caption key: [‘1000268201_693b08cb0e’, ‘1001773457_577c3a7d70’, ‘1002674143_1b742ab4b8’, ‘1003163366_44323f5815’, ‘1007129816_e794419615’]

Sample feature keys: [‘1000268201_693b08cb0e’, ‘1001773457_577c3a7d70’, ‘1002674143_1b742ab4b8’, ‘1003163366_44323f5815’, ‘1007129816_e794419615’]

Missing image features for 0 images. Example []

- **Tokenisation**

Arabic captions were tokenised using NLTK, and a simple custom tokenizer was created:

```
# Function to create a tokenizer using NLTK
def create_tokenizer(lines):
    all_tokens = []
    for line in lines:
        tokens = word_tokenize(line)  # Tokenize using nltk
        all_tokens.extend(tokens)

    # Build vocabulary: Create a dictionary of word frequency counts
    vocabulary = set(all_tokens)  # Set removes duplicates
    word_index = {word: i+1 for i, word in enumerate(vocabulary)}  # Assign each
word an index (starting from 1)

    # Create a tokenizer-like dictionary
    tokenizer = {
        'word_index': word_index,
        'index_word': {i: word for word, i in word_index.items()},
        'vocabulary_size': len(vocabulary)
    }

    return tokenizer

# tokenize captions
tokenizer = create_tokenizer(captions_list)

if tokenizer['word_index']:
    print("Tokenizer created successfully!")
else:
    print("Tokenizer not created.")

# Save the extracted features in the required directory:
token_directory = 'C:\master'
```

```
# Save tokenizer to a file
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)

#the size of the vocabulary is
vocab_size = len(tokenizer['word_index']) + 1
print('Vocabulary Size: %d' % vocab_size)
```

Output:

Tokenizer created successfully!

Vocabulary size: 10902

- **Maximum Caption Length**

```
#the legth of the descriptions with the most frequent words
def max_length(captions_list):
    return max(len(d.split()) for d in captions_list)

max_length = max_length(captions_list)
  # In this case the maximum caption's length is 24.
print("Max Caption Length: ", max_length)
```

Output:

Max Caption Length: 29

- **Model Architecture**

The model combines:

Image feature encoder: Dense layer to project 4096-D image vector into the latent space

Caption decoder: Embedding → LSTM → Dense with softmax for next-word prediction

```
from keras.layers import Input, Dense, LSTM, Embedding, Dropout, concatenate
from keras.models import Model
```

```
#build the deep learning model RNN-LSTM
def define_model(vocab_size, max_length):
    inputs1 = Input(shape=(4096,))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)

    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)

    decoder1 = concatenate([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)

    outputs = Dense(vocab_size, activation='softmax')(decoder2)

    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

    print(model.summary())
    return model
model = define_model(vocab_size, max_length)
```

Output:

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 29)	0	-
input_layer (InputLayer)	(None, 4096)	0	-
embedding (Embedding)	(None, 29, 256)	2,790,912	input_layer_1[0][0]
dropout (Dropout)	(None, 4096)	0	input_layer[0][0]
dropout_1 (Dropout)	(None, 29, 256)	0	embedding[0][0]
not_equal (NotEqual)	(None, 29)	0	input_layer_1[0][0]
dense (Dense)	(None, 256)	1,048,832	dropout[0][0]
lstm (LSTM)	(None, 256)	525,312	dropout_1[0][0], not_equal[0][0]
concatenate (Concatenate)	(None, 512)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	131,328	concatenate[0][0]
dense_2 (Dense)	(None, 10902)	2,801,814	dense_1[0][0]

Figure IV.7: A Sample from the Model's Output (Model's Summary)

- **Data Sequencing and Generator**

1. A data generator was defined to:
2. Create sequences: <image features, partial caption>→next word
3. Yield batched to the model

```
#create sequance of images, descriptions and next description
def create_sequence(tokenizer, max_length, desc_list, photo):
    print("Photo shape in create_sequence:", photo.shape)

    X1, X2, y = list(), list(), list()
    for desc in desc_list:
        tokens = word_tokenize(desc)
        seq = [tokenizer['word_index'][word] for word in tokens if word in
tokenizer['word_index']]
        if len(seq) < 2:
            print(f"Too short: {desc}")
            continue
        for i in range(1, len(seq)):
            in_seq, out_seq = seq[:i], seq[i]
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
```

```
            out_seq = to_categorical([out_seq], num_classes=len(tokenizer['word_index'])
+ 1)[0]
            X1.append(photo)
            X2.append(in_seq)
            y.append(out_seq)

    return array(X1), array(X2), array(y)
#data generator
def data_generator(descriptions, photos, tokenizer, max_length):
    for key, desc_list in descriptions.items():
        if key not in photos:
            continue
        photo = np.squeeze(photos[key])  # convert (1, 4096) -> (4096,)
        in_img, in_seq, out_word = create_sequence(tokenizer, max_length, desc_list,
photo)
        if len(in_img) == 0:
            continue
        yield (in_img, in_seq), out_word
```

Output:

Photo shape in created_sequence: (4096,)

- **Training Configuration and Experimental Phases**

As mentioned earlier in **Training Phases and Epochs,** to evaluate the model's performance under different dataset sizes and training regimes, three distinct phases were executed.

Below are the results extracted from each phase:

- **First Iteration:**

The purpose of this phase was baseline learning curve observation.

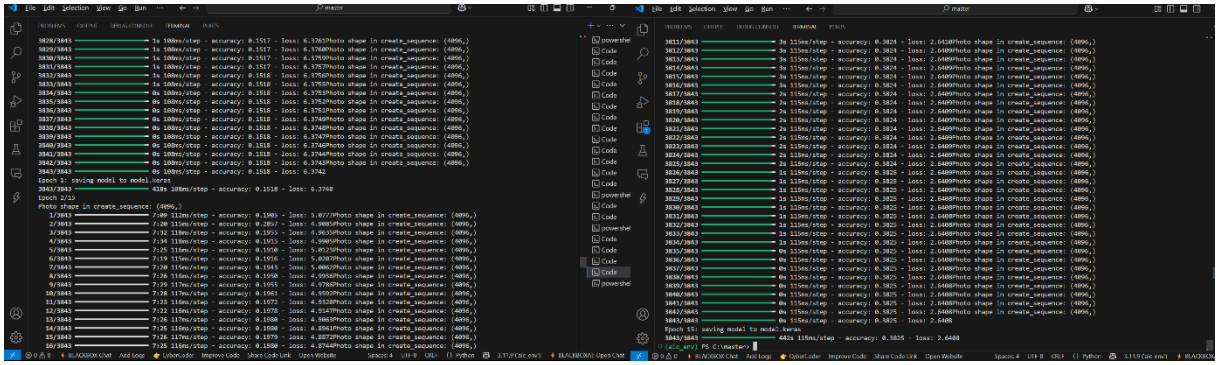


Figure IV.8: Samples from the First and Last Epoch Iteration (15 Epochs)

- **Second Iteration**

The purpose of this phase was examining the behaviour of the model under smaller data and longer training.

- **Results of the First Training (10 epochs):**

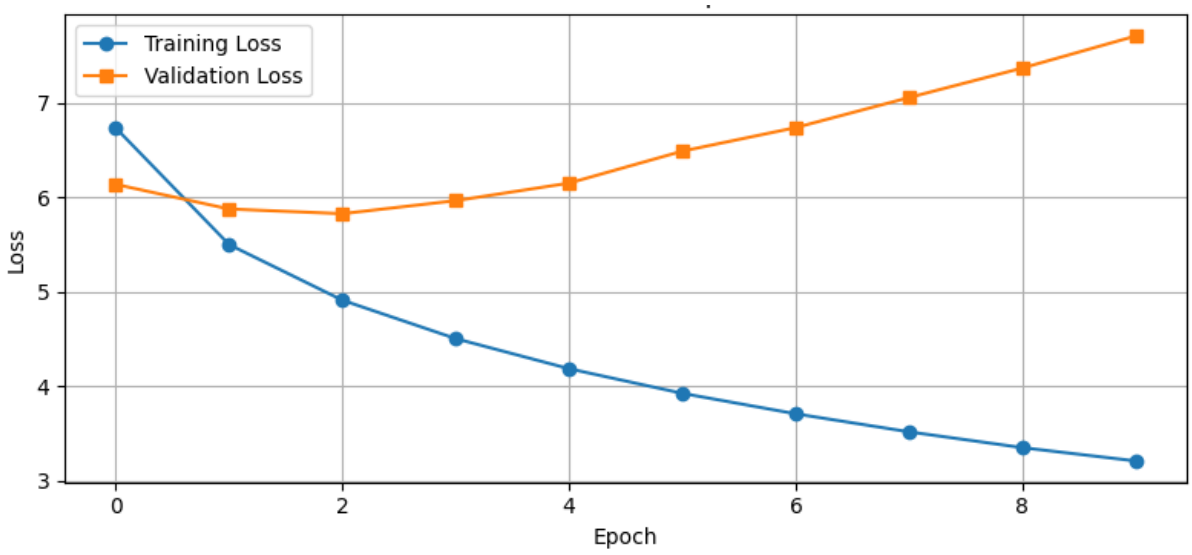


Figure IV.9: Loss Curve Over 10 Epochs

Results of the Second Training (20 epochs):

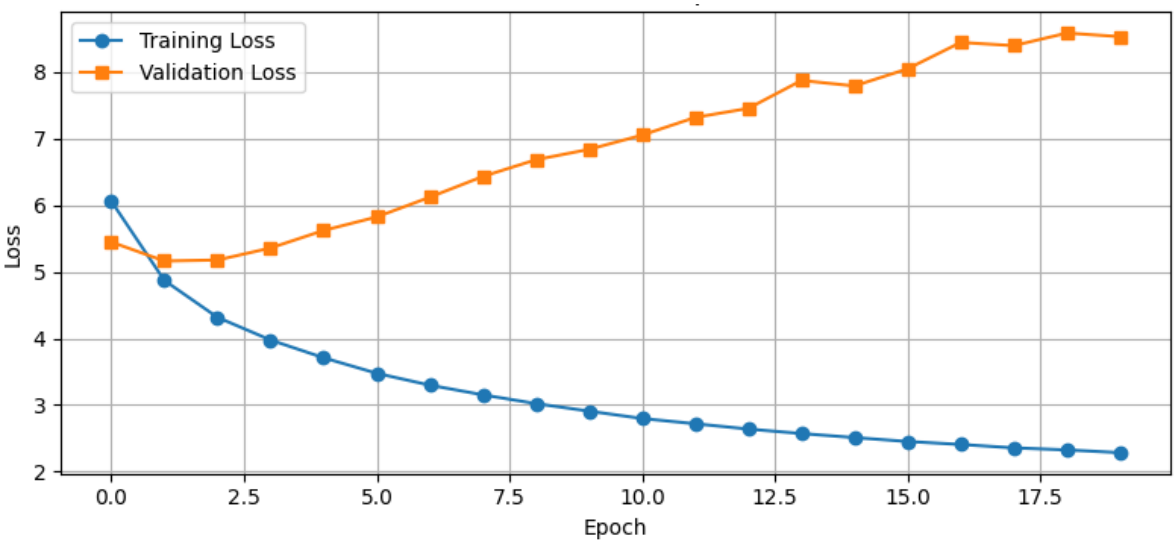


Figure IV.10: Loss Curve Over 20 Epochs

- **Results of the Third Training (30 Epochs):**

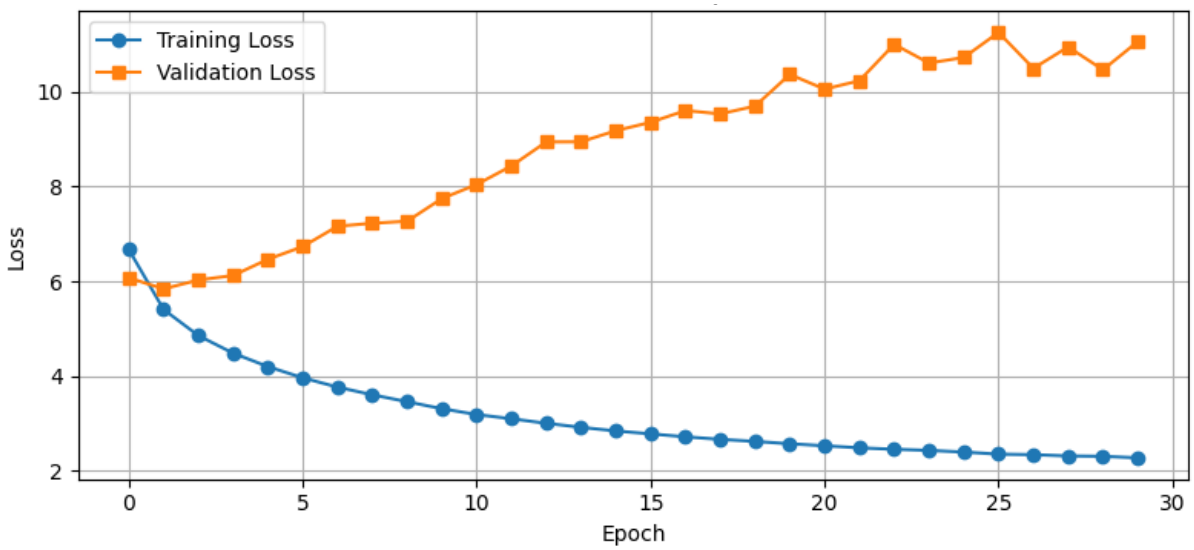


Figure IV.11: Loss Curve Over 30 Epochs

- **Results of the Fourth Training (40 Epochs):**

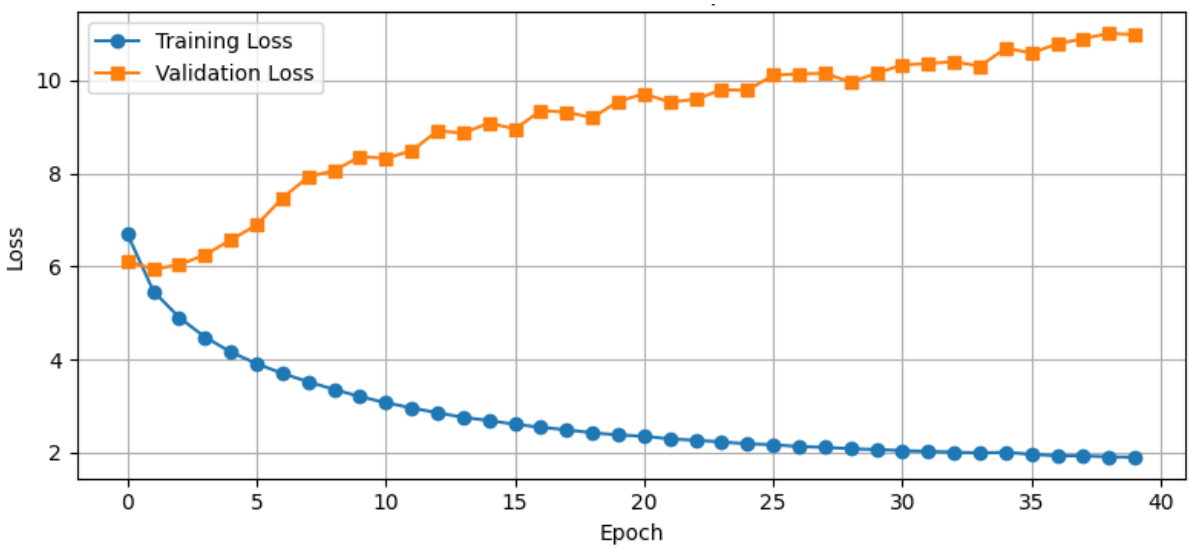


Figure IV.12: Loss Curve Over 40 Epochs

- **Results of the Fifth and Last Training (50 Epochs):**

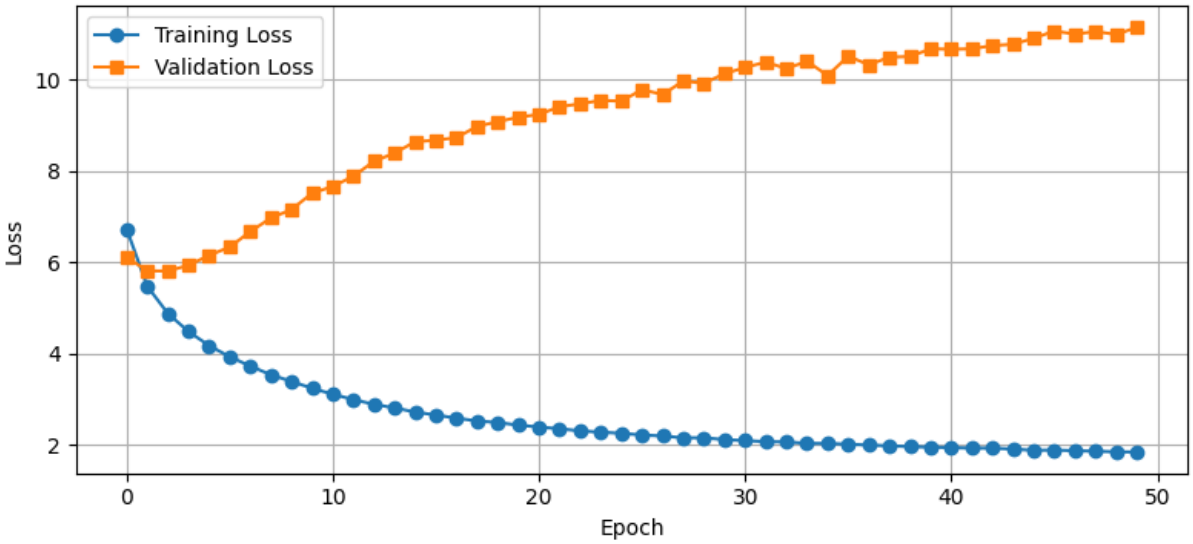


Figure IV.13: Loss Curve Over 50 Epochs

- **Third Iteration**

The purpose of this phase was validating learning on larger data, test generalisation.

- **Results of the First Training (20 Epochs):**

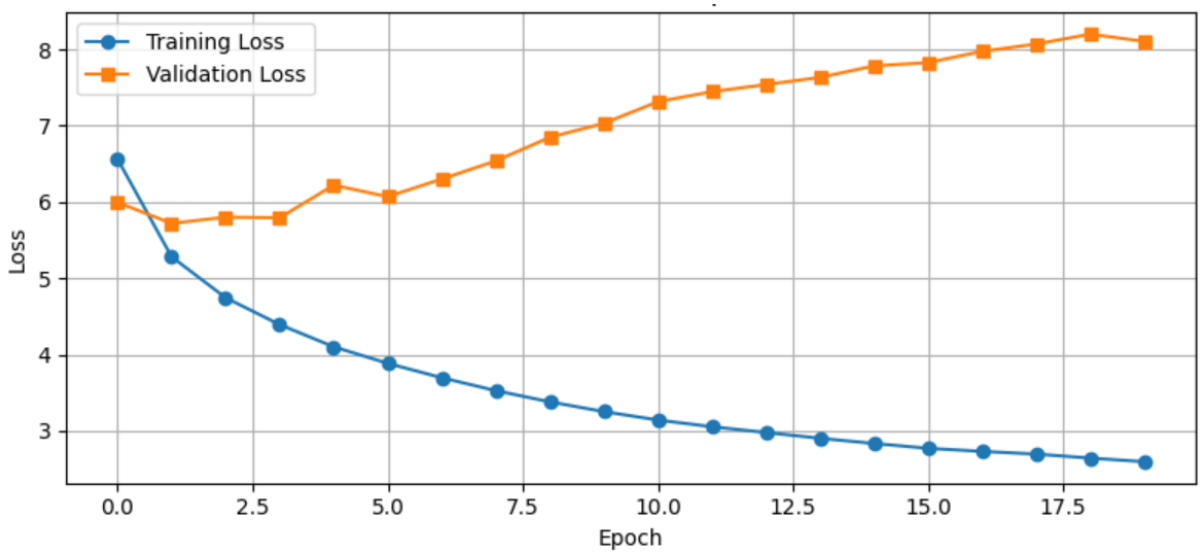


Figure IV.14: Loss Curve Over 20 Epochs

- **Results of the Second Training (40 Epochs):**

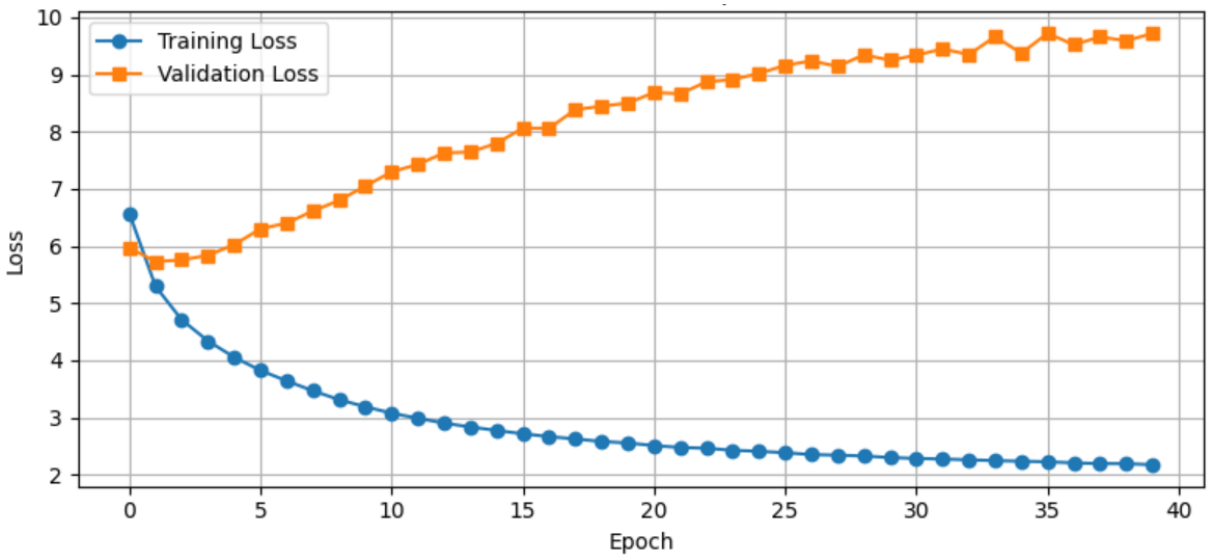


Figure IV.15: Loss Curve Over 40 Epochs

- **Baseline Model Architecture Comparison**

Aspect	ElJundi's Model	Our Model
Encoder	VGG16	VGG16
Decoder	Embedding→LSTM→Dense	Embedding→LSTM→Dense + Dropout regularisation
Special Layer	None	Dropout (0.5)
Feature Dim	4096	4096

Caption Embedding	Standard	Tuned to 256-dim
Loss	Categorical Cross entropy	Categorical Cross entropy
Optimiser	Adam	Adam

Table IV.8: Baseline Comparison between the Two Models' Architecture

IV.6.6. Phase 6: Caption Generation and Evaluation

In the final phase, the trained model was tested on unseen images. The objective was to generate natural, grammatically accurate Arabic descriptions and evaluate their quality using both automatic and qualitative methods.

- **Generated captions**

To improve variation and avoid repetitive phrases, search beam *[61]* was used. Beam search is a widely-used decoding strategy for sequence generation tasks like translation and captioning. Instead of picking the single best word at each step (like greedy), it keeps multiple candidate sequences and expands then in parallel.

This balances:

- Higher fluency.
- Better global context.
- Fewer nonsense phrases than pure sampling.

- **Snippets from the Model:**

```
# === Updated caption generator with beam search ===

def generate_desc_beam(model, tokenizer, photo, max_length, beam_size=3):
    word_index = tokenizer.get('word_index', {})
    index_word = tokenizer.get('index_word', {})

    # --- Ensure startseq and endseq exist ---
    if 'startseq' not in word_index:
        print("[WARNING] Adding fake 'startseq' -> 1")
        word_index['startseq'] = 1
        index_word[1] = 'startseq'
    if 'endseq' not in word_index:
        print("[WARNING] Adding fake 'endseq' -> 2")
        word_index['endseq'] = 2
        index_word[2] = 'endseq'

    start_index = word_index['startseq']
    end_index = word_index['endseq']

    sequences = [[[start_index], 0.0]]  # (sequence, score)
```

```
    for _ in range(max_length):
        all_candidates = []
        for seq, score in sequences:
            if seq[-1] == end_index:
                all_candidates.append([seq, score])
                continue

            sequence = pad_sequences([seq], maxlen=max_length)
            yhat = model.predict([photo.reshape((1, 4096)), sequence], verbose=0)[0]

            for idx, prob in enumerate(yhat):
                if prob > 0:
                    candidate = [seq + [idx], score - np.log(prob + 1e-8)]
                    all_candidates.append(candidate)

        ordered = sorted(all_candidates, key=lambda tup: tup[1])
        sequences = ordered[:beam_size]

    best_seq = sequences[0][0]
    final_caption = [index_word.get(idx, '') for idx in best_seq]

    if 'endseq' in final_caption:
        end_idx = final_caption.index('endseq')
        final_caption = final_caption[:end_idx+1]

    return ' '.join(final_caption)
```

- **Automatic Evaluation**

BLEU-1 and BLEU-2 scored the generated captions, for n-gram precision, with smoothing for small samples.

Code:

```
def evaluate_bleu(candidate, references):
    smoothie = SmoothingFunction().method4
    candidate = candidate.split()
    references = [ref.split() for ref in references]

    bleu1 = sentence_bleu(references, candidate, weights=(1, 0, 0, 0),
smoothing_function=smoothie)
    bleu2 = sentence_bleu(references, candidate, weights=(0.5, 0.5, 0,
0), smoothing_function=smoothie)
    return bleu1, bleu2
```

Example Output:

Generated: startseq صبي يلعب في أمواج عند غروب شمس endseq

BLEU-1: 0.1333, BLEU-2: 0.0959

- **Baseline Comparison**

To validate improvements, the results were compared with Obeida ElJundi's baseline *[54]*:

Metric	Eljundi's Work	Our Work
BLEU-1	~0.45	0.1200
BLEU-2	N/A	0.0670

Table IV.9: Baseline Evaluation Comparison between Our work and Obeida ElJundi's

- **Results from Testing Phase:**

The results are in Arabic; an image with the generated caption followed by ground truth caption.

Image	Generated Caption	Ground Truth Caption
	دراجة على على دراجة على على في في	فتاتين جالستين بالقرب من دراجتيهما على ضفة النهر
	ترتدي ترتدي يقف على محيط على محيط	طفل يلعب عند شاطئ البحر وهو مرتدي ملابسه وقت غروب الشمس
	من من من في في في ف	مهرجان المناطيد عند غروب الشمس وسط حضور الجماهير

Table IV.10: Results of the First Test

It is evident that the generated captions contain repetitive and irrelevant words compared to the ground truth captions, indicating that the model needs improvement in generating coherent and contextually accurate descriptions.

Image	Generated Caption	Ground Truth Caption
	هناك دراجتان تقفان خلف شخصين يجلسان على عشب قرب من مسطح مائي	فتاتين جالستين بالقرب من دراجتيهما على ضفة النهر
	صبي يلعب في أمواج البحر عند غروب شمس	طفل يلعب عند شاطئ البحر وهو مرتدي ملابسه وقت غروب الشمس
	تصطف سبعة بالونات كبيرة في ليل بالقرب من حشد من ناس	مهرجان المناطيد عند غروب الشمس وسط حضور الجماهير

Table IV.11: Results of the Second Test

The new outputs are closer in structure and content to the reference captions, demonstrating the positive impact of the search beam decoding strategy on generation quality.

It is important to recognise that AI models, especially in complex tasks like image captioning, are inherently imperfect and rarely deliver optimal performance on the first attempt. Developing a robust AI system is an iterative process that requires continuous experimentation, testing, and refinement based on observations and failure analysis.

IV.7. Future Improvements and Optimisations

The current image captioning model achieved low scores predicted by BLEU (1 and 2) indicating significant room for improvement. To address these shortcomings and enhance future performance.

To further enhance the quality and accuracy of the generated captions, future improvement should focus on optimising model parameters, fine-tuning on larger and diverse datasets (such as MS COCO or Flickr30K). The model relies on VGG16 model, an outdated model compared to modern CNN backbones like ResNet [62] and EfficientNet [63]. Switching to this architecture could provide richer, more robust image features, which is critical for generating meaningful captions. Transitioning from

RNN-base decoders to Transformer-beased models, such as Vision-and-Language Transformer [64] or Vision Transformers [65], can better handle long-range dependencies and improve caption fluency.

Additionally, evaluating the model with human feedback and iterative retaining can help reduce repetitive or irrelevant outputs and align the generated descriptions more closely with human-like language.

IV.7. Conclusion

Our work presented an end-to-end Arabic image captioning system based on a convolutional encoder and a recurrent decoder, trained on a manually cleaned and normalised version of the Flickr8k dataset translated into Modern Standard Arabic. The model was evaluated using standard metrics, where it demonstrated improved performance.

However, like all models, our project had limitations, particularly in handling complex scenes and generating descriptions due to vocabulary constraints. Additionally, the dependency on pre-trained models restricted the adaptability across diverse domains.

In conclusion, this project provided a concrete foundation for Arabic image captioning and displayed the practicality and value of combining classic encoder-decoder architectures with modern Arabic NLP tools.

General Conclusion

General Conclusion

The work presented in this thesis explored the development of an Arabic image captioning system by integrating concepts from image processing, artificial intelligence, and natural language processing. Through a progressive and structured approach, we examined the theoretical foundations, technical methodologies, and practical implementations necessary to tackle the challenges of automatic image description in the Arabic language.

In the first chapter, we established the importance of digital images and image processing techniques, which serve as the foundation for visual data understanding in intelligent systems. The second chapter introduced the key principles of artificial intelligence, with a particular focus on deep learning and neural networks, highlighting their crucial role in analysing and interpreting visual information. Building on this foundation, the third chapter reviewed the evolution of image captioning models, emphasizing the linguistic and technical obstacles faced when working with Arabic, such as its complex morphology, limited resources, and dialectal diversity.

The practical contributions of this work were detailed in the fourth chapter, where we implemented an end-to-end Arabic image captioning system based on a convolutional encoder and recurrent decoder architecture. The system was trained on a manually curated Arabic dataset derived from the Flickr8k corpus and demonstrated promising results, generating coherent and semantically relevant descriptions for simple visual scenes. Nonetheless, the work revealed certain limitations, particularly when dealing with complex images and domain adaptation.

Overall, this thesis contributes to addressing the gap in Arabic image captioning research by providing a concrete, functional system and by illustrating the potential of combining classical computer vision techniques with modern Arabic language models. Future work can explore architectures that are more advanced, larger and more diverse datasets, and further integration of linguistic resources to enhance the system's performance and generalization capabilities across different domains.

References

[1] M. S. K. S. C. a. M. S. Singh, “Various Image Compression Techniques: Lossy and Lossless.,” *International Journal of Computer Applications,* pp. 23-26., 2016.

[2] A. &. S. W.-Z. P, ““Encryption Algorithms for Color Images: A Brief Review of Recent Trends”,” *International Journal of Advanced Computer Science and Applications,* vol. 7, no. 10, p. 2, 2016.

[3] J. Sachs, “"Digital image basics." 1999 (1996).,” *Digital Light & Color,* 1999.

[4] A. McAndrew, An introduction to digital image processing with MATLAB, Course Technology Press, 2004.

[5] R. R. S. A. Amit Choudhary, “Off-Line Handwritten Character Recognition using Features,” *AASRI Procedia,* vol. 4, pp. 406-312, 2013.

[6] K. Elasnaoui, Indexation et recherche d'images par le contenu: Algorithmes et application au Lifelogging, 2017.

[7] A. C. Amnar Kumar, *Audio Stegging using Evolutionary Algorithms,* 2023.

[8] Elsevier, Methods and Applications in Petroleum and Mineral Exploration and Engineering Geology, Elsevier, 2022.

[9] GeeksforGeeks, “Types of Image,” GeeksforGeeks, 02 November 2022. [Online]. Available: https://www.geeksforgeeks.org/types-of-image/. [Accessed 06 May 2025].

[10] A. C. L. M. S. M. M. T. S. R. T. W. A. G. Alejandro Giacometti, “Visualising Macroscopic Deterioration of Parchment and Writing via Multispectral Images,” in *Care and Conservation of Manuscripts 15,* 2016.

[11] S. X. S. Z. Peng Lei, “An art-oriented pixelation method for cartoon images,” *The Visual Computer,* vol. 40, no. 2, 2023.

[12] T. B. Chris Solomon, Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab, 2010.

[13] M. J. B. Wilhelm Burger, Digital Image Processing: An Algorithmic Introduction Using Java, Springer, 2008.

[14] A. Leong, “Do you know python grayscale histogram analysis?,” Medium, 06 Octobre 2023. [Online]. Available: https://medium.com/@meiyee715/do-you-know-python-grayscale-histogram-analysis-b1f8825aef7d. [Accessed 06 May 2025].

[15] GeeksforGeeks, “Visualizing Colors in Images Using Histogram in Python,” GeeksforGeeks, 03 January 2023. [Online]. Available: https://www.geeksforgeeks.org/visualizing-colors-in-images-using-histogram-in-python/. [Accessed 06 May 2025].

[16 Computer Science, “How to Plot a Histogram in Python using OpenCV,” YouTube, 06 July 2021. [Online]. Available: https://www.youtube.com/watch?v=tUs7Glv7lpA. [Accessed 06 May 2025].

[17 Rajilini, “Histogram Equalization — All You Want to Know!,” Medium, 30 January 2020. [Online]. Available: https://medium.com/@rajilini/histogram-equalization-all-you-want-to-know-13b0884c5296. [Accessed 06 May 2025].

[18 W. H.-W. C. Y.-C. L. J.-S. S. W.-K. H. N.-I. Chen Tseng-Yi, “Integrating an e-book Software with Vector Graphic Technology on Cloud Platform,” *Procedia - Social and Behavioral Sciences,* vol. 176, 2015.

[19 A. K. A. R. Vedarutvija Joopally, “Image Processing: Comparison & Analysis of,” *International Journal for Research in Applied Science & Engineering Technology (IJRASET),* vol. 11, no. 6, June 2023.

[20 D. T. Y. I. K. N. Yuji Takeuchi, “ASCII art generation using the local exhaustive search on the GPU,” *First International Symposium on Computing and Networking,* 2013.

[21 I. H. C. D. H. R. H. J. R. L. a. P. A. G. Richard H. Wiggins, “RadioGraphics,” vol. 21, no. 3, pp. 789-798, 2001.

[22 abdelhayajaib, “Most Popular SVG Files for Your Crafting Projects,” popular svg, 26 November 2024. [Online]. Available: https://popularsvg.com/most-popular-svg-files-for-your-crafting-projects/. [Accessed 06 May 2025].

[23 GeeksforGeeks, “Digital Image Processing Basics,” GeeksforGeeks, 22 February 2023. [Online]. Available: https://www.geeksforgeeks.org/digital-image-processing-basics/. [Accessed 06 May 2025].

[24 A. Kumar, “What Is Image Processing : Overview, Applications, Benefits, and More,” somplilearn, 13 April 2025. [Online]. Available: https://www.simplilearn.com/image-processing-article#:~:text=Image%20processing%20involves%20performing%20operations,important%20details%20from%20the%20image.. [Accessed 06 May 2025].

[25 G. R. V. S. L. J. Ashwini A. Salunkhe, “Progress and Trends in Image Processing Applications in Civil Engineering: Opportunities and Challenges,” *Advances in Civil Engineering,* no. 10.1155/2022/6400254., pp. 1-17, 2022.

[26 B. a. S. P. Chitradevi, “An overview on image processing techniques.,” *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 2, no. 11, pp. 6466-6472, 2014.

[27 MathWorks, “Feature Extraction,” MathWorks, 2024. [Online]. Available: https://www.mathworks.com/discovery/feature-extraction.html. [Accessed 6 May 2025].

[28 A. Elelu, “ MFCC — The Dummy’s Guide,” Medium, 26 July 2020. [Online]. Available: https://medium.com/@abdulsalamelelu/mfcc-the-dummys-guide-fd7fc471db76. [Accessed 6 May 2025].

[29 GeeksforGeeks, “What is Image Classification?,” GeeksforGeeks, 04 July 2023. [Online]. Available: https://www.geeksforgeeks.org/what-is-image-classification/. [Accessed 6 May 2025].

[30 SuperAnnotate, “Image Classification Basics,” SuperAnnotate, 2023. [Online]. Available: https://www.superannotate.com/blog/image-classification-basics#how-image-classification-works. [Accessed 6 May 2025].

[31 X. &. L. Y. &. L. J. &. W. T. &. Q. Z. Zhu, Advances in Knowledge Discovery and Data Mining, 2018.

[32 S. R. a. P. Norvig, “Artificial Intelligence: A Modern Approach”.

[33 CHAPTER 20 - Convolutional neural networks.

[34 T. Wang, "Recurrent Neural Networks", 2015.

[35 M. Hayat, “Dimension reduction methods and artificial learning for the facial pose estemation”.

[36 “MIT Press Chapter 10: Sequence Modeling: Recurrent and Recursive Nets,” 2016.

[37 “Gated Recurrent Unit Networks (GRU) in Deep Learning,” 2025. [Online].

[38 E. Reiter, Natural Language Generation. Springer, 2024.

[39 A. T. S. B. D. E. Oriol Vinyals, Show and Tell: A Neural Image Caption Generator.

[40 R. Szeliski, Computer Vision: Algorithms and Applications (2nd ed.). Springer., 2022.

[41 T. e. a. Young, Recent Trends in Deep Learning Based Natural Language Processing., 2018.

[42 “www.shaip.com,” [Online]. Available: www.shaip.com.

[43 J. H. &. C. Stryker, “Whats NLP,” 2024. [Online].

[44 p. ting, *A survey on automatic generation of medical imaging reports based on deep learning”.*

[45 M. Z. e. a. Hossain, “Comprehensive Survey of Image Captioning.,” 2019.

[46 geeksforgeeks, “VGG-16 | CNN model,” [Online]. Available: https://www.geeksforgeeks.org/computer-vision/vgg-16-cnn-model/.

[47 Neurohive, “VGG16 - Convolutional Network for Classification and Detection,” Neurohive, 20 November 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/.

[48 university stanford, “DenseCap: Fully Convolutional Localization Networks for Dense Captioning,” [Online]. Available: https://cs.stanford.edu/people/karpathy/densecap/.

[49 S. R. T. W. a. W.-J. Z. Kishore Papineni, “Bleu: a Method for Automatic Evaluation of Machine Translation,” p. 311–318.

[50] S. &. L. A. Banerjee, METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. ACL., 2005.

[51] R. Z. C. L. &. P. D. Vedantam, CIDEr: Consensus-based Image Description Evaluation. CVPR., 2015.

[52] C. e. al, Unsupervised Cross-lingual Representation Learning at Scale (XLM-R), 2020.

[53] H. D. K. &. A. A. Mubarak, " Arabic Dialect Identification in the Wild. LREC.," 2020.

[54] M. D. K. M. H. H. D. A. Obeida ElJundi, "Resources and End-to-End Neural Network Models for Arabic Image Captioning," *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP),* vol. 5, pp. 233 - 241, 2020.

[55] A. Jain, "Flickr 8k Dataset," Kaggle, 27 April 2020. [Online]. Available: https://www.kaggle.com/datasets/adityajn105/flickr8k.

[56] gettyimages, "4,977 Algeria Tourism Stock Photos, High-Res Pictures, and Images - Getty Images," gettyimages, 27 April 2024. [Online]. Available: https://www.gettyimages.com/photos/algeria-tourism?utm_source=chatgpt.com.

[57] K. Y. P. Y. P. R. S. S. Shetty Ashish, "Optimal transformers based image captioning using beam search," *Multimedia Tools and Applications,* vol. 83, pp. 1 - 15, 2023.

[58] Y. B. A. C. Ian Goodfellow, Deep Learning, MT Press, 2016.

[59] J. L. B. Diederik P. Kingma, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *International Conference for Learning Representations,* vol. 3, pp. 1 - 15, 2015.

[60] P. C. John G. Carney, "The epoch interpretation of learning," *IEEE Transaction on Neural Networks,* vol. 8, pp. 111 - 116, 1998.

[61] C. L. L. R. C. H. W. R. Sofia Lemons, "Beam Search: Faster and Monotonic," *The International Conference on Automated Planning and Scheduling,* vol. 32, pp. 222 - 230, 2022.

[62] X. Z. S. R. J. S. Kaiming He, "Deep Residual Learning for Image Recognition," *IEEE conference on computer vision and pattern recognition,* pp. 770 - 778, 2016.

[63] Q. V. L. Mingxing Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International conference on machine learning,* pp. 6105 - 6114, 2019.

[64] B. S. I. K. Wonjae Kim, "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision," *International conference on machine learning,* pp. 5583 - 5594, 2021.

[65] L. B. A. K. D. W. X. Z. T. U. M. D. M. M. G. H. S. G. J. U. N. H. Alexey Dosovitskiy, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *ICLR,* pp. 1 - 22, 2021.

[67 M. S. K. S. C. a. M. S. Singh, “"Various Image Compression Techniques: Lossy and Lossless."," *International Journal of Computer Applications* , pp. 23-26., May 2016.

[68 R. L. J. D. S. H. W. F. E. S. S. A. D. a. H. B. Khorasani, “ 1998. Web-based digital radiology teaching file: facilitating case input at time of interpretation.,” *AJR. American journal of roentgenology,* vol. 1, 1998.

[69 OpenCV.org, “Histogram Calculation,” OpenCV, Janvier 2018. [Online]. Available: https://docs.opencv.org/3.4/d1/db7/tutorial_py_histogram_begins.html. [Accessed 06 May 2025].

[70 M. Nielsen, "Neural Networks and Deep Learning", 2015.

[71 C. Gershenson, Artificial Neural Networks for Beginners.

[72 “What is a Convolutional Neural Network? – Zilliz.com – certified by AICPA SOC & ISO,” [Online]. Available: https://zilliz.com/glossary/convolutional-neural-network.

[73 A. G. a. S. Pal, Deep Learning with Keras, 2017.

[74 N. Habash, Introduction to Arabic Natural Language Processing. Springer., 2010.

[75 A. K. L. Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”.

[76 F. B. H. H. Wissam Antoun, “AraBERT: Transformer-based Model for Arabic Language Understanding,” *Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, vol. 4, pp. 9 - 15, 2020.

[77 J. B. L. D. M. F. Y. C. Ari Holtzman, “THE CURIOUS CASE OF,” in *ICLR*, 2020.