Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Babylon

College of Information Technology

Software Department

Semantic-based Image Captioning Using Deep Learning with Visual Attention Mechanism

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology - Software

by

Ali Saleem Haleem Jaber

Supervised by

Prof. Dr. Israa Hadi Ali

2024 A.D. **1445 A.**

بسم الله الرحمن الرحيم

حَتَّىٰٓ إِذَآ أَتَوْا۟ عَلَىٰ وَادِ ٱلنَّمْلِ قَالَتْ نَمْلَةٌ يَٰٓأَيُّهَا
ٱلنَّمْلُ ٱدْخُلُوا۟ مَسَٰكِنَكُمْ لَا يَحْطِمَنَّكُمْ سُلَيْمَٰنُ وَجُنُودُهُۥ وَهُمْ
لَا يَشْعُرُونَ ﴿١٨﴾ فَتَبَسَّمَ ضَاحِكًا مِّن قَوْلِهَا وَقَالَ رَبِّ أَوْزِعْنِىٓ
أَنْ أَشْكُرَ نِعْمَتَكَ ٱلَّتِىٓ أَنْعَمْتَ عَلَىَّ وَعَلَىٰ وَٰلِدَىَّ وَأَنْ أَعْمَلَ
صَٰلِحًا تَرْضَىٰهُ وَأَدْخِلْنِى بِرَحْمَتِكَ فِى عِبَادِكَ ٱلصَّٰلِحِينَ
﴿١٩﴾

صدق الله العظيم

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Software \ Collage of Information Technology \ University of Babylon, by **Ali Saleem Haleem Jaber** as a partial fulfillment of the requirements for the degree of **Master in Information Technology**.

Signature:
Name: **Prof. Dr. Israa Hadi Ali.**
Date: / / 2024

The Head of the Department Certification
In view of the available recommendation, we forward this thesis for debate by the examining committee.

Signature:
Name: **Prof. Sura Zaki Naji Alrashid.**
Date: / /2024

Certification of the Examination Committee

We, the undersigned, certify that (**Ali Saleem Haleem Jaber**) candidate for the degree of Master in Information Technology - Software, has presented his thesis of the following title (**Semantic-based Image Captioning Using Deep Learning with Visual Attention Mechanism**) as it appears on the title page and front cover of the thesis that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: / /2024.

Signature:

Name:

Title:

Date: / / 2024

(Chairman)

Signature:

Name:

Title:

Date: / / 2024

(Member)

Signature:

Name:

Title:

Date: / / 2024

(Member)

Signature:

Name:

Title:

Date: / / 2024

(Member & Supervisor)

Signature:

Name:

Title:

Date: / / 2024

(Dean of Collage of Information Technology)

Declaration

I hereby declare that this thesis, submitted to University of Babylon in partial fulfillment of requirement for the degree of Master in Information Technology - Software department, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source are appropriately cited in the references.

Signature:

Name: Ali Saleem Haleem Jaber

Date: / / 2024

Dedication

This work is dedicated to…

The martyrs of Iraq of all sects and nationalities.

The one who guides me to the way of Allah and Allah gave me guidance on his hands,

my Shafi'i on the Day of Deen,

the Beloved Prophet Muhammad (PBUH) and his successor Imam Ali (AS).

My father and My mother,

I ask Allah to protect you from all evil and I will not disappoint you.

To my beloved brothers and sisters,

who stand by me when things look very difficult.

Acknowledgements

All praise is for Allah. We praise Him, we seek His aid and we ask for His forgiveness. We seek Allah's refuge from the evils of ourselves and our evil actions. Whosoever Allah guides, no one can misguide him; and whosoever Allah misguides, there is no one who can guide him.

As humans, Allah has bestowed on us the nature to be grateful and we should thus express that gratitude not just to Allah but to the people whom we deal with as well. In many places in the Qur'an, Allah divides people as being grateful and as ungrateful to motivate us to join the camp of those who are grateful.

There are no proper words to convey my deep gratitude and respect for my supervisor, **Prof. Dr. Israa Hadi Ali**: I am forever grateful for her patience, guidance, wise counsel and most importantly, she has provided positive encouragement and warm spirit to finish this thesis.

To my wonderful Parents: Words cannot express my appreciation to you, the best father and mother. I wish to give you all thanks and love for your guidance, advice, invitations, and endless support.

My deepest gratitude goes to all of my family members. my brothers and sisters for the endless support. It would not be possible to write this thesis without their support.

Last but not least, I would like to thank those who helped me with a word of encouragement or scientific information that benefited me in this thesis: **Dr. Hassan Alrehamy, Dr. Ali Hammoud Al-Saadi, Mohammad Baqer Haleem, Ali Saadi Abbas, Qusai AL-Durrah** and all my teachers who have been so supportive along the way of doing my thesis. I thank them wholeheartedly.

Ali Saleem Haleem Jaber

Declaration Associated with this Thesis

Some of the work presented in this thesis has been accepted for publication as described below.

(First Paper)

Name of conference: The second International Conference on Scientific Research and Innovation 2023 (2ICSRI 2023).

Paper Title: A Deep Learning Based Approach for Image Captioning: Exploiting Priority Factor to Enhance Accuracy and Relevance.

Index Database: Scopus**,** Cite Score = 0.7

Publisher: American Institute of Physics (AIP) conference Proceedings

Authors: Ali S. Haleem**,** Israa H. Ali

(Second Paper)

Name of conference: International Conferences on Engineering, Applied and Nano Sciences and (ICEANZ-2023).

Paper Title: Feature Selection Approach for Image Caption Generation Using an Attention Mechanism.

Index Database: Scopus

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Authors: Ali S. Haleem**,** Israa H. Ali

Abstract

Image captioning involves transforming an input image into a textual description (caption), bridging the gap between Vision and Language in a creative manner. Various datasets can be annotated manually with appropriate captions for each image. However, such maneuvers can quickly become burdensome relative to the dataset size or the number of datasets. Therefore, there exists a critical need for adequate automatic image caption generation. The focal aim of this task is to create a human-like description of the content and context (semantic) of the image, often including details about objects, actions, and relationships within the scene.

This thesis proposes semantic-based captioning system to maximize the accuracy of caption generation. The system employs an encoder-decoder architecture that relies on attention mechanism to stimulate the selective observation of important image features from a set of available features. It comprises three key phases: Preprocessing, Feature Extraction (Encoding), and Feature Decoding. In the first phase, both image and caption data undergo preprocessing. In the second phase, two types of features are extracted: Generic features (deep visual features) extracted from pre-trained model and Specific features (object features) extracted from the object detection pre-trained model.

To improve the accuracy of the proposed system, this thesis exploits object detection to compute a new feature called Priority Factor that maximizes the activation of semantically important objects in a given image. Finally, these various features are combined together in the concatenation step. Unlike previous methods, the concatenation occurs before feature reduction, resulting in a raw and unprocessed feature matrix. The third phase is a deep neural network that consists of three processing blocks: Reduction, Attention, and Language. During this phase, the input

feature set is passed through extra layers (Dense, LSTM) for vector-based manipulation to learn feature-vocabulary associations and this, in turn, leads to generating a trained model.

The experiments are conducted using a common dataset in computer vision (MSCOCO) and evaluated its performance using eight metrics. The results prove the effectiveness of incorporating specific features. This improves various assessment measures, including BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR, CIDEr, ROUGH-L and SPICE, especially after adding the proposed Priority Factor feature to object features. Hence, a description corresponding to the main semantic content of the image is generated. Additionally, this system outperforms four state of the art approaches in BLEU-1 (76.2), BLEU-2 (60.1), BLEU-3 (47.2), and SPICE (15). As a result, the thesis findings can help individuals who are blind or visually impaired by generating descriptions of images found on the web, in documents, or in social media posts, allowing visually impaired individuals to understand the content of images they cannot see.

Table of Contents

1 Chapter One: General Introduction 1

1.1 Introduction 1

1.2 Research Problem 2

1.3 Related Works 3

1.4 Aims and Objectives 8

1.5 Thesis Significance 8

1.6 Thesis Challenges 9

1.7 Thesis Organization 9

2 Chapter Two: Theoretical Background 10

2.1 Overview 10

2.2 Image Captioning 10

2.2.1 Visual Features 11

2.2.2 Textual Features 12

2.3 Deep Learning 13

2.3.1 Deep Learning Types 13

2.3.2 Activation Function 14

2.3.3 Loss Function 18

2.3.4 Optimization Algorithms 19

2.3.5 Convolution Neural Network 21

2.3.6 Pre-trained Models for Image Classification 25

2.3.7 Recurrent Neural Network 30

2.3.8 Encoder-Decoder Architecture 33

2.3.9 Attention Mechanism 36

2.4 Object Detection 39

2.4.1 Two Stage Object Detection 39

2.4.2 One Stage Object Detection 39

2.5 Text Analysis 43

2.5.1 Text Preprocessing 43

2.5.2 Feature Extraction 44

2.6 Evaluation Measures 46

3 Chapter Three: The Proposed System 49
3.1 Overview 49
3.2 The Proposed System 49
3.2.1 The Used Dataset 51
3.2.2 The Preprocessing Phase 51
3.2.3 The Features Extraction Phase (Encoding) 54
3.2.4 Features Decoding Phase 61
3.3 Training of The Proposed System 67
4 Chapter Four: Experimental Results and Discussion 71
4.1 Overview 71
4.2 Software and Hardware Configurations 71
4.3 Implementation Details 72
4.4 Results of Data Preprocessing 72
4.5 Experiments 76
4.5.1 Results of the First Experiment 77
4.5.2 Results of the Second Experiment 80
4.5.3 Results of the Third Experiment 82
4.5.4 Results of the Fourth Experiment 83
5 Chapter Five: Conclusions and Future Work 85
5.1 Conclusions 85
5.2 Future Works 86

List of Figures

Figure 2.1: ReLU Activation Function [35] 15
Figure 2.2:Logistic (Sigmoid) Activation Function [33] 16
Figure 2.3: Tangent Hyperbolic (Tanh) Activation Function [33] 17
Figure 2.4: An example of convolution operation [54] 22
Figure 2.5:FC layers example with two layers [56] 24
Figure 2.6: Scaling the depth, width, and image resolution to create different variations of the EfficientNet model [64] 27
Figure 2.7: Training and Parameter efficiency of the EfficientNetV2 model compared with other state of the art models [21] 28
Figure 2.8:MBConv and Fused-MBConv Structure [21] 28
Figure 2.9:Traditional Residual Block and MBConv Block Structures [71] 29
Figure 2.10:LSTM Cell [76] 31
Figure 2.11:GRU Cell [75] 33
Figure 2.12:Encoder-Decoder Architecture of machine translation [5] 35
Figure 2.13:Encoder-Decoder architecture of image captioning [1] 35
Figure 2.14:The Bahdanau architecture [79] 37
Figure 2.15:ELAN and E-ELAN architecture [90] 42
Figure 2.16:Model Scaling of YOLOv7 [90] 42
Figure 3.1: A block diagram of the proposed system 50
Figure 3.2:MSCOCO dataset samples 52
Figure 3.3: Deep visual features extraction from the EfficientNetV2 model 56
Figure 3.4: Object Features Extraction and Bounding Box Representation 57
Figure 3.5:The results of the mathematical modeling of observations (A and B) with computing the pf feature for each detected object. 59
Figure 3.6: Priority Factor Feature Embedding with Standard Features of YOLOv7 59
Figure 3.7: Objects Sorting Ascending According to pf 60
Figure 3.8: Features Concatenation Workflow 60
Figure 3.9: The Architecture of Features Decoding Phase 61
Figure 3.10: Visual Representation of Attention Results. 65

Figure 3.11: Preparing Dataset for Training 70
Figure 4.1: Resize Sample of the Dataset Images 73
Figure 4.2: Comparison of the results of experiment 1. OD denotes Object Detection..... 78
Figure 4.3: Examples of captions generated by experiment 1. OD denotes Object Detection 79
Figure 4.4: Comparison of the results of experiment 2 81
Figure 4.5: Examples of captions generated by experiment 2 82
Figure 4.6: Comparison of the experiment 4 results between our system and state of the art approaches..... 84

List of Tables

Table 1.1: Summary of the Rreviewed Literature 7
Table 4.1: Sample of the dataset captions before and after adding unique tokens 73
Table 4.2: The captions sample before and after the tokenization process..... 74
Table 4.3: The captions sample before and after the dictionary construction 75
Table 4.4: Caption Transformation to Vector..... 75
Table 4.5 The use of padding..... 76
Table 4.6: Results of experiment 1. Bold indicates the highest score, and OD denotes Object Detection. 77
Table 4.7: Results of experiment 2. Bold indicates the highest score..... 80
Table 4.8: Results of experiment 3. Bold indicates the highest score..... 83
Table 4.9: Results of experiment 4. Bold indicates the highest score..... 83

List of Algorithms

Algorithm 3. 1: The Preprocessing Phase..... 53
Algorithm 3. 2: Reduction Block..... 62
Algorithm 3. 3: Attention Block (semantic stage)..... 64
Algorithm 3. 4: Language Block 67
Algorithm 3. 5: Training Iteration 69

List of Abbreviations

Abbreviation	Meaning
1D	One Dimensional
2D	Two Dimensional
Adam	Adaptive Moment Estimation
AI	Artificial Intelligent
AIC	Automatic Image Captioning
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CBIR	Content-Based Image Retrieval
CIDEr	Consensus-Based Image Description Evaluation
CNN	Convolution Neural Network
DBN	Deep Belief_Network
DenseNet	Densely Connected Convolutional Networks
ED	Encoder-Decoder
E-ELAN	Extended Efficient Layer Aggregation Network
ELAN	Efficient Layer Aggregation Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
FC layer	Fully Connected Layer
GAN	Generated Adversarial Networks
GloVe	Global Vectors
gLSTM	Guided Long short-term memory
GPT	Generative Pre-trained Transformer
GRU	Gated Recurrent Unit
HOG	Histogram of Oriented Gradients
LBP	Local Binary Patterns
LCS	longest common subsequence
LSTM	Long short-term memory

METEOR	Metric for Evaluation of Translation with Explicit Ordering
MSCOCO	Microsoft Common Objects in Context
MSE	Mean Square Error
MSR-VTT2016	Microsoft Research Video to Text
NLP	Natural Language Processing
NMT	Neural Machine Translation
OWL	Web Ontology Language
PF	Priority Factor
RBM	Restricted_Boltzmann_Machine
RDF	Resource Description Framework
ReLU	Rectified Linear Activation
ResNet	Residual Network
RNN	Recurrent neural network
ROUGE-L	Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SMS	Social Media Sites
SPICE	Semantic Propositional Image Caption Evaluation
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machines
Tanh	Hyperbolic Tangent
VGG	Visual Geometry Group
VGGNet	VGG Very Deep Convolutional Networks
YOLO	You only live once

Chapter One
General Introduction

Chapter One: General Introduction

1.1 Introduction

The integration of computer vision and Natural Language Processing (NLP) paves the way for a captivating task known as image captioning, which aims to generate descriptive and coherent textual descriptions for images. In today's digital era, images and videos are pervasive across the internet, social media, e-commerce, and various other online platforms. While the visual content in such files can offer rich and interesting elements, interpreting the semantics of such contents may present a significant challenge from a machine readability viewpoint [1].

Automatic Image Captioning (AIC) is a computer vision task that aims to redeem the gap between visual and semantic comprehension by enabling machines to generate descriptive and contextual captions for images. AIC often involves detecting and recognizing objects within an image, reasoning the relationships between them, and then utilizing a language model to describe them with short sentences. AIC offers benefits for an overload of applications in different domains such as biomedicine, education, digital libraries, semantic web, and so on [2].

Existing AIC systems utilize deep learning models to automatically generate captions [3], [4]. By training such models on large image sets, a deep learning approach learns the best model to annotate an unseen image with appropriate caption. Such approach learns relations between overall image features and corresponding words in the caption. Recent systems are typically based on Encoder-Decoder (ED) architecture [5], which involves a two-step process. The encoder analyzes the input image to extract its features, which are then passed to the decoder using a language model based on Long Short-Term Memory (LSTM) [6], Gated Recurrent Unit (GRU) [7], or their derived adaptations, generating a coherent and

descriptive caption based on these features. The attention mechanism assumes a central role within the domains of AIC and machine translation, prominent aspects of deep learning research. Empirical evidence underscores its effectiveness in augmenting accuracy by compelling the model to selectively focus on diverse aspects of the input data during the generation of output sequences [8].

To comprehend an image, the AIC system employs a pre-trained Convolution Neural Network (CNN) [9] to extract generic feature matrices from the final convolutional layers. Such workflow facilitates the comprehensive understanding of various object aspects and their interconnections within the image. Many approaches have been proposed to investigate the incorporation of object detection features along with generic features, so this type of feature guides the attention toward particular regions within the image [10]. Therefore, the model may learn particular relation features and their corresponding words in the language model. Notably, the models employed in these investigations usually are YOLOv3 [11], YOLOv4 [12], and YOLO9000 [13].

1.2 Research Problem

In this era of big data, images contain a large portion of unstructured data available on the internet. However, this information source must be available for various information management tasks (such as information retrieval, semantic search, and summarization). Accordingly, previous typical AIC systems are based on ED architecture, which involves a two-step process. The encoder analyzes the input image to extract its features, which are then passed to the decoder using a specific language model, generating a coherent and descriptive caption based on those features. However, there can be always certain features in a given image which may be regarded as more important than others in accurately defining the semantics of the image's content. Therefore, such a system should investigate the incorporation of specific

features to guide the attention toward particular regions of importance within the image.

1.3 Related Works

Some of the related works are reviewed here. Table 1.1 shows a summary of the reviewed studies.

Vinyals et al [3] in 2015 introduced an AIC approach that consists of caption generation and image feature extraction components. This approach extracts image features from an input image and produces captions using CNN and LSTM. LSTM initial state incorporates image information, and then the current time step and the prior hidden state are used to generate words from the language model. This workflow continues until the final token is discovered. Since image data is only fed at the initial phase of the workflow, it may suffer a vanishing gradient. For empirical evaluation, the researcher used multiple datasets, namely: PASCAL, Flickr 30k, SBU, and MSCOCO. The performance analysis shows improved captioning task, with BLEU-1 score of 59 on PASCAL dataset, 66 on Flickr 30k dataset, and 28 on SBU dataset. Whereas with MSCOCO dataset they achieve a BLEU-4 of 27.7, METEOR 23.7, and CIDER 85.5.

Jia et al [14] in 2015 introduced an augmentation of the LSTM model referred to as Guided Long short-term memory (gLSTM) in order to address the gradient issue during the generation of lengthy sentences. The gLSTM incorporates global semantic knowledge into both the gate and cell state of LSTM. In their empirical evaluation, they obtained scores of 0.647, 0.459, 0.318, 0.216, and 0.201 on BLEU-1, BLEU-2, BLEU-3, BLEU-4, and METEOR respectively for the Flickr8k dataset. Correspondingly, for the Flickr30k dataset, the scores were 0.646, 0.466, 0.305, 0.206, and 0.179, which outperformed the scores for Flickr8k. Meanwhile, the

highest scores achieved were on the MSCOCO dataset, with values of 0.670, 0.491, 0.358, 0.264, and 0.227 respectively. Such CNN/Recurrent neural network (RNN) based approaches which utilize unidirectional LSTM architectures are characterized by their limited depth. In the case of unidirectional language generation techniques, the prediction of the next word relies on the visual context and all preceding textual contexts. Consequently, the employment of unidirectional LSTM poses limitations in generating contextually coherent captions.

In 2016, Wang et al [15] proposed an advanced approach for generating image captions, using a deep architecture based on bidirectional LSTM (Bi-LSTM). The proposed method shows the capability to produce image captions that are both contextually and semantically comprehensive. The researchers conducted experiments using various datasets, including Flickr 8k, Flickr 30k, and MSCOCO. Notably, their approach yielded the most favorable outcomes on the Flickr 8k dataset, achieving a BLEU-2 score of 46.8, a BLEU-3 score of 32.0, and a BLEU-4 score of 21.5.

Yang et al [16] proposed in 2017 a multi neural network model designed to automatically generate descriptive captions for images. The model follows a two-step process, it starts with the extraction of object information and its respective spatial locations within the image using VGGNet and Fast R-CNN pre-trained models. Subsequently, an RNN with LSTM units and an attention mechanism are employed to generate captions based on this extracted information. By incorporating both object information and spatial locations, the model can effectively capture the content of images and generate contextually relevant captions. For empirical evaluation, they achieved the best results on MSCOCO dataset with BLEU-3 score of 39.2, CIDEr 85, and ROUGH-L 52.1.

Yin and Ordonez [17] in 2017 introduced an ED-based approach that encodes a

given input as a sequence of objects along with the position of each object using LSTM network during the encoding phase. During decoding, the approach utilizes LSTM as the language model for interpreting the input representation and assigning appropriate captions. Although the researchers achieved good captioning results using YOLO9000 for object detection and Visual Geometry Group (VGG) for visual feature processing, they also demonstrated better results by combining CNN and YOLO. Thus, they improve an image captioning model from 0.863 to 0.950 (CIDEr score) on MS-COCO dataset. Notably, they did not use all available data from the object features produced by YOLO, such as object dimensions and confidence.

Vo-Ho et al [18] in 2019 developed an AIC system, encompassing the extraction of specific features derived from YOLO9000 and Faster Region-based Convolutional Neural Network (Faster R-CNN). Subsequently, an attention module was employed to process each category of features that represent the area that the model is now focusing on. Employing an LSTM model, term probabilities within the predefined vocabulary were estimated iteratively using the incorporation of the two sets of localized features. The refinement of findings and output optimization was achieved via a beam search methodology, facilitating the selection of the most optimal caption outcome. Accordingly, they achieved on MSCOCO dataset the scores 0.723, 0.554, 0.414, 0.310, 0.250, 0.532, and 0.942 on BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR, ROUGE-L, and CIDEr respectively.

Iwamura et al [19] in 2021 suggested an approach that involves feature extraction (using ResNet-101), motion estimation, object detection (using Faster R-CNN,) as well as captioning. For empirical evaluation, the researcher used multiple datasets, namely: a public image collection, MSR-VTT2016, and MSCOCO. The performance analysis shows improved captioning task, with BLEU score 49.9 and METEOR 16.1 on MSR-VTT2016, whereas with BLEU 75.9 and METEOR 26.7

on MSCOCO.

Al-Malla et al [20] in 2022 presented an ED-based system with improved architectural design by incorporating the concept of attention. The system relies on an Xception a pre-trained model to generate convolutional features. The system incorporates object features obtained from YOLOv4, which was pre-trained on MSCOCO dataset. This work introduces a notable contribution of a novel positional encoding scheme for object features known as the important factor. The main idea here is to balance the significance of foreground large objects by assigning them higher confidence scores, while background small objects receive relatively insignificant confidence scores. By incorporating the important factor, the approach effectively prioritizes the representation and attention given to prominent objects within the image. The researchers evaluated their system using MSCOCO and Flickr30k datasets and compared its performance to that of similar works. Their new feature extraction system led to a 15.04% increase in the CIDEr score, but on the other hand, their system decreased the METEOR score from 0.164 to 0.163, especially after adding the importance factor.

All of the previous studies mentioned above have not considered the advantages of all the object features provided by the object detection model, which has recently been shown to be rich in semantic content. In addition, they did not directly use the raw features extracted from the feature extraction model, which contains more context information that enhances the understanding of image content. Unlike them, this thesis takes into account all available features of the object as well as it uses used raw features directly.

Table 1.1: Summary of the Rreviewed Literature

References	Image Encoding	Features Decoding	Dataset	Evaluation Metrics
2015 [3]	CNN	LSTM	PASCAL	BLEU-1 (59)
			Flickr 30k	BLEU-1 (66)
			SBU	BLEU-1 (28)
			MSCOCO	BLEU-4 (27.7), METEOR (23.7), CIDER (85.5)
2015 [14]	VGGNet	LSTM	Flickr 8K	BLEU-1 (64.7), BLEU-2 (45.9), BLEU-3 (31.8), BLEU-4 (21.6), METEOR (20.1)
			Flickr 30K	BLEU-1 (64.6), BLEU-2 (46.6), BLEU-3 (30.5), BLEU-4 (20.6), METEOR (17.9)
			MS COCO	BLEU-1 (67), BLEU-2 (49.1), BLEU-3 (35.8), BLEU-4 (26.4), METEOR (22.7)
2016 [15]	VGGNet	Bi-LSTM	Flickr 8K	BLEU-1 (65.5), BLEU-2 (46.8), BLEU-3 (32), BLEU-4 (21.5)
			Flickr 30K	BLEU-1 (62.1), BLEU-2 (42.6), BLEU-3 (28.1), BLEU-4 (19.3)
			MS COCO	BLEU-1 (67.2), BLEU-2 (49.2), BLEU-3 (35.2), BLEU-4 (24.4)
2017 [16]	VGGNet, Fast R-CNN	LSTM	MS COCO	BLEU-1 (70.4), BLEU-2 (53.1), BLEU-3 (39.2), BLEU-4 (29), METEOR (23.8), CIDEr (85), ROUGH-L (52.1)
2017 [17]	YOLO9000, VGGNet	LSTM	MS COCO	BLEU-4 (25.3), METEOR (23.8), CIDEr (92.2), ROUGH-L (50.7)
2019 [18]	ResNet, YOLO9000, Faster R-CNN	LSTM	MS COCO	BLEU-1 (72.3), BLEU-2 (55.4), BLEU-3 (41.4), BLEU-4 (31), METEOR (25), CIDEr (94.2), ROUGH-L (53.2)
2021 [19]	Faster R-CNN, ResNet-101	LSTM	MSR-VTT2016	BLEU-1 (49.9), BLEU-2 (32.2), BLEU-3 (21.5), BLEU-4 (14.5), METEOR (16.1), CIDEr (32.7), ROUGH-L (39.5)
			MSCOCO	BLEU-1 (75.9), BLEU-2 (59.9), BLEU-3 (46), BLEU-4 (35.2), METEOR (26.7), CIDEr (109.9), ROUGH-L (55.8)
2022 [20]	Xception, YOLOv4	GRU	MS COCO	BLEU-1 (49.2), BLEU-2 (29.6), BLEU-3 (17.4), BLEU-4 (10.1), METEOR (16.3), CIDEr (39), ROUGH-L (35.8), SPICE (10.8)
			Flickr30k	BLEU-1 (39.8), BLEU-2 (22.1), BLEU-3 (11.6), BLEU-4 (6.1), METEOR (12.9), CIDEr (15), ROUGH-L (29.8), SPICE (7.4)

1.4 Aims and Objectives

The primary aim of this thesis is to propose a system that can understand the image content (semantic context) and then try to describe such semantics with appropriate labels. This goal is divided into three distinct objectives:

- The first objective involves preprocessing the dataset to be well-suited for the specific task at hand.
- The second objective is to build an encoder model that extracts two types of features from the input image: deep visual features (i.e. Generic), and object features (i.e. Specific).
- The third objective is to build an attention-based decoder model that employs these various features as input and the output is a sequence of encoded term set that represents the final image caption.

1.5 Thesis Significance

Overall, the framework utilized in this thesis is built upon and affected by several previous studies. The main contributions are as follows:

- During the encoding phase, we generate and combine two types of features: Generic features (deep visual features), and Specific features (object features). Unlike the previous works which encode specific and general features separately and then combine them through concatenation, this thesis concatenates specific and generic features before the features dimension reduction stage.
- Additionally, we introduce a new method to generate a new object driven feature called Priority Factor (*pf*) as a mean to rank all the objects that are detected during the encoding phase based on their semantic prominence in the image's landscape.

1.6 Thesis Challenges

There are two challenges faced during different stages of this thesis:

- The first challenge is addressing the issue of caption accuracy while considering the extensive embedding capacity of words within the textual content.
- The second challenge is overcoming the inherent compositional nature of both NLP and computer vision, which involves handling complex interactions between linguistic and visual elements.

1.7 Thesis Organization

This thesis is organized as follows: Chapter Two presents a comprehensive description of the main principles used in thesis "theoretical background". Chapter Three explains the details of the proposed system. In Chapter Four, we discuss the evaluation and empirical analysis in terms of metrics, performance, and comparison against multiple state of the art approaches. Finally, this work and its future directions are conducted in Chapter Five.

Chapter Two
Theoretical Background

Chapter Two: Theoretical Background

2.1 Overview

The fascinating and quickly evolving topic of image captioning lies at the convergence of Natural Language Processing (NLP) and computer vision. It focuses on developing Artificial Intelligent (AI) systems that can generate descriptive textual captions for images, allowing machines to understand and communicate the content of visual data. This technology has deep implications for various industries, including healthcare, autonomous vehicles, and social media, as it enables computers to provide context and meaning to visual information. This chapter introduces the fundamental information and concepts employed throughout this thesis to achieve the image captioning task.

2.2 Image Captioning

Many images are available every day from various sources, including the internet, news stories, document diagrams, and commercials. Viewers would have to interpret such images in these sources. Although most images lack a description, many individuals can still grasp them without the help of their captions. Nevertheless, if individuals require automatic generation of image captions, a machine must be able to understand some kinds of image caption [1].

Image captioning is crucial for a variety of causes. They can be applied, for instance, to automatically index images. Image indexing has several applications in e-commerce, biomedicine, education, military, web searches, and digital libraries since it is crucial for Content-Based Image Retrieval (CBIR). Facebook and Twitter have the ability to produce descriptions from images directly. The descriptions can

include where we are (e.g., beach, cafe), what we wear and importantly what we are doing there [21], [22].

Additionally, image captioning represents a prominent area of AI research, addressing the confluence of image comprehension and natural language description generation. Image comprehension necessitates the detection and recognition of objects, understanding scene features, object attributes, and their interrelationships. The task of generating linguistically well-structured sentences requires a comprehensive grasp of both syntax and semantics within the language domain [23]. Image captioning indeed depends on two fundamental types of features: Visual features and Textual features. These features are essential for bridging the gap between the visual content of an image and the textual description generated by an AI system.

2.2.1 Visual Features

Visual features are used to understand image content. They capture information about objects, scenes, shapes, colors, and other visual elements. The methods utilized to extract these features can be generally classified into two groups: Deep machine learning methods and Traditional machine learning methods [20], [24].

Handcrafted features such as Scale-Invariant Feature Transform (SIFT) [25], the Histogram of Oriented Gradients (HOG) [26], Local Binary Patterns (LBP) [27], and a combination of these features are commonly employed in traditional machine learning. These methods take the input data and extract its features. After that, they are given to a classifier such as Support Vector Machines (SVM) [28] to classify an item. Extracting features from a huge and diverse amount of data is not practicable since handcrafted features are task specific. Real-world data, including videos and images, are also complicated and have various semantic meanings [1].

Deep machine learning methods are capable of handling a wide variety of videos and images since their features are automatically learned from training data. For instance, feature learning is frequently accomplished using Convulsion Nurul Network (CNN), while classification is accomplished using a classifier like Softmax [29].

2.2.2 Textual Features

Textual features, also known as linguistic features, are representations of the language used in captions. They capture syntactic and semantic information in the text, including the choice of words, grammar, and the relationships between words. Textual features are vital for image captioning as they enable the AI model to generate coherent and contextually appropriate descriptions in natural language. These features help ensure that the captions are grammatically correct, semantically meaningful, and aligned with the visual content [30]. There are several methods to extract features from captions. One common approach is to use NLP techniques to convert the words in the captions into numerical vectors. This can be done using methods like Word2Vec [31], GloVe [32], or FastText [32], which map words to high-dimensional vectors based on their semantic meanings.

Another method involves using Recurrent Neural Network (RNN) or transformers to process the sequential nature of captions. Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) are types of RNN commonly used for this purpose. Transformers, especially models such as Bidirectional Encoder Representations from Transformers (BERT) [33], can also capture contextual information from words in the captions [34].

Additionally, pre-trained language models, such as OpenAI's GPT or Google's BERT, can be fine-tuned on captioning datasets to extract relevant features. These

models learn to understand the context and relationships between words in a sentence, making them effective for feature extraction in image captioning tasks [35].

2.3 Deep Learning

Deep learning is a field of AI and both are elements of machine learning. It consists of hierarchical architecture, where each higher layer builds on its previous lower layer, which makes it popular for the first time as hierarchical learning. The beginning of deep learning was in 2007 and it works well with a massive amount of data to solve a complicated problem. Many applications improved with it such as Object Recognition, Object detection, Automatic Machine Translation, Automatic Image Captioning, Investment Modeling, drug discovery, and so on [36].

Additionally, deep learning can extract high-level features from the input data. For example, image processing, image segmentation, edge extraction, face verification and shape recognition. Notably, deep learning algorithms, such as CNN, surpass traditional algorithms in terms of efficiency, largely due to their ability to learn automated behaviors without necessitating extensive manual feature engineering, as is typically required in traditional machine learning models [37].

2.3.1 Deep Learning Types

There are three main types of deep learning models, each has its specific network architectures inside and has specific use.

a) Supervised Deep Learning

In Supervised Deep Learning algorithms, the algorithm is trained on pre-labeled data. Then the model is used the learning algorithm to adjust itself, and during the testing phase, the model should determine the correct answer without relying on any

label. Supervised deep learning has two main fields: classification problems and regression problems. One of the most frequently supervised models used is a CNN. Some of the CNN-based supervised deep learning architectures are AlexNet, LeNet-5, VGGNet, GoogleNet, ResNet, InceptionNet, EfficientNet, and others [38], [39].

b) Unsupervised Deep Learning

In the unsupervised Deep Learning algorithms, the model is trained based on unlabeled data, and the model tries for extracting patterns and features on its own. Some of the unsupervised deep learning architectures are Deep Belief_Network (DBN), Restricted_Boltzmann_Machine (RBM) and Generated Adversarial Networks (GAN) [40].

c) Semi-Supervised Deep Learning

Semi-Supervised Deep Learning takes on an intermediate stage between supervised and unsupervised learning. It takes a lot of categorized data to support a larger collection of unlabeled data. This approach is especially useful when it is difficult to extract relevant data features, and when labelling the samples takes a long time [41].

2.3.2 Activation Function

In the context of deep learning, an activation function is a mathematical operation applied to the output of a neuron (or node) in a neural network. An activation function determines if a neuron is activated or not. This means that during the prediction phase, it will employ simpler mathematical operations to decide if the neuron's input to the network is necessary or not. Activation function is to create output from a set of input values presented to a node (or a layer) [42]. There are several activation functions that are commonly used in deep learning. Each is with its own characteristics. In this thesis used the most four activation functions that are

important for use in hidden layers such as ReLU, Sigmoid, Tanh, and Softmax are used.

a) Rectified Linear Activation

Rectified Linear Units (ReLU) is widely used in neural networks, especially deep learning models. It sets the threshold at 0, simply stated, it produces 0 when x is zero or a negative value and returns the same value in another state. The function is represented by Equation 2.1 [43]. Figure 2.1 represents the ReLU Activation Function plot.

$$ReLU\ (x) = \max(0, x) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Where x represents the input to a neuron.

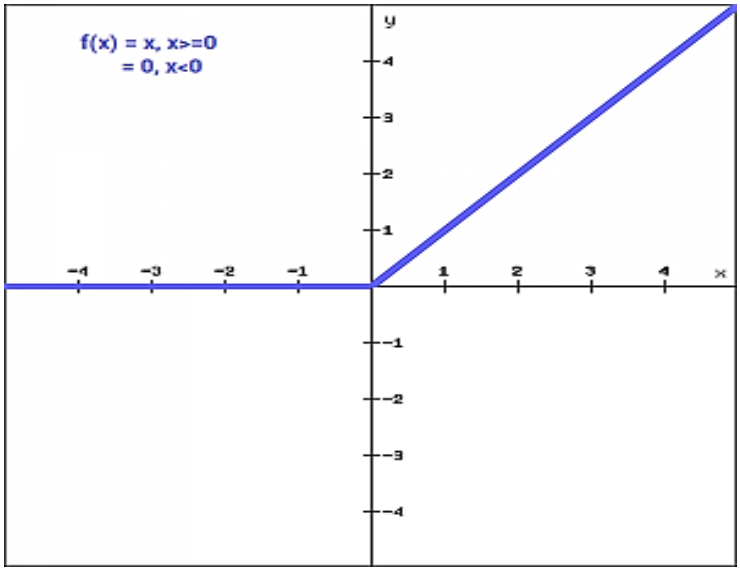


Figure 2.1: ReLU Activation Function [35]

b) Sigmoid

Since sigmoid is a non-linear function, it is the most commonly employed especially in binary classification. The Sigmoid activation function transforms the value in the

[0 – 1] range [42]. As shown in Figure 2.2, it can be summed up as presented in Equation 2.2.

$$f(x) = \frac{1}{(1+e^{-x})} \quad \text{..... (2.2)}$$

Where x represents the input to a neuron.

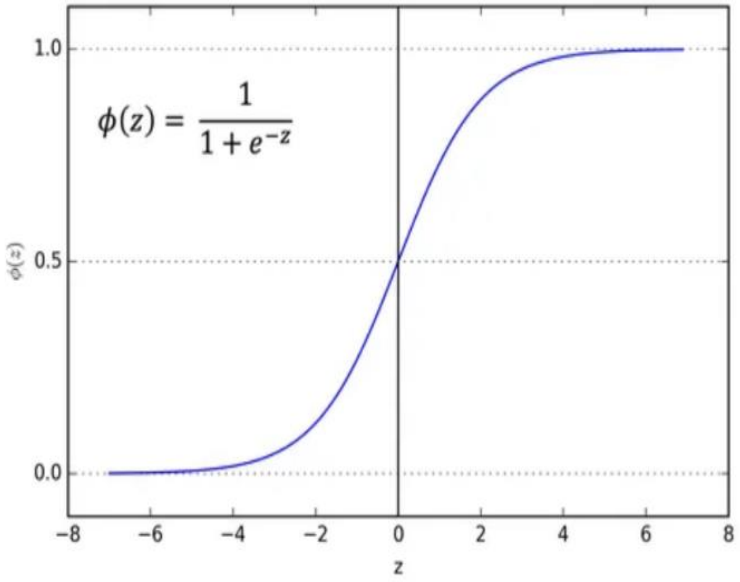


Figure 2.2:Logistic (Sigmoid) Activation Function [33]

c) Tangent Hyperbolic

The Tangent Hyperbolic (Tanh) function resembles the sigmoid activation function. However, it is symmetric across the origin. As a consequence, the outputs from previous layers would have different signs when supplied as input to the following layer [44]. It can be calculated as in Equation 2.3.

$$f(x) = 2 \times sigmoid(2x) - 1 \quad \text{..... (2.3)}$$

Where x represents the input to a neuron.

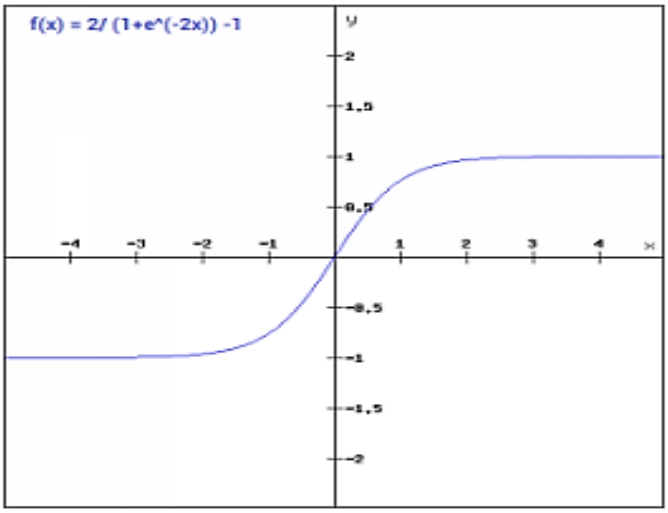


Figure 2.3: Tangent Hyperbolic (Tanh) Activation Function [33]

The Tanh activation function is a continuous and differentiable function with values ranging from -1 to 1 as shown in Figure 2.3. The Tanh function has a steeper gradient than the Sigmoid function.

d) Softmax

The softmax function is one of the activation functions that is widely used in neural computing at the last layer to calculate the multiple probability distributions of multi classes of more than two classes by using a vector of real numbers. The output of Softmax function values in the range between 0 and 1, where the summation of the probabilities is equal to 1, and the target class which have the highest value. Equation 2.4 represents the Softmax function [44].

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \quad \ldots\ldots\ldots\ldots\ldots (2.4)$$

Where:

k: are real numbers

x_i: represents an element of the input vector x

2.3.3 Loss Function

In early models of neural networks, the error is measured by calculating the difference between the real output and the expected output. At the moment, several formulas have appeared for calculating errors in neural networks, these formulas are called Loss Functions [45]. Using various loss functions can lead to a different error value for the same prediction. Therefore, the type of loss function has a major impact on the output of the network. There are three main loss function types: Classification Loss, Regression Loss, and Embedding Loss Functions [46]. The classification loss functions are used with classification problems. The Regression Loss functions are used in regression problems, while the Embedding loss functions are used with tasks that need to measure the similarity between two inputs [47].

a) Binary Cross Entropy

Binary Cross Entropy is a loss function commonly used in binary classification tasks, especially in machine learning and neural networks. It measures the difference between two probability distributions: the actual distribution and the predicted distribution. In the context of binary classification, where the target variable can take on two possible values (usually 0 and 1), Binary Cross Entropy quantifies the difference between the predicted probabilities and the actual binary outcomes [48].

b) Categorical Cross Entropy

This loss function uses in Multi-class classification problems. In these problems, the target can only belong to one out of many possible categories. This function is designed to calculate the difference between two probability distributions [49]. Equation 2.5 illustrates the categorical cross-entropy.

$$L(X_i, Y_i) = -\sum_{j=1}^{c} y_{ij} * \log(p_{ij}) \quad \text{..... (2.5)}$$

$$where\ Y_i\ is\ one-hot\ encoded\ target\ vector\ (y_{i1}, y_{i2}, \ldots\ldots, y_{ic})$$

$$y_{ij} = \begin{cases} 1, & if\ the\ i_{th} element\ is\ in\ class\ j \\ 0, & otherwise \end{cases}$$

$$p_{ij} = f(X_i) = Probability\ that\ i_{th} element\ is\ in\ class\ j$$

c) Mean Square Error

Mean Square Error (MSE) is one of the important regression loss functions, which measures the square difference between the real and expected values [48].

d) Euclidean Distance Loss

This loss function is an embedding loss usually used in problems that need to measure the similarity between two inputs, not for classification problems. It measures the distance between two distinct points or two vectors [48].

This study relied on the categorical entropy loss function due to its increased use in the literature recently and the good results it achieved.

2.3.4 Optimization Algorithms

Optimization algorithms in deep learning are techniques used to minimize the error or loss function during the training of neural networks. The main goal is to find the optimal set of parameters (weights and biases) that allows the model to make accurate predictions on new, unseen data. The main types of optimization methods in machine learning are batch or deterministic gradient methods, which handle all training instances in a big batch at the same time, and stochastic methods, which work with only one instance at a time and mini-batch which takes a specific number of the training instances at a time [38].

Additionally, there are two categories of optimization algorithms: algorithms of adaptive learning and algorithms with constant learning rates. In the first group, the

learning rate η is chosen manually. The task of choosing the learning rate in this type of algorithm is rather difficult. When choosing a relatively small learning rate, the learning process is slowed, and the training time becomes too large. While choosing a relatively large learning rate, can lead to instability in the loss value around the minimum value, and this hinders the convergence process. While the algorithms of the second group, they do not need to set the learning rate manually, they have a heuristic approach that takes care of adjusting the learning rate value [50]. Several algorithms appeared that belong to these two categories, the most important of which are illustrates as follows :

a) Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the Constant Learning Rate Algorithms that try to update the network parameters more repeatedly. In this algorithm, the network parameters are modified on each training sample after the loss calculation. Therefore, if the dataset contains 900 samples, the network parameters will be modified 900 times in one cycle of the dataset. The SGD is used to update parameters in a neural network [51].

b) Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) is considered one of the adaptive learning rate optimization algorithms. It measures individual learning rates for various parameters. Adam sets the learning parameter automatically, this was done by using the first and second-moment estimation. The moment is the expectation of a random variable at the power of n [52]. The moment can be illustrated in Equation 2.6.

$$m_n = E[X^n] \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.6)$$

Where m_n is the_moment, and X^n is a random_variable. Equations 2.7 and 2.8 are used to estimate, the first and second moment of Adam.

$$\widehat{m}_t = \frac{m_t}{1-\beta_1} \quad \text{.....(2.7)}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2} \quad \text{..... (2.8)}$$

where m_t is the previous first moment and v_t is the previous second moment and they are initialized with 0 in the first step. β1 and β2 are new parameters inserted into the algorithm. They have default values of 0.9 and 0.999 respectively. After calculating the value of the first and second moments, it is used to update the network weights according to Equation 2.9 [52].

$$w_t = w_{t-1} - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} \quad \text{..... (2.9)}$$

where w is network weights, η is Stepsize, and $\epsilon = 10^{-8}$

The Adam optimizer was used in this thesis due to its frequent use in previous studies in the image captioning field [17], [20].

2.3.5 Convolution Neural Network

Convolution Neural Network (CNN) is a type of deep discriminative architecture that has been demonstrated to be effective in processing Two Dimension (2D) data such as images and videos, with a grid-like layout. The architecture of CNNs is based on brain cortical architecture in animals. During the 1960s, CNNs is the first successful architecture for deep learning because of the excellent training of the hierarchical layers. The CNN architecture takes advantage of spatial relationships to reduce the parameter number of the network, which improves performance when typical back propagation algorithms are used. The growth of computation techniques

and GPUs-accelerated, have been used to train CNNs more effectively. The CNN performs two functions, the extraction function then the prediction function. Each function used certain layers to achieve a particular purpose [9].

a) Convolution Layer

The core part of the CNN architecture that handles feature extraction is the convolution layer. Typically, this layer combines linear and nonlinear operations, such as the activation function and convolution operation, to extract features [53].

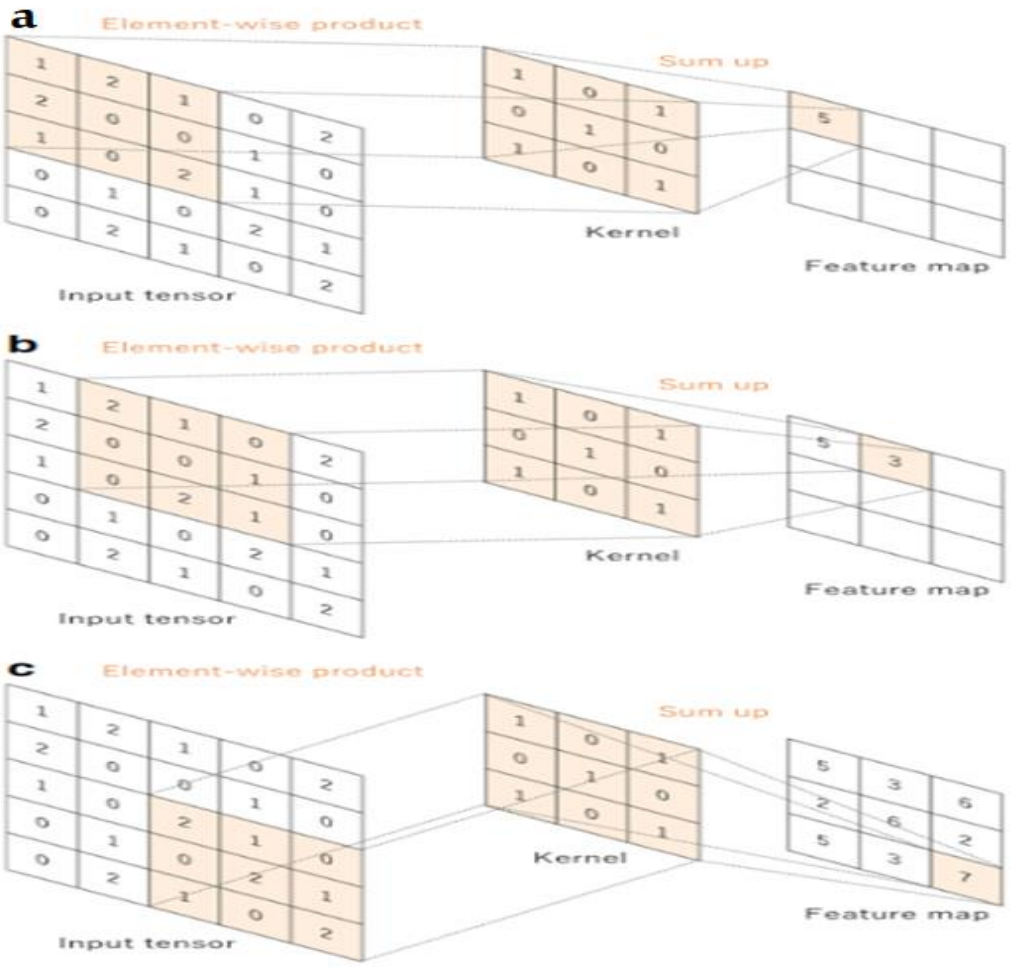


Figure 2.4: An example of convolution operation [54]

A small array of numbers known as a kernel is applied across the input, which is an array of numbers known as a tensor, in a specialized kind of linear operation

known as convolution, which is used for feature extraction. A feature map, as shown in Figure 2.4 a–c, is the result of multiplying each element of the kernel by the input tensor element-wise at each location of the tensor and adding the results to generate the output value at the corresponding position of the output tensor. As a result, different kernels can be thought of as distinct feature extractors. This process is repeated, applying several kernels to create an arbitrary number of feature maps that reflect various aspects of the input tensors. The convolution operation is defined by two primary hyperparameters: the size and the number of kernels. Usually 3 × 3, the former can also be 5 × 5 or 7 x 7. The depth of the output feature maps is decided by the latter, which is arbitrary [54].

b) Pooling Layers

Pooling layers are used to down sample the spatial dimensions of feature maps, reducing computational complexity and controlling overfitting. Common pooling operations include max-pooling and average-pooling, which select the maximum or average value from a local region, respectively. Pooling layers help preserve the most essential information while reducing the dimensionality of the feature maps [55].

c) Fully Connected Layer

Fully Connected layers (FC layers), also known as Dense Layers or Fully Connected Neural Networks, are a fundamental component in many artificial neural network architectures, particularly in deep learning. They play a crucial role in processing and transforming input data through a series of interconnected neurons. In an FC layer, each neuron is connected to every neuron in the previous layer. This means that the output of each neuron in the FC layer is a weighted sum of the inputs it receives from the previous layer, followed by the application of an activation

function (See Figure 2.5) [37]. Mathematically, the output of a neuron in an FC layer can be expressed based on Equation 2.10.

$$\mathcal{Y}_i = f\left(\sum_{j=1}^{N} \mathcal{W}_{ij}\mathcal{X}_i + b_i\right) \ldots\ldots\ldots\ldots (2.10)$$

where $\mathcal{Y}_i$ is the output of the ith neuron in the FC layer, f is the activation function (e.g., ReLU, sigmoid, or tanh), $\mathcal{W}_{ij}$ represents the weight of the connection between the jth neuron in the previous layer and the ith neuron in the FC layer, $\mathcal{X}_i$ is the output of the jth neuron in the previous layer, and b_i is the bias term for the ith neuron in the FC layer.

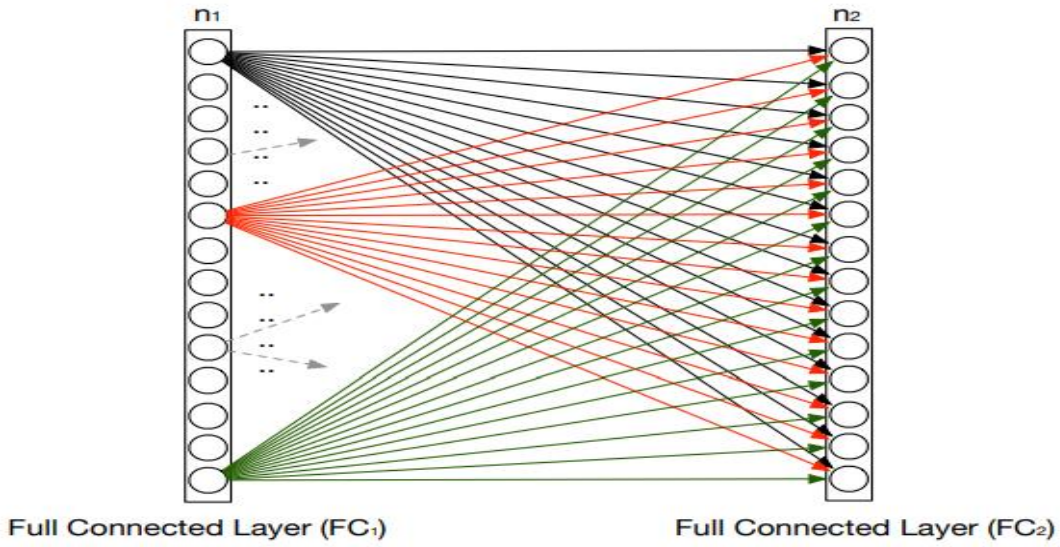


Figure 2.5:FC layers example with two layers [56]

FC layers are typically form the final layers of the network, providing the network's output. Before the introduction of FC layers, convolutional layers and other specialized layer types are commonly used for feature extraction in tasks like image recognition, while FC layers are used for making final predictions or decisions based on those extracted features [56].

d) Dropout Layers

Dropout layers are a regularization technique used to prevent overfitting. During training, dropout randomly deactivates a fraction of neurons, forcing the network to rely on different pathways for information. This helps improve the model's generalization to unseen data [57].

e) Normalization Layers

Normalization layers, such as Batch Normalization, are used to stabilize and accelerate training. They normalize the activations within a layer by adjusting their mean and variance, reducing internal covariate shift [58].

2.3.6 Pre-trained Models for Image Classification

Pre-trained models for image classification are deep learning models that have been trained on large datasets containing millions of images to learn to recognize various objects and patterns within them. These models are a type of transfer learning, where a model trained on a large dataset for one task can be fine-tuned or used as a feature extractor for a different but related task, such as image classification. Some popular pre-trained models for image classification are: AlexNet [59], VGG, Inception [60], ResNet [61], DenseNet [62], MobileNet [63], and EfficientNet [64]. These pre-trained models are often trained on huge datasets like ImageNet, which contains millions of labeled images across thousands of categories. Once trained, they can be fine-tuned on smaller datasets for specific image classification tasks, which can save a lot of time and computational resources compared to training a model from scratch [65]. This thesis used the second version of EfficientNet (i.e. EfficientNetV2) due to its speed and good accuracy.

a) EfficientNet

EfficientNet, developed by Google researchers in 2019, is a family of deep CNN that aim to achieve top-level performance in computer vision tasks while keeping computational requirements minimal. The central concept behind EfficientNet revolves around optimizing both the model's architecture and its scaling employing a balanced and efficient approach to scale these dimensions. Unlike traditional model scaling methods that typically involve increasing network depth, width, or resolution, which often results in a significant rise in computational costs [66].

EfficientNet adopts a different strategy by employing a special compound scaling method that modifies the network's depth, width, and resolution all at once, under the control of a single parameter. Finding the perfect ratio of model size to performance is made easier with this streamlined approach. In order to find the best scaling factors for different tasks, researchers combined neural architecture search with grid search. As shown in Figure 2.6, this method has produced a family of models varying from EfficientNet-B0 to EfficientNet-B7 and offer a wide range of trade-offs between model size and accuracy. While larger variations perform well in difficult tasks like segmentation, object detection, and image classification, smaller variants are best suited for applications requiring low processing ability [64].

Due to its ability to provide excellent outcomes with comparatively low processing requirements, EfficientNet has gained a lot of interest in the computer vision field and is useful in a variety of real-world applications and deployment settings. In order to make use of the generalization characteristics of these models, researchers and practitioners frequently utilize pre-trained versions of EfficientNet for transfer learning on their specific tasks [67], [68].

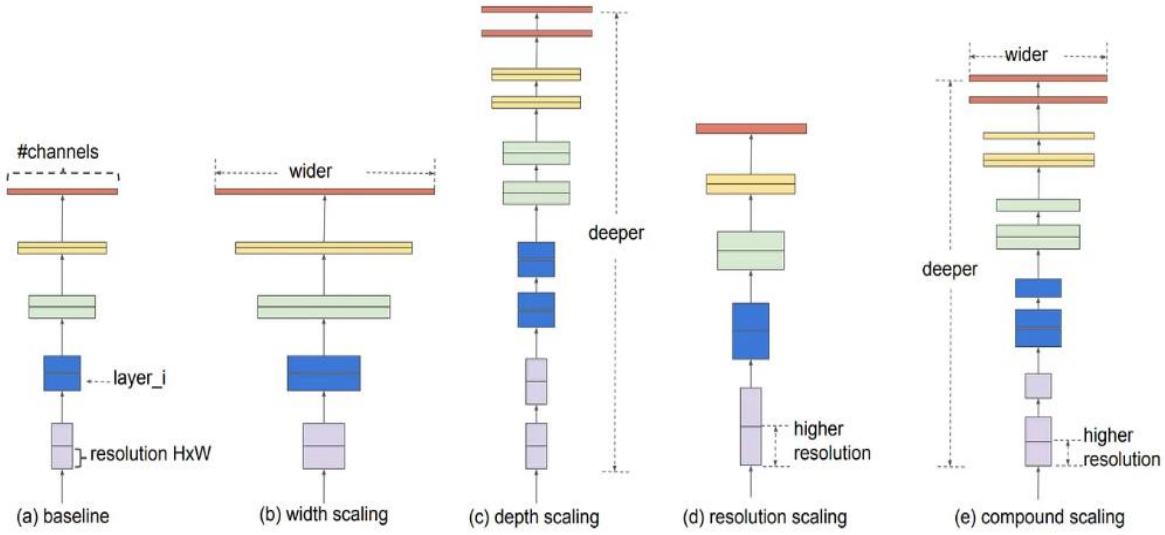
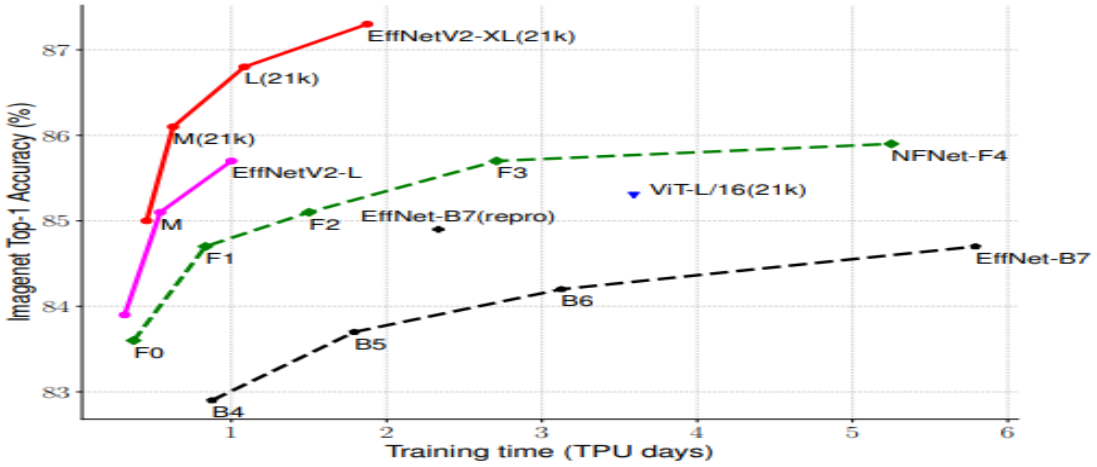


Figure 2.6: Scaling the depth, width, and image resolution to create different variations of the EfficientNet model [64]

b) EfficientNetV2

EfficientNetV2 goes one step further than EfficientNet to increase training speed and parameter efficiency. This network is generated by using a combination of scaling (width, depth, resolution) and neural architecture search. The main goal is to optimize training speed and parameter efficiency. In the end, the authors obtained the EfficientNetV2 architecture which is much faster than previous and newer state of the art models [69], as shown in Figure 2.7.

Architecturally, the primary differences lie in EfficientNetV2 strategic utilization of both MBConv and the recently incorporated fused-MBConv primarily in the initial layers. The primary distinction in the structures of MBConv and Fused-MBConv is in the final two blocks. MBConv employs a depthwise convolution (3x3) followed by a 1x1 convolutional layer, whereas Fused-MBConv simplifies this by combining these two layers into a single 3x3 convolutional layer [70], as depicted in Figure 2.8.



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Figure 2.7: Training and Parameter efficiency of the EfficientNetV2 model compared with other state of the art models [21]

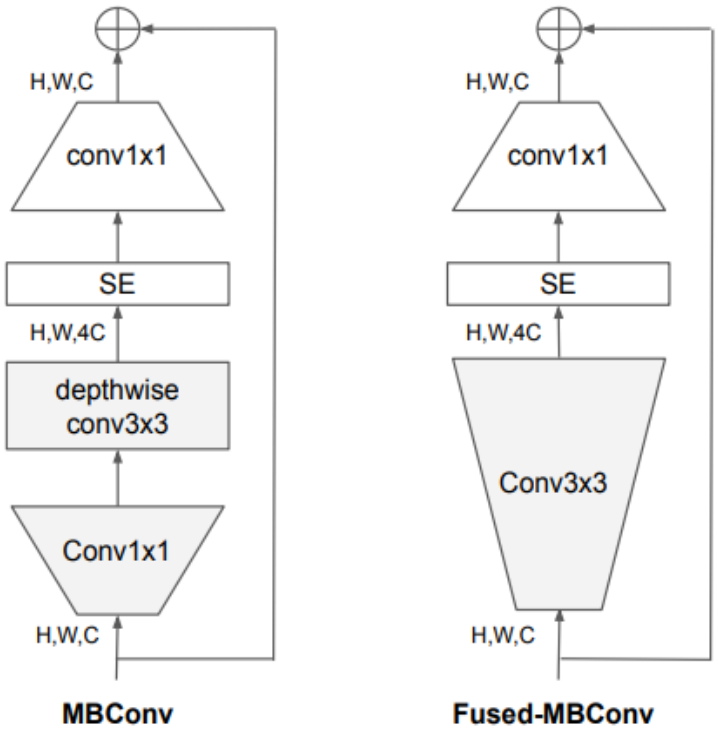
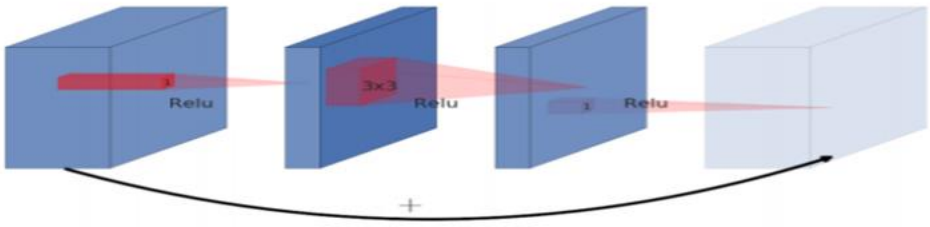
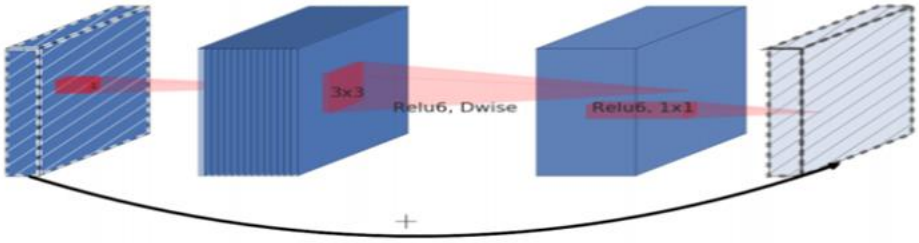


Figure 2.8:MBConv and Fused-MBConv Structure [21]

The MBConv Block, also known as an Inverted Residual Block, is a unique type of residual block employed in image models to enhance efficiency. It is originally introduced in the context of the MobileNetV2 CNN architecture, it has found applications in various mobile-optimized CNNs [71]. Unlike the traditional Residual Block, which is characterized by a wide-to-narrow-to-wide structure of channel numbers [61], [72]. The MBConv Block takes an inverted approach, following a narrow-to-wide-to-narrow sequence. It begins by expanding the number of channels with a 1x1 convolution, followed by a 3x3 depthwise convolution, and concludes with a 1x1 convolution to compress channel numbers for input-output addition as shown in Figure 2.9.



(a) Traditional Residual Block



(b) MBConv Block

Figure 2.9:Traditional Residual Block and MBConv Block Structures [71]

EfficientNetV2 is available in seven models, which are B0 to B3 and S, M, L each one of them is offering different detection accuracy and performance. The S model is used in this thesis [69].

2.3.7 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of artificial neural network designed for processing sequences of data. Unlike traditional feedforward neural networks, which process data in a single pass from input to output, RNNs have connections that loop back on themselves, allowing them to maintain a hidden state or memory of previous inputs. This recurrent connection makes them particularly well-suited for tasks involving sequential data, such as time series prediction, NLP, speech recognition, and more [46].

RNN consists of key layers: Input, Hidden, and Output. The Input layer is responsible for receiving input data at each time step in the sequence. The Hidden layer maintains a hidden state that includes information from previous time steps in the sequence, effectively performing as a form of memory within the network. Finally, the Output layer generates predictions or classifications based on the information contained in the hidden state [38].

The mathematical operations within an RNN are typically formulated to enable the network to acquire and adjust to patterns within sequential data. Nevertheless, traditional RNNs encounter a limitation referred to as the "vanishing gradient" problem, which poses challenges in capturing extensive dependencies across sequences [73]. This limitation prompted the development of more sophisticated RNN variants, such as LSTM networks and GRU networks. These advanced models incorporate mechanisms that mitigate the vanishing gradient problem and enhance the network's ability to effectively learn long-term dependencies in sequential data [74].

a) Long Short-Term Memory

Long Short-Term Memory (LSTM) network is a special traditional RNN type. Despite the complex structure of this network, it efficiently addresses the gradient vanishing problem faced by RNN [75]. Another important benefit of LSTM is its ability to learn long-term dependencies. As shown in Figure 2.10, each LSTM neuron contains a memory cell c_t in which information can be stored. tanh () and σ () are hyperbolic tangent and sigmoid functions. In addition, LSTM consist of three gates: input gate i_t forget gate f_t and output gate o_t. The benefit of these three gates is to regulate the information flow of the memory cell [76].

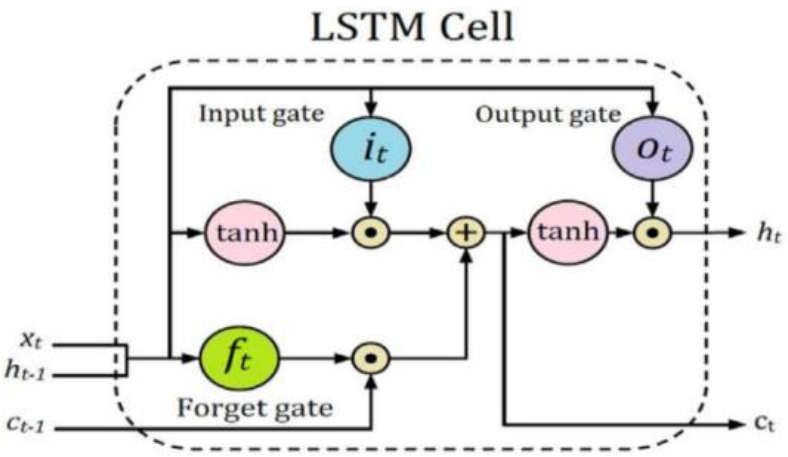


Figure 2.10:LSTM Cell [76]

The forget gate focuses on the information that needs to be forgotten and is calculated according to Equation 2.11 [77].

$$f_{(t)} = \sigma(w_{(f)}\, h_{t-1} + U_{(f)} x_t + b_{(f)}) \ldots\ldots\ldots \quad (2.11)$$

where h_t and x_t, are the hidden state vector and input vector at time t. b is the bias vectors. U is the weight matrix for the input vector, and W is the weight matrix for the hidden state.

The input gate focuses on information that are needed to be stored and is calculated according to Equations 2.12, 2.13, and 2.14 [77].

$$i_{(t)} = \sigma(w_{(i)}\, h_{t-1} + U_{(i)} x_t + b_{(i)}) \ldots\ldots\ldots\ldots\ldots \quad (2.12)$$

$$\hat{c}_{(t)} = tanh(w_{(i)}\, h_{t-1} + U_{(i)} x_t + b_{(i)}) \ldots\ldots\ldots\ldots \quad (2.13)$$

$$c_{(t)} = f_{(t)} \odot\ c_{t-1} + i_{(t)} \odot \hat{c}_{(t)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.14)$$

Finally, the output gate focuses on which parts need to be outputted in the cell state and is calculated according to Equations 2.15 and 2.16 [77].

$$o_{(t)} = \sigma(W_{(o)}\, h_{t-1} + U_{(o)} x_t + b_{(o)}) \ldots\ldots\ldots\ldots\ldots \quad (2.15)$$

$$h_{(t)} = o_{(t)} \odot\ \tanh\,(\, c_{(t)}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.16)$$

The future context and preceding context are captured by combining backward ($\overleftarrow{h}_t$) and forward ($\vec{h}_t$) hidden layers.

b) Gated Recurrent Unit

Gated Recurrent Unit (GRU) network is a simpler version of LSTM distinguished by its capability to memorize short and long sequences. As shown in Figure 2.11, GRU has two gates: an update gate (z_t) and a reset gate (r_t). The update gate is a mix between the input gate and the forget gate that focuses on transferring information to the current state, whereas the reset gate controls whether or not the hidden state is ignored [78].

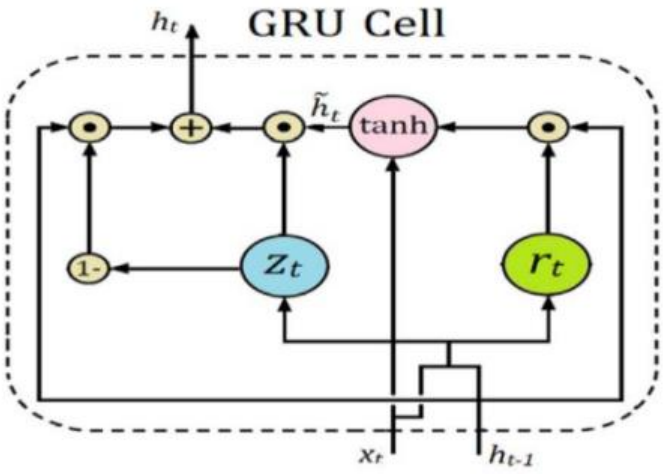


Figure 2.11:GRU Cell [75]

2.3.8 Encoder-Decoder Architecture

The Encoder-Decoder (ED) architecture represents a foundational neural network framework widely applied in a range of machine learning and deep learning tasks, notably in domains such as NLP and computer vision. This architecture comprises two essential elements: the encoder and the decoder, each fulfilling specific roles in processing and creating data sequences. It has found utility across various applications, including machine translation [79], text summarization, image captioning [3], and more.

The encoder assumes the role of the initial component within the architecture, primarily tasked with capturing meaningful representations (often referred to as embeddings or context) from the input data, which can encompass text, images, audio, or other structured data. In the domain of NLP, the encoder processes input sequences, such as sentences in a source language, and transforms them into fixed-dimensional vectors or sequences of vectors while preserving relevant information. Notably, RNN and their derivatives (e.g., LSTM and GRU) are widely employed encoders in NLP, alongside transformer-based architectures such as BERT and GPT-

3. Transformers, in particular, have garnered significant attention due to their efficient handling of long-range dependencies [5].

The decoder, on the other hand, constitutes the second component in this framework and is primarily responsible for generating output sequences based on the information encoded by the encoder. In the context of machine translation, for example, the decoder takes the encoded representation of the source sentence and produces the corresponding translated sentence in the target language. Like the encoder, the decoder can also leverage recurrent or transformer-based architectures [80].

a) Encoder-Decoder Architecture of Machine Translation

The process of automatically translating text or sequences from one language to another, known as machine translation, often utilizes the ED architecture in deep learning. This architectural approach is notably favored in the context of Neural Machine Translation (NMT) systems. It comprises two primary components: an encoder and a decoder, working in pairs to facilitate the translation of text as shown in Figure 2.12 [79].

The encoder's primary role is to handle the input text in the source language. It takes the source sentence and transforms it into a fixed-length representation known as a context vector or thought vector. This context vector captures the crucial information from the source sentence, which the decoder subsequently utilizes to generate the target translation. Typically, the encoder is constructed using RNN, LSTM, or more contemporary transformer-based models [1].

The decoder takes the context vector generated by the encoder and produces the translated sentence in the target language. It processes the context vector step by step

to generate each word or subword of the translation. Like the encoder, the decoder can also be implemented using RNN, LSTM, or transformers [81].

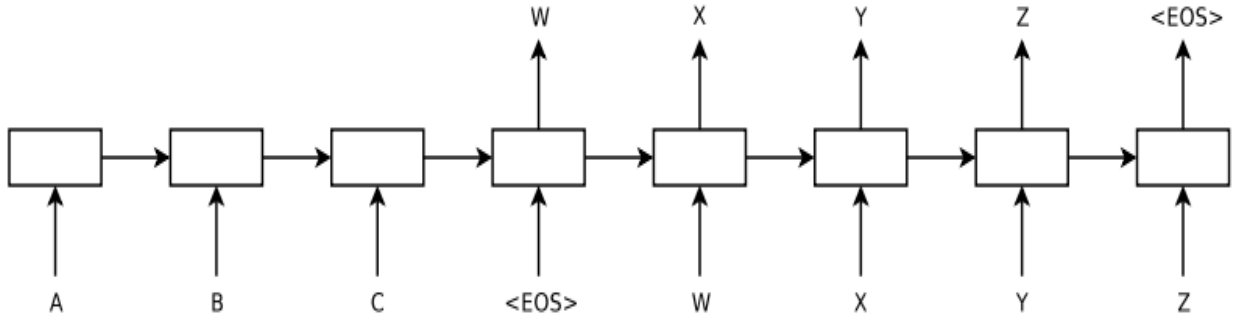


Figure 2.12:Encoder-Decoder Architecture of machine translation [5]

b) Encoder-Decoder Architecture of Image Captioning

The ED architecture for image captioning is a deep learning approach employed to produce natural language descriptions (captions) for images. This method integrates two neural networks, an encoder and a decoder, to accomplish this task. The encoder, which is typically implemented as a CNN, extracts meaningful features from the input image. In contrast, the decoder, usually implemented using a LSTM network, generates a coherent and contextually relevant caption based on these extracted features as depicted in Figure 2.13. This architecture is inspired by machine translation models like those used in NMT, where the encoder processes the source language and the decoder generates the target language [3].

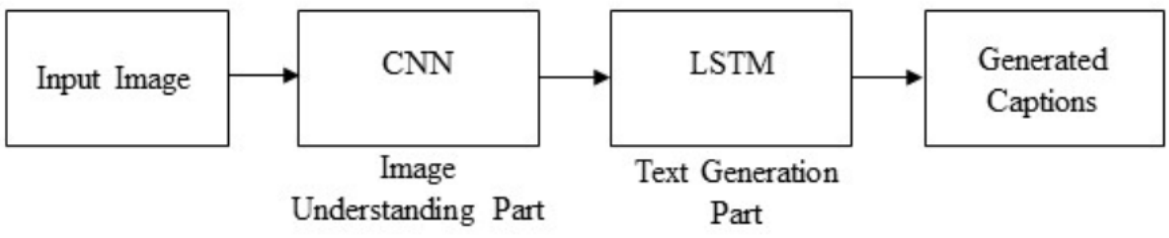


Figure 2.13:Encoder-Decoder architecture of image captioning [1]

The encoder network is typically a CNN, such as a pre-trained model like VGG, ResNet, or Inception. Its purpose is to process the input image and extract high-level feature representations. These features capture various aspects of the image, such as objects, shapes, and spatial relationships, in a fixed-length vector representation. These fixed-length feature vectors are passed on to the decoder for generating captions [1].

The decoder network is usually an RNN, specifically a type of RNN called a LSTM or a GRU. It takes the feature vectors from the encoder and generates captions word by word. At each time step, the decoder produces a probability distribution over the vocabulary of possible words. This distribution is used to predict the next word in the caption. The decoder's hidden state is updated at each time step based on the previously generated words and the features from the encoder. The process continues until an end-of-sequence token is generated [82].

2.3.9 Attention Mechanism

Attention mechanisms in deep learning are crucial in various neural network architectures to focus on specific parts of input data or context while making predictions. These mechanisms have proven to be especially effective in NLP tasks and have found applications in computer vision such as machine translation, image captioning, and other domains [1]. Some common types of attention mechanisms in deep learning are: Transformer Attention [81], Bahdanau Attention (Additive Attention) [79], Luong Attention (Multiplicative Attention) [83], and Hard Attention and Reinforcement Learning-based Attention. This thesis used Bahdanau Attention because it has become a useful component in many state of the art image captioning models, improving their performance in describing visual content [84].

a) Bahdanau Attention (Additive Attention)

Bahdanau attention is commonly used in sequence-to-sequence models, such as the ED architecture. It computes a weighted sum of the encoder's hidden states, considering each hidden state's compatibility with the current decoder state. In the context of NMT, they have an encoder that reads the source sentence denoted as $\{X_1, X_2, \dots, X_T\}$ and produces a sequence of hidden states, denoted as $\{h_1, h_2, \dots, h_T\}$ as depicted in Figure 2.14, where T is the length of the source sentence. They also have a decoder that generates the target sentence one token Y_t at a time. At each decoding step t, the decoder has a hidden state denoted as s_t [79].

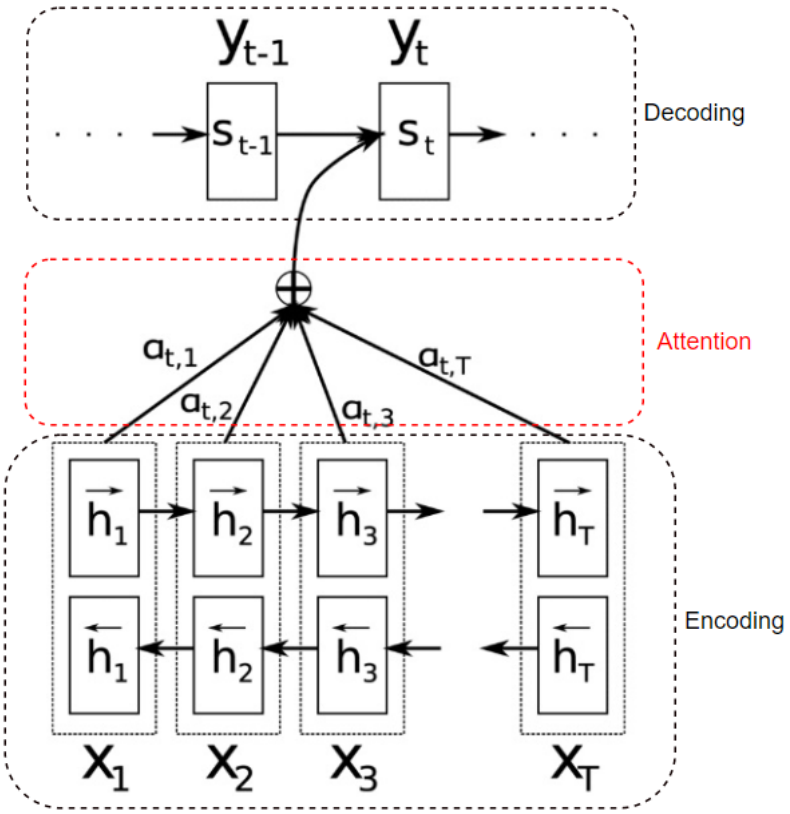


Figure 2.14:The Bahdanau architecture [79]

Bahdanau Attention computes a context vector c_t for each decoding step t by considering the source sequence's hidden states and the current decoder hidden state.

The attention score $e_{t,i}$ for each source hidden state h_i with respect to the decoder hidden state s_t is calculated as in Equation 2.17 [85]:

$$e_{t,i} = v^T \times \tanh\,(W_a \times h_i + W_s \times s_t) \ldots\ldots\ldots \text{(2.17)}$$

where W_a and W_s are learnable weight matrices. v^T is a learnable weight vector.

These attention scores are computed for each source hidden state h_i with respect to the current decoder state s_t, resulting in a vector of scores$[e_1, e_2, \ldots, e_T]$. Next, these scores are transformed into attention weights α_i using the softmax function σ show in Equation 2.18 [20]:

$$\alpha_i = \sigma(e_i) \ldots\ldots\ldots\ldots\ldots \text{(2.18)}$$

where α_i represents the attention weight for source state h_i.

Finally, the context vector c_t is computed as a weighted sum of the source hidden states, where each weight is determined by α_i this computed based on Equation 2.19 [79]:

$$c_t = \sum_i(\alpha_i \times h_i) \ldots\ldots\ldots \text{(2.19)}$$

This context vector c_t is then used as an additional input to the decoder to generate the next token in the target sequence [85].

b) Luong Attention (Multiplicative Attention)

Luong attention is another variant of attention used in sequence-to-sequence models, similar to Bahdanau attention. Instead of using a feedforward network for scoring, Luong attention uses a simple dot product between the decoder state and encoder hidden states [83].

2.4 Object Detection

Object detection is a computer vision task that involves identifying and locating objects of interest within an image or video frame. It uses a neural network to recognize items in an image and to highlight them with bounding boxes. Therefore, the term "object detection" refers to the location and identification of objects within an image that fall into a predefined set of classes. The wide application of tasks like recognition, detection, and localization in practical situations makes object detection a key area in computer vision. Generally, there are two methods for detecting objects, which are: Two-stage object detection, and One-stage object detection [86].

2.4.1 Two Stage Object Detection

"Two-stage object detection" refers to the applying of techniques that divide the object detection problem statement into two steps:

- Detecting potential object regions.
- Classifying the image into object classes in those regions.

Region Proposal Network, which proposes possible object containing regions of interest, is frequently used in common two step algorithms like Fast-RCNN and Faster-RCNN [87].

2.4.2 One Stage Object Detection

One-stage detectors expect all bounding boxes in a single neural network pass. It is significantly quicker and more suited for portable devices. The most popular one-stage object detectors include DetectNet, Single Shot MultiBox Detector (SSD), SqueezeDet, and You Only Look Once (YOLO). Two-stage object detection models achieve the highest accuracy rates, which are often slower. In contrast, one-stage object detection models are much faster than two-stage object detectors but achieve lower accuracy rates. Even while two-stage object detection models produce

accurate findings with a high mean Average Precision, they require a lot of iterations within the same image, which prevents real-time detection and slows down the algorithm detection speed [88].

YOLO is an algorithm that in real-time detects and identifies various elements in images. The object identification process in YOLO, which is carried out as a regression problem, offers the class probabilities of the identified images. CNN are used by the YOLO method to recognize objects in real-time. The method just needs one forward propagation through a neural network to detect objects, as suggested by the name. This shows that prediction is carried out throughout the entire image using a single algorithm run. The CNN is used to forecast multiple bounding boxes and class probabilities at once. Various versions of the YOLO algorithm are available. In terms of popularity, YOLOv3, YOLOv4, and YOLOv5 are the most common three. YOLOv6 and YOLOv7, two recent updates, each provide a different level of performance and detection accuracy [89]. The most recent YOLOv7 version was used in this thesis.

YOLOv7 is the most recent official version of YOLO, developed by the initial developers of the YOLO architecture. In July 2020, it was presented to the YOLO family. Has a number of upgrades over the versions that followed. The usage of anchor boxes is among the primary enhancements. Anchor boxes are a group of pre-made boxes with various aspect ratios that are meant to identify variously shaped objects. With nine anchor boxes, YOLOv7 is able to recognize a greater variety of object shapes and dimensions than earlier iterations, which contributes to reducing the amount of false positives [90].

Furthermore, YOLOv7 employs a brand-new loss function known as "focal loss". The standard cross-entropy loss function, which is known to be less successful at identifying small objects, was utilized in earlier iterations of YOLO. Focal loss

addresses this problem by highlighting the hard examples, the objects that are difficult to detect while down weighting the loss for examples that are well-classified. Additionally, YOLOv7 displays an improved resolution than versions before it. YOLOv3 utilizes a resolution of 416 by 416 pixels, whereas it processes images at a resolution of 608 by 608 pixels, so it is capable of identifying small objects and has a higher overall accuracy due to its higher resolution [91].

The essential architecture of YOLOv7 is constructed on Efficient Layer Aggregation Network (ELAN) and is adapted from YOLOv4, Scaled YOLOv4, and YOLO-R. In order for deeper networks to successfully converge and learn, ELAN takes into account the architecture of an efficient network by managing the shortest and longest gradient path as shown in Figure 2.15 (a). YOLOv7 introduces modifications to the ELAN architecture, which is referred to as the Extended Efficient Layer Aggregation Network (E-ELAN). ELAN employs techniques such as expansion, shuffling, and cardinality merging to enhance the model's capacity for learning without disrupting the pathways for gradient flow. Regarding the architectural changes, these adjustments are limited to the computational block, while the transition layer maintains the same structure as ELAN, as depicted in Figure 2.15 (b) [90].

Model scaling is a critical concept, enabling adjustments in model width, depth, and image resolution. When scaling a model, the selection of layers to include or exclude is pivotal. Also, scaling the number of channels within the model's architecture relates to its width. Scaling factors for both width and depth are defined in the model architecture files [89]. In YOLOv7, the architectural structure is intertwined with other layers. Consequently, when executing scaling with regard to the depth parameter within a computational block, it becomes critical to compute the resulting alterations in output kernels. This, in turn, necessitates a corresponding

adjustment in the width component, mirroring the changes calculated in the kernels. This comprehensive scaling strategy ensures the preservation of the architectural characteristics, design integrity, and optimally structured framework. The compound scaling procedure is visually depicted in Figure 2.16 [92].

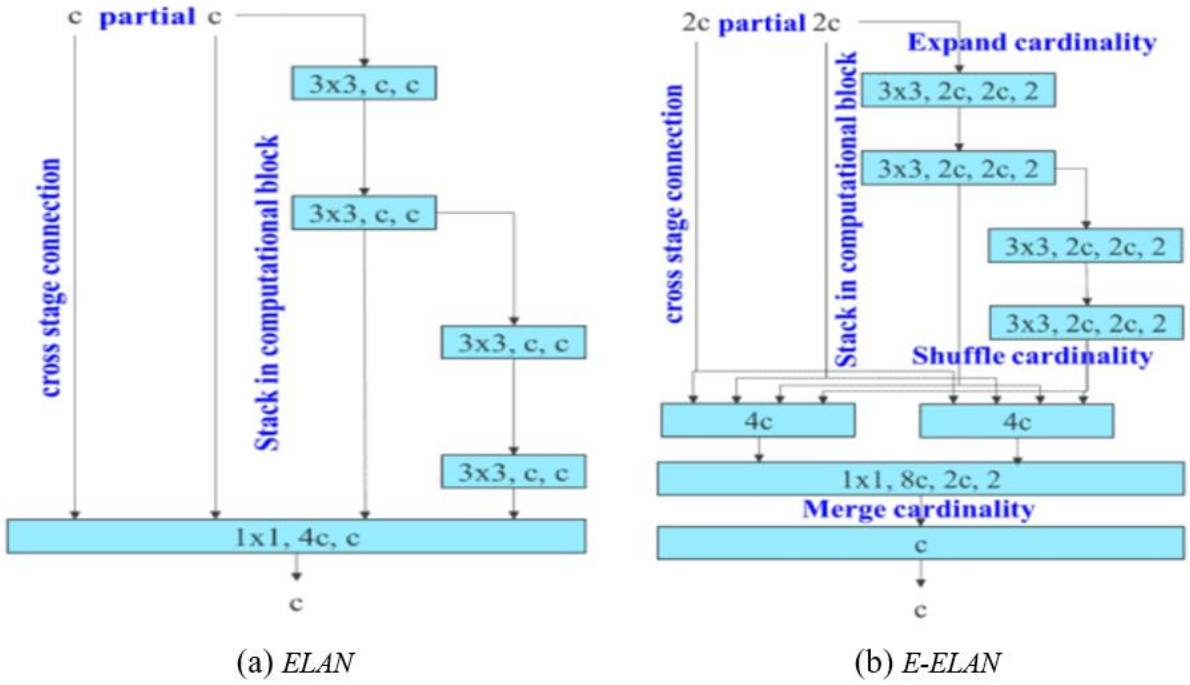


Figure 2.15:ELAN and E-ELAN architecture [90]

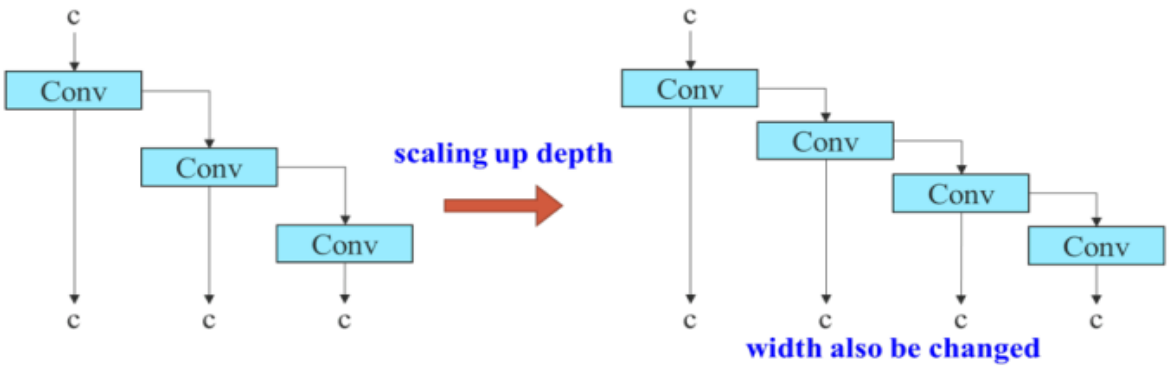


Figure 2.16:Model Scaling of YOLOv7 [90]

2.5 Text Analysis

Text analysis, also known as text mining or NLP, is the process of examining and extracting meaningful information from textual data. It involves using various techniques and tools to gain insights, discover patterns, and extract valuable knowledge from unstructured text. Text analysis can be applied to a wide range of applications, including sentiment analysis, language translation, image captioning, information retrieval, and more [93].

2.5.1 Text Preprocessing

In this thesis, many text preprocessing techniques are utilized, such as:

a) Tokenization

Is splitting the text into individual words or tokens. This process makes it easier to analyze and work with text at the word level rather than as a single string [94].

b) Lowercasing

One of the quickest and most efficient methods of text preparation is lowercasing. It is useful in cases where the dataset is not too large and can be applied to the majority of text mining and NLP challenges [95].

c) Stemming and Lemmatization

Which refers to the process of reducing the inflection of words. Stemming is a primitive heuristic procedure that chops off the ends of words in the hope of appropriately changing them into their base form. The goal of stemming is to reduce word count, match stems accurately, memory space, and save time. Moreover, lemmatization is quite similar to stemming in its overall purpose. It eliminates inflections and maps a word to its root form. It returns the changed words to their root [96].

d) Stop-Words Elimination

Stop-words are a group of terms that are frequently employed in a language. "a", "the", "is", "are", and other words are examples of stop-words in English. The purpose of stop-words is to remove keywords with low information from the text [95].

e) Removing Punctuations

Is the removal of uninformative information such as punctuation a necessary stage in text preprocessing. The method of removing punctuation is to delete punctuations that frequently appear and usually have no meaning. Punctuations help only the composition of sentences and how they should be read such as '#', '$', '!', '(', '"', '&', '%', ')', '+', '"', ',', '-', '/', '*', '.', '<', ';', '=', '@', ':', '?', '\\', '^', '>', '_', '`', '}', '[', '|', '{', ']', '~'.

2.5.2 Feature Extraction

A feature extraction procedure is used to extract attributes or significant features from a raw text in order to be prepared for feeding to machine learning or statistical methods. This method is referred to as vectorization, since it produces numerical vectors from raw text tokens. The reason is that machine learning algorithms work on numerical vectors and cannot work directly on raw text data because it includes formats that are not acceptable to these algorithms. Feature extraction methods are used in order to feed extracted features into machine learning algorithms for learning patterns that may be applied to new data points [97].

There are several common feature extraction techniques such as Word2Vec, BERT, and GloVe whereas in neural networks for image captioning, a learnable embedding layer is often used as the first layer in the model. This layer takes word indices as input and converts them into word vectors. These word vectors are learned during

the training process and are typically fine-tuned along with other model parameters [98]. As a result, this layer was used in this thesis.

a) Word2Vec

In NLP, Word2Vec is employed, its force lies in its ability to cluster diverse vectors with related words. Word2Vec can produce accurate meaning estimations for words based on their frequency in text, given that the dataset is sufficiently large. These estimates produce word associations with other words in the corpus [31].

b) BERT

A pre-trained language model is called BERT. It achieves state of the art performance for some NLP issues by leveraging transformer model architecture. The two approaches that can be employed with the BERT model are feature-based and fine-tuning-based. BERT uses a trained model to represent text data as fixed feature vectors in the feature-based process. In contrast to fine-tuning, which involves using a model that has already been trained for a certain task and adjusting it to perform a second identical task [33].

c) GloVe

GloVe represents a technique that integrates two distinct methodologies. The initial approach is count-based, while the other aligns with direct prediction methods, similar to word2vec. In contrast to word2vec, which relies primarily on local word contexts within proximity, the GloVe algorithm takes into account word co-occurrence patterns and global statistics to establish semantic relationships among words in the corpus [32].

d) Embedding Layer

An embedding layer is a fundamental component in many NLP and machine learning models, especially neural networks (i.e. CNN, RNN), that are designed to

work with text data. Its primary purpose is to convert discrete categorical data, such as words or tokens in a text, into continuous vectors (embeddings) that can be used as input features for machine learning models [99].

Practically, the embedding layer takes as input a sequence of discrete symbols, typically words or tokens, from a text corpus. Each unique word in the input text is associated with an integer index that represents its position in a predefined vocabulary. For example, "cat" might be assigned index 5, "dog" index 10, and so on. The core function of the embedding layer is to map each word's integer index to a continuous vector representation, known as a word embedding. These embeddings are learned during the training process and aim to capture semantic relationships between words. Word embeddings have a fixed dimensionality (e.g., 50, 100, 300 dimensions), which is a hyperparameter defined when creating the embedding layer [98].

Internally, the embedding layer maintains an embedding matrix or lookup table. This matrix has dimensions (Vocabulary Size x Embedding Dimension), where the rows correspond to each word in the vocabulary, and the columns represent the dimensions of the embeddings. During training, the model updates the values in this matrix to optimize the embeddings for the specific task. The objective is to position similar words closer together in the embedding space [99].

2.6 Evaluation Measures

When developing a machine learning model, it is critical to evaluate performance and efficiency. For the machine learning model to be dependable, an assessment tool that is appropriate for the nature of the model's work must be chosen. When evaluating machine learning models, it is common to utilize more than one scale to

verify that the model is correctly evaluated. The primary machine learning assessment metrics that are widely used in the image captioning field are:

a) BLEU

Bilingual Evaluation Understudy (BLEU) is a widely used metric for evaluating the quality of machine-generated translations or text. It measures the similarity between the generated text and one or more reference texts (human-generated) using n-gram precision. It is calculated based on the precision of n-grams (1-gram, 2-gram, 3-gram, etc.) between the generated text and reference texts. The BLEU score is computed as in Equation 2.20 [100]:

$$BLEU = BP.e^{\sum_{n=1}^{N}(w_n * \log Pn)} \text{..... (2.20)}$$

where Pn is the modified precision for n gram, w_n is weight between 0 and 1, BP is the brevity penalty to penalize short machine translations, and N is the maximum n-gram size considered (typically 4).

b) METEOR

Metric for Evaluation of Translation with Explicit Ordering (METEOR) is an automatic evaluation metric used primarily in machine translation. It combines precision, recall, stemming, synonymy matching, and word order into a single metric as shown in Equation 2.21 [101].

$$METEOR = \frac{10*Precision*Recall}{Recall+9*Precision} \text{..... (2.21)}$$

c) CIDEr

Consensus-Based Image Description Evaluation (CIDEr) is a metric specifically designed for evaluating image captions. It focuses on grammaticality and saliency and measures the consensus between generated captions and reference captions

using a weighted version of cosine similarity. It considers not only individual words but also the consensus between multiple references. The CIDEr score is computed as in Equation 2.22 [102].

$$CIDEr(x_i, Y_i) = \sum_{n=1}^{N} w_n \; CIDEr_n(x_i, Y_i) \ldots\ldots\ldots\ldots (2.22)$$

where Y_i denotes the caption generated by a machine, x_i is the original caption.

d) ROUGE-L

Recall-Oriented Understudy for Gisting Evaluation-Longest Common Subsequence (ROUGE-L) is a metric used for evaluating the quality of machine-generated text, particularly for tasks like text summarization, and image captioning. It measures the longest common subsequence (LCS) between the generated text and reference text as follows. The ROUGE-L score is computed as in Equation 2.23 [103]:

$$ROUGH - L = \frac{(1+\beta^2) * R_{lcs} * P_{lcs}}{R_{lcs} + \beta^2 * P_{lcs}} \ldots\ldots\ldots\ldots (2.23)$$

where R_{lcs} and P_{lcs} denote to LCS-based Precision and Recall.

e) SPICE

Semantic Propositional Image Caption Evaluation (SPICE) is a metric designed specifically for image captioning tasks, where the quality of generated captions for images is evaluated based on the presence of semantic content and relationships. It computes a score based on the precision of extracted semantic propositions as shown in Equation 2.24 [104]:

$$SPICE(x, Y) = \frac{2 * Precision(x,Y) * Recall(x, Y)}{Precision(x,Y) + Recall(x, Y)} \ldots\ldots\ldots\ldots\ldots (2.24)$$

where Y denotes the caption generated by a machine, x is the original caption.

Chapter Three
The Proposed System

Chapter Three: The Proposed System

3.1 Overview

This chapter provides an overview of the practical aspects of the thesis, outlining the key phases and methodology employed in constructing the proposed system. It encompasses several stages aimed at achieving the task of transforming input images into specific descriptions. This involves initially comprehending the image's content and subsequently generating a descriptive caption that aligns with the semantic information present in the image.

3.2 The Proposed System

The proposed system employs Encoder-Decoder (ED) architecture attention-guided. It comprises three phases as shown in Figure 3.1: Preprocessing, Features extraction (Encoding), and Features decoding. In the first phase, both image and caption data undergo preprocessing. In the second phase, two types of features are extracted: Generic features (deep visual features) and Specific features (object features). Additionally, we compute a new feature called Priority Factor (*pf*) and add it to features of each object. Finally, these various features are combined in the concatenation step. The third phase is a deep neural network that consists of three processing blocks: Reduction, Attention, and Language. During this phase, the input image feature set and caption vector are passed through extra layers (Dense, LSTM) to output a sequence of encoded term set that represents the final image caption.

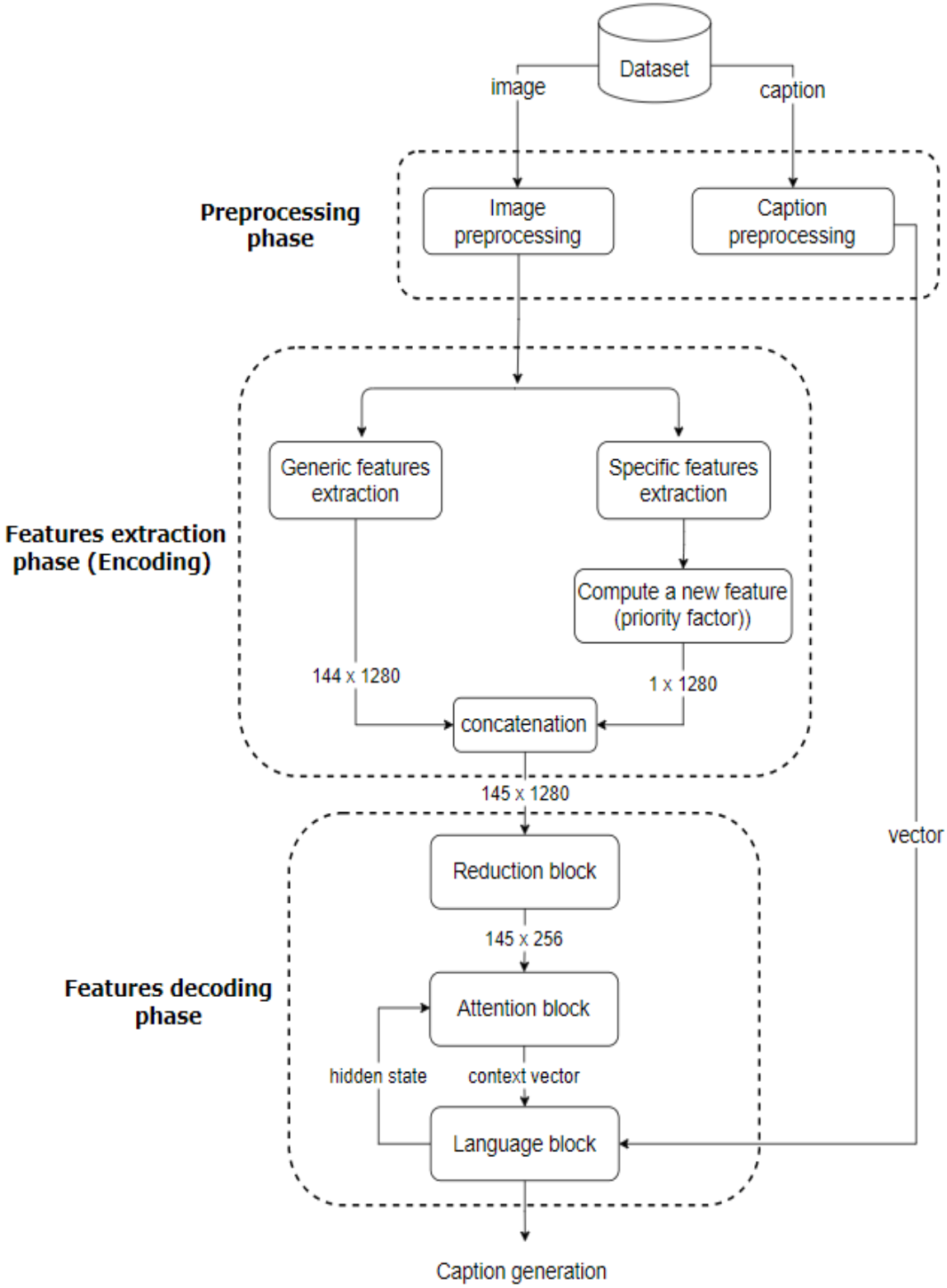


Figure 3.1: A block diagram of the proposed system

3.2.1 The Used Dataset

MSCOCO [105] dataset is often used in the image captioning literature. This dataset primarily comprises images depicting common objects prevalent in daily existence, each accompanied by five captions meticulously crafted by domain experts. The encompassed objects includes a diverse collection, ranging from animals and fruits to household articles and vehicles, among others. The dataset standard partitioning consists of 83000, 41000, and 41000 images for training, validation, and testing respectively. This thesis is based on the widely used split regime in literatures (113287 training, 5000 validation, and 5000 testing) as applied in [4], for the purpose of evaluation and comparison.

The MS COCO dataset includes two main formats: JSON and image files. The JSON format includes the following attributes:

- Images: a list of all the images in the dataset, including the file path, image id, and other metadata.
- Captions: a list of all captions for each image (5 captions for each image), including the image id, caption id.

The image files are the actual image files that correspond to the images in the JSON file. These files are typically provided in JPEG or PNG format and are used to display images in the dataset as shown in Figure 3.2.

3.2.2 The Preprocessing Phase

Preprocessing is an essential part of every system. In this section, a comprehensive description is provided regarding the preprocessing phase involving both images and captions presented in the dataset. Algorithm 3. 1 depicts the steps involved in the preprocessing of images and captions.

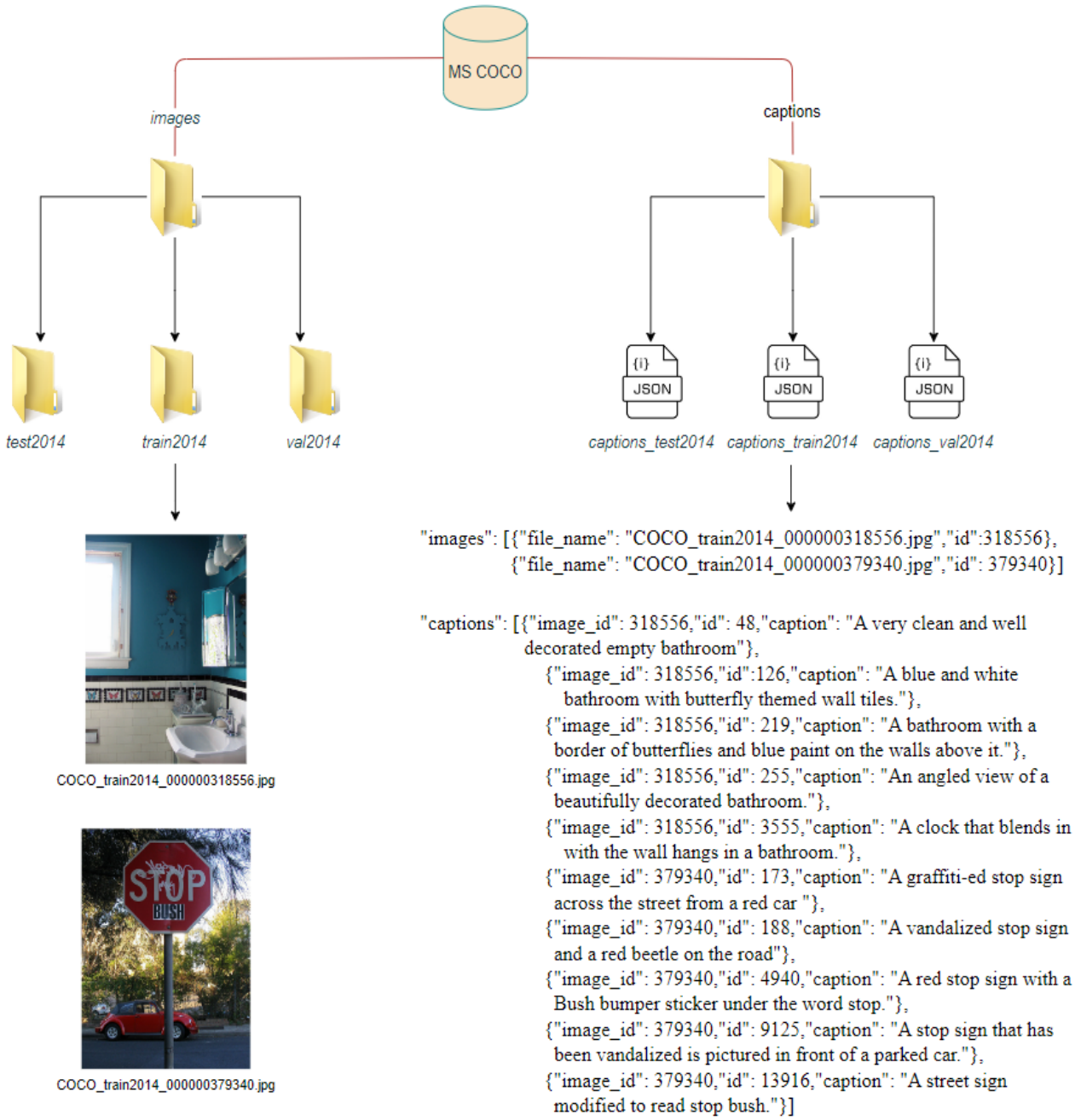


Figure 3.2:MSCOCO dataset samples

a) Image Preprocessing

Image preprocessing includes all images in the dataset. First, creating a special dictionary for all images in the captions file (JSON file). The dictionary includes “image_id” and “file_name” for each image. Thus, it is possible to exploit this dictionary in the captions fetching process corresponding to each image. The next

step is to read each image in the images file, followed by decoding image format (such as JPEG, PNG, etc.) into a 3-channels (RGB), where the output values range from 0 to 255. Finally, the diverse dimensions of images are considered. It is essential to resize the input images to the specified 384×384 dimensions as per the architectural requirements of the EfficientNetV2 CNN model, whereas the YOLOv7 model does not require this. It implicitly resizes the image through one of its layers.

b) Caption Preprocessing

The architecture requires the preprocessing workload of the caption before the workflow can take place. Firstly, each caption in the captions file ((JSON file) must be tagged with <start> and <end>. The next step involves constructing a traditional dictionary for usage later in translating captions to vectors. This is performed by tokenizing the textual content for each caption into separated words based on punctuations, special characters, and white spaces. The resulting tokens are then counted and sorted based on their frequencies. Only the tokens within the top 10,000 ranks are chosen as vocabulary to avoid overfitting according to previous works. Since sentences are of various lengths, they need to be unified through padding. The final preprocessing step involves randomly shuffling the dataset to ensure that the training process can converge quickly and eliminate any undesired bias that may arise from the model learning specific rankings of the training samples. By shuffling, the model is exposed to a diverse range of samples during training, facilitating better generalization and reducing the risk of overfitting to specific patterns or sequences.

Algorithm 3. 1: The Preprocessing Phase

Input: Dataset (DS) // The dataset includes two files
Output: List of captions vectors, List of tensor images (2D array)
Variables' definition • img: a dictionary involves information for each image • image_id_index: a dictionary involves (file name and id) for each image • annot: a dictionary involves (caption, id, and image_id)

- C: Caption (string)
- P: full image name path (string)
- L_1: a list involves a full path for each image
- L_2: a list of captions corresponding for each image
- L_3: a list of captions vectors
- L_4: a list of captions vectors with padding

Begin

Step 1: Read DS
caption file path = 'D:\\ms_coco\\captions\\captions_train2014.json'
image file path = 'D:\\ms_coco\\train2014'
caption = Read the json file (caption file path)
Step 2: Create a dictionary of images
    FOR img in captions['images'] **DO**
        image_id_index[img['id']] ⟵ img['file_name']
    End
Step 3: Adding <start> and <end> tokens for each caption and store captions and image names in lists
    FOR annot in captions['captions'] **DO**
        C ⟵ '<start> ' + annot['caption'] + ' <end>'
        ID ⟵ annot['image_id']
        P ⟵ image file path + '\\' + image_id_index [ID]
        L_1 ⟵ P
        L_2 ⟵ C
    End
Step 4: Shuffle for all captions and images in L_1 and L_2 together (correspond)
Step 5: Tokenizing for each caption in L_2
Step 6: Dictionary constructing
Step 7: L_3 ⟵ Convert each caption in L_2 into a vector
Step 8: L_4 ⟵ Add padding (zeros) for each caption in L_3
Step 9: Read each image in L_1 and convert it into three channels (RGB) as 2D array
Step 10: Resize the images

End

3.2.3 The Features Extraction Phase (Encoding)

In this section, we delineate the details of the features extraction phase. The encoder is designed to take an image as its input and generate embedded features that are the

visual attributes of the image. To achieve this the encoder exploits two types of feature sets: (1) generic features extracted from CNN based image classification task (EfficientNetV2) and (2) specific features extracted from object detection model (YOLOv7). In addition, a new feature i.e. *pf* is generated and added to features of each object. These features are combined through concatenation, resulting in the creation of a feature matrix that contains raw information.

a) Generic Features Extraction

This stage utilizes the EfficientNetV2, which is pre-trained on the ImageNet dataset, for extracting deep visual features (i.e. Generic). EfficientNetV2 is a CNN model commonly used for image classification tasks, where it first extracts features from input images and then classifies them into specific categories (e.g., dog, cat, etc.).

This thesis focuses specifically on the feature extraction part of this architecture, ignoring its classification aspect. It extracts deep visual features from the final layer of block 5 in Figure 3.3, in alignment with the latest methodologies in the field of image captioning research. These features have dimensions of (12×12×1280), where 12×12 denotes image dimensions (width × height) and 1280 represents the number of channels (filters). Subsequently, they are reshaped into (144×1280) dimensions to facilitate more efficient matrix manipulation as shown in Figure 3.3. These features carry generic spatial information about the entire image, enabling a comprehensive understanding of the objects in the image and their relationships.

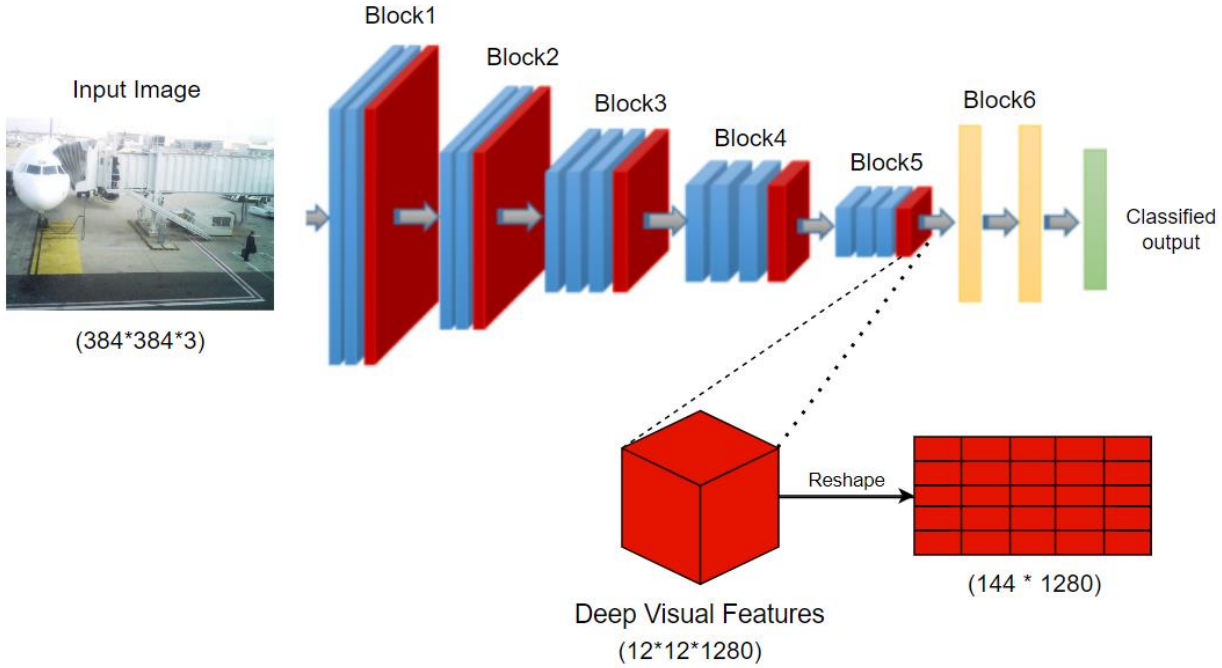


Figure 3.3: Deep visual features extraction from the EfficientNetV2 model

b) Specific Features Extraction

The proposed system in this stage exploits the object detection model to extract specific features (object features), which contain rich semantic information. It utilizes YOLOv7, which is a pre-trained model on the MSCOCO dataset due to its heightened accuracy compared to its predecessors, rendering it well-suited for detecting small objects and achieving enhanced overall precision.

The extracted features are 2D array (i.e. row and column) encapsulating object features. Each row corresponds to a bounding box delineating an object, including (b_x, b_y, b_w, b_h, p_c and c) where b_x and b_y represent coordinates of the top left point of the bounding box, b_w and b_h denote dimensions of the bounding box (i.e. width and height). Empirically, b_w and b_h represent coordinates of the down right point of the bounding box. p_c is confidence factor and c denotes class number as shown in Figure 3.4.

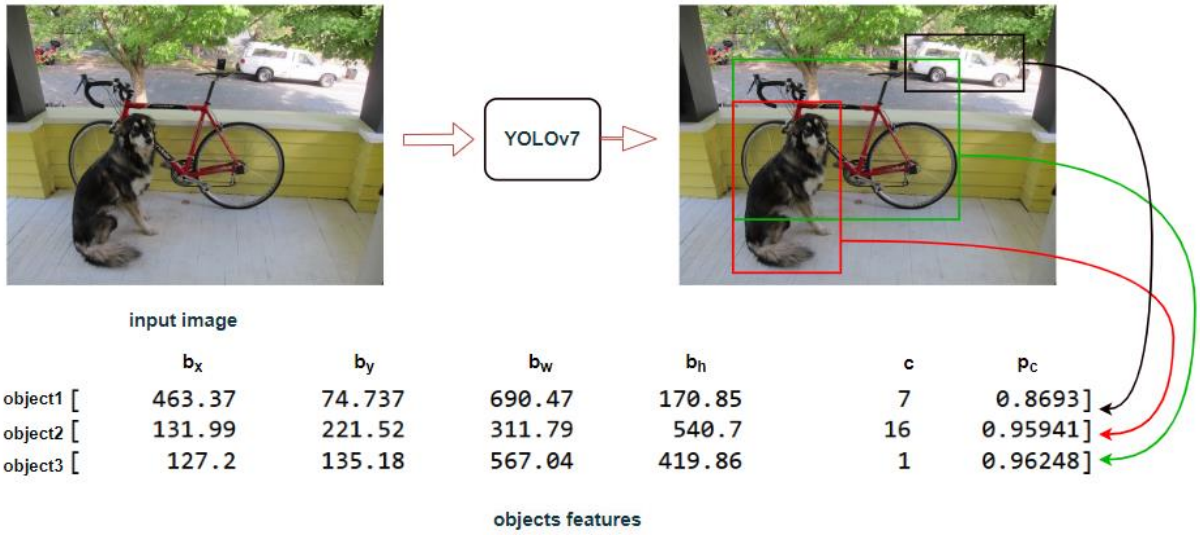


Figure 3.4: Object Features Extraction and Bounding Box Representation

c) Extracting the Priority Factor Feature

In this stage, a new feature called Priority Factor (pf) is generated to the set of features for each object in the input image. As indicated earlier, the purpose of pf is a score derived from the features of an object in order to rank its semantic prominence with respect to other objects within the image's landscape.

pf is conceptually based on the following empirical observations:

A. Prominent objects tend to be larger in terms of area with respect to their counterparts within the same image. Furthermore, such tendency is inclusive for all prominent objects.
B. A minute's reflection on photography field often yields us to readily observe that prominent objects within an arbitrary image tend to collocate either in or close to the center of the image's landscape.

Here, further explanation is provided on how these observations can be modeled and further exploited to improve the accuracy of the captioning task. Given an image

p, the set $O = \{o_0, o_1, \ldots, o_n\}$ to represent all objects within p and detected by the object detection model (YOLOv7), and an object o_i in O. Empirically, the object detection model recognizes any object as a square or rectangle, hence we can compute the width and length of each o_i using Equations 3.1 and 3.2.

$$width = x_2 - x_1 \qquad (3.1)$$

$$length = y_2 - y_1 \qquad (3.2)$$

where x_2 and x_1 represent coordinates of the top left point of the o_i bounding box. y_2 and y_1 represent coordinates of the down right point of the o_i bounding box. After that the spatial area of o_i is compute using Equation 3.3, the mean of the areas of all objects in O is compute using Equation 3.4.

$$a(o_i) = \quad width * length \qquad (3.3)$$

$$M_p = \frac{\sum_j^N a(o_j)}{N}, N > 0 \qquad (3.4)$$

where $a\ (o_i)\ is$ the area of o_i, M_p is the mean of the areas of all objects in O and N is the number of objects in O. To activate observation (A), the area ratio of o_i is readily calculated using Equation 3.5.

$$r(o_i) = \frac{a(o_i)}{M_p} \qquad (3.5)$$

By viewing p, a plane observation (B) can be activated by calculating the distance between the center of p, and the center of o_i, using a spatial distance measure. In this step, Euclidian Metric as shown in Equation 3.6 is used.

$$d(o_i) = \sqrt[2]{((x_i - x_c)^2 + (y_i - y_c)^2)} \qquad (3.6)$$

where (x_c, y_c) represent coordinates of the center point of the p. (x_i, y_i) represent coordinates of the center point of the o_i.

Finally, the combination of the mathematical modeling of observations (A and B) for each detected object in the input image represents the priority factor's score of that object as shown in Figure 3.5. Such a combination is donated by Equation 3.7. Figure 3.6 illustrates the embedding of pf in the feature set of the object o_i.

$$pf(o_i) = \frac{d(o_i)}{r(o_i)} \qquad (3.7)$$

	objects area	mean	ratio	image center	objects center	distance	pf
0	21827.126933	68141.737857	0.320319	(384.0, 288.0)	(576.9182373046875, 122.79302558898925)	253.989745	792.926283
1	57386.692303	68141.737857	0.842167	(384.0, 288.0)	(221.89080047607422, 381.1133674621582)	186.947832	221.984394
2	125211.394334	68141.737857	1.837514	(384.0, 288.0)	(347.11689147949215, 277.51868591308596)	38.343469	20.867036

Figure 3.5:The results of the mathematical modeling of observations (A and B) with computing the pf feature for each detected object.

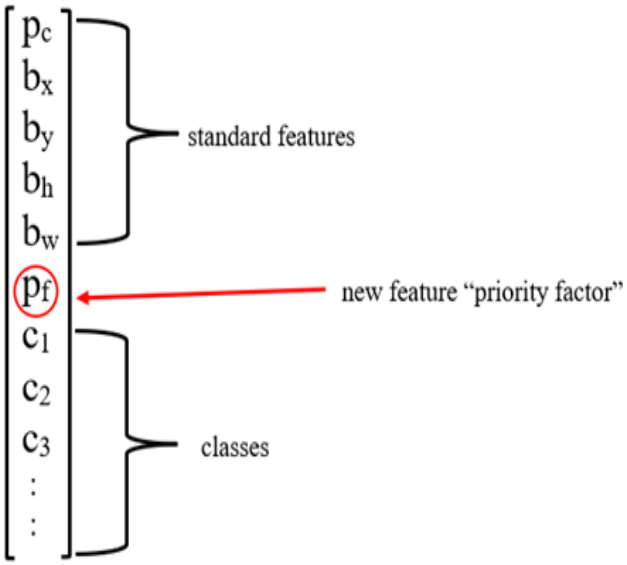


Figure 3.6: Priority Factor Feature Embedding with Standard Features of YOLOv7

After computing pf for every o_i in O, objects are sorted in ascending order according to pf using heap sorting algorithm as depicted in Figure 3.7. The objects feature array undergoes a transformation into a 1D array of specified length,

subsequently being padding with zero values spanning a length of 1280 units. This adaptation is performed to align with the output configuration of the EfficientNetV2 model, yielding a resultant 1D array measuring 1×1280 in length.

	b_x	b_y	b_w	b_h	c	p_c	pf
[	127.2	135.18	567.04	419.86	1	0.96248	20.867]
[	131.99	221.52	311.79	540.7	16	0.95941	221.98]
[	463.37	74.737	690.47	170.85	7	0.8693	792.93]

Figure 3.7: Objects Sorting Ascending According to pf

d) Features Concatenation

Upon the availability of all features (i.e. Deep visual features, Object features with *pf* feature), they become ready for concatenation to form a matrix (𝕍) of dimension 145x1280 that contains essential knowledge of various features for the input image as depicted in Figure 3.8.

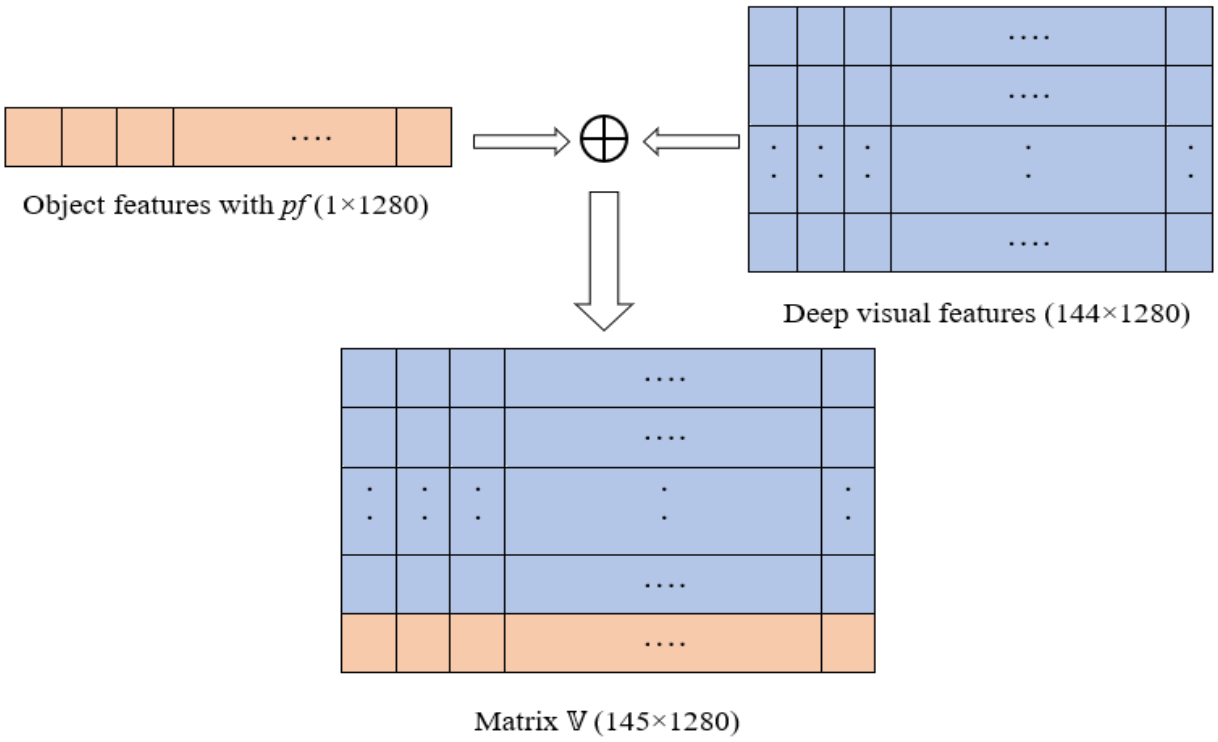


Figure 3.8: Features Concatenation Workflow

3.2.4 Features Decoding Phase

The features decoding phase of the architecture is a deep neural network that comprises three blocks with embedded layers, namely: Reduction Block, Attention Block (semantic stage) and Language Block, as illustrated in Figure 3.9

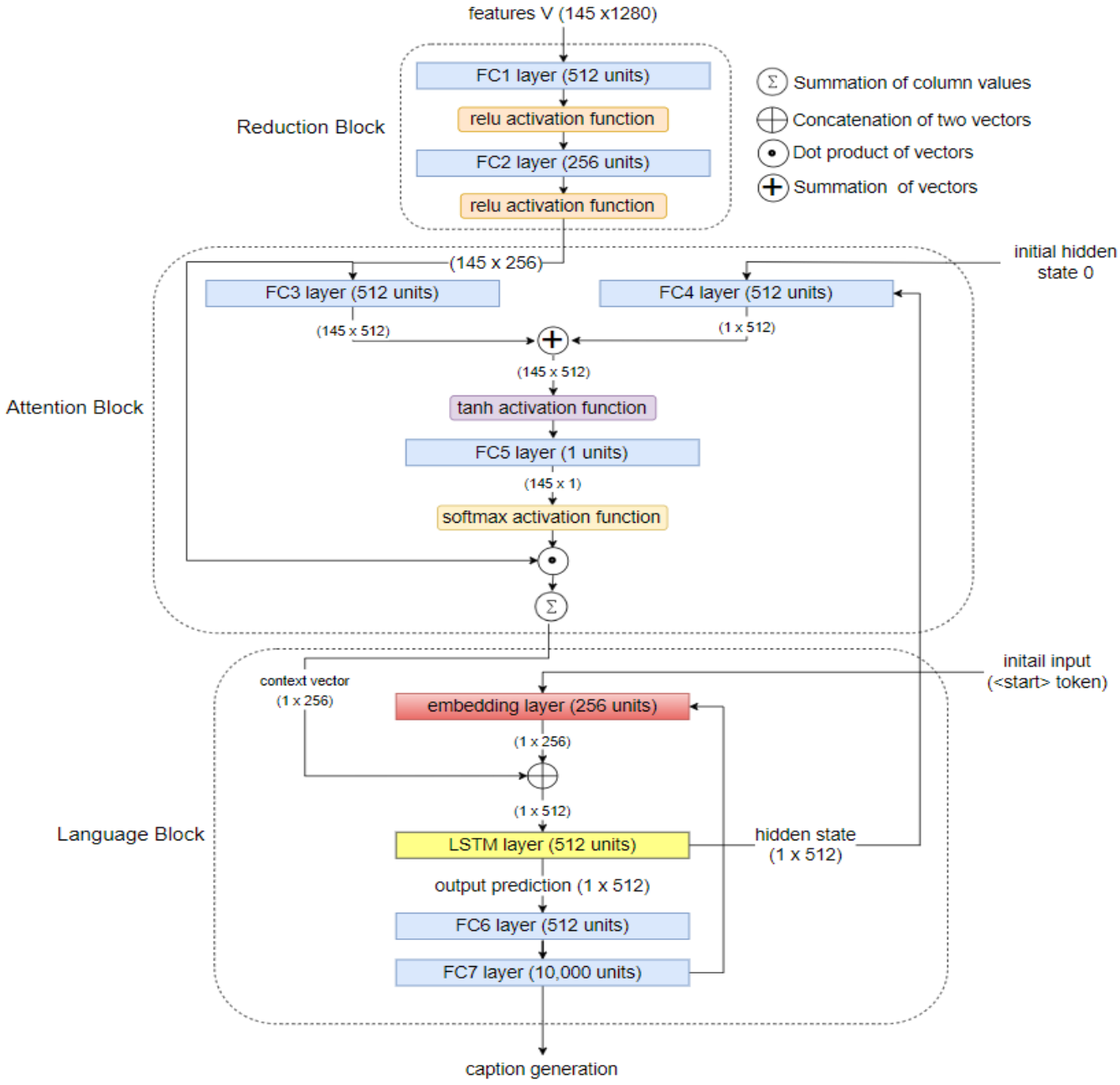


Figure 3.9: The Architecture of Features Decoding Phase

a) Reduction Block

Due to concatenation, $\mathbb{V}$ can readily become highly dimensional. Thus, it becomes difficult to understand by the language generator down the stream. Accordingly, the proposed system utilizes a two-FC layer traditional neural network for dimensionality reduction. The first layer with a length of 512, and the second with a length of 256 which represents the output layer. Unlike previous works, this system concatenates both sets of features before the reduction of $\mathbb{V}$. This network results in an array with dimensions of 145 × 256.

Algorithm 3. 2 describes the details of the reduction block. The algorithm takes the image features resulting from the concatenation step (i.e., $\mathbb{V}$) as input. A feature $\overline{v_j} \in \mathbb{V}$ is a vector of dimensions (145×1280). In step (1) the algorithm process each $\overline{v_j}$ in order to unify its cardinality with size 512 using $FC_1()$ layer and the result is a 2D array S with size 145×512 where each $\overline{s_i}$ is a transformed vector within S feature space. After that each s_{ij} is input to $relu$ () activation function to normalize the values. In step (3) the algorithm uses a $FC_2()$ layer to process each $\overline{s_i}$ in order to unify its cardinality with size 256. The result is R with size (145×256). Finally, applying $relu$ () activation function for each r_{ij} , $\forall\, r_{ij} \in R$.

Algorithm 3. 2: Reduction Block

---

input*:*

$\mathbb{V}$: $image\ features$, $\forall\, \overline{v_j} \in \mathbb{V}$, $0 < j \leq |\mathbb{V}|, \left|\overline{v_j}\right| = 1280$

output:

R: features dimensions reduction to 145×256

1) reduction of the features with size (145 x 512)

$\overline{s_i} \longleftarrow FC_1\left(\overline{v_j}\right)$, $\forall\, \overline{s_i} \in S$

2) use relu function to normalize the values

$s_{ij} \longleftarrow relu\left(s_{ij}\right)$

3) reduction of the features with size (145 x 256)

$\overline{r_i} \longleftarrow FC_2(\overline{s_i}), \forall \overline{r_i} \in R$

4) use relu function to normalize the values

$r_{ij} \longleftarrow relu(r_{ij})$

---

b) Attention Block (semantic stage)

The proposed system regards attention as a means to select the features that are more relevant according to semantic information learned by Gradient Descent during training. It utilizes the global attention introduced by Bahdanau to signify the prominence of object features. As such, this can help readily collect class and position information to grasp context (semantic) regarding the image rather than merely deep visual features. This architectural design allows the decoder component to dynamically utilize the most relevant sections of the input sequence, and this can be achieved by combining all the encoded input vectors into a weighted combination, where the vectors deemed most relevant are assigned higher weights. Therefore, assures that the decoder focuses on the most important information during the generation of output sequences.

Figure 3.10 illustrates the visual representation of attention results. Although this result relies on gradients, as humans, we do not directly perceive these differentials. Instead, they are analyzed and understood by our system managing the training process. The system adjusts the parameters based on these gradients without us having a clear awareness of the precise changes being made.

Algorithm 3. 3 outlines the complicated structure of the attention network architecture. The input of this algorithm is as follows:

- The image features $\boldsymbol{R}$ is extracted by the reduction block where $\overline{\boldsymbol{r_j}} \in \boldsymbol{R}$, $\overline{\boldsymbol{r_j}}$ is vector represents a feature within the set. Empirically, $\boldsymbol{R}$ is a 2D array of probabilistic distributions ($\mathbf{145 \times 256}$).
- The second input is a vector $\overline{\boldsymbol{h}}$ obtained from the language block, $\overline{\boldsymbol{h}}$ is a 1D array ($\mathbf{1 \times 512}$).

In step (1) the algorithm process each $\overline{\boldsymbol{r_j}}$ in order to unify its cardinality with $\overline{\boldsymbol{h}}$ using the FC layer $\boldsymbol{FC_1}()$ and the result is a 2D array $\boldsymbol{W1}$ with size $\mathbf{145 \times 512}$. After that it pass $\overline{\boldsymbol{h}}$ vector to $\boldsymbol{FC_2}()$ and the result is $\overline{\boldsymbol{w2}}$ 1D array of probabilistic distributions ($\mathbf{1 \times 512}$). Next step we perform vector addition $(\overline{\boldsymbol{w2}} + \overline{\boldsymbol{w1_i}})$ where $\overline{\boldsymbol{w1_i}}$ is a transformed vector within $\boldsymbol{W1}$ feature space. The result is further processed by applying $\boldsymbol{tanh}()$ activation function to normalize the values in the range (-1,1). The motivation behind such normalization is identifying the importance of a $\overline{\boldsymbol{w1_i}}$ as part of activation maximization undertaking. $\boldsymbol{T}$ undergoes a transformation to reduce its dimensions to (145×1) through the use of the function $\boldsymbol{FC_3}()$. In step (5) each $\boldsymbol{a_i}$ is normalized using softmax function $\boldsymbol{\sigma}()$ to become a suitable probability distribution over the predicted output classes. The results are positive scores that represent attention weights which can be stored back in their locations in $\boldsymbol{\acute{A}}$. After that the algorithm computes $(\boldsymbol{\acute{A}} \cdot \boldsymbol{R})$ The reason behind such computation is to determine the relevant features of the corresponding word. Finally, we summate each $\boldsymbol{w_j}$ in $\boldsymbol{W}$ to produce the context vector $\overline{\boldsymbol{c}}$ of size $\mathbf{1 \times 256}$.

Algorithm 3. 3: Attention Block (semantic stage)

input*:*

$R: image\ features, \forall\ \overline{r_j} \in R\ , 0 < j\ \leq |R|, |\overline{r_j}| = 256$
$\overline{h}: Hidden\ State\ Vector, \forall\ h_i \in \overline{h}, 0 < i \leq 512$

output:

$\overline{\boldsymbol{c}}$ *: context vector*

1) extending features dimensions from (145 x 256) to (145 x 512)

$\overline{w1_i} \leftarrow FC_1(\overline{r_j}), \forall \overline{w1_i} \in W1$

2) processing $\overline{h}$ vector using $FC_2()$ layer

$\overline{w2} \leftarrow FC_2(\overline{h})$

3) vector addition ($\overline{w2} + \overline{w1_i}$) The result is further processed by applying tanh function

$\overline{t_i} \leftarrow tanh(\overline{w2} + \overline{w1_i}), \forall \overline{t_i} \in T$

4) T is transformed to flatten using $FC_3()$

$a_i \leftarrow FC_3(\overline{t_i}), \forall a_i \in A$

5) each a_i is normalized using softmax function

$\acute{a}_\iota \leftarrow \sigma(a_i) \ , \forall \acute{a}_\iota \in \acute{A}$

6) compute the dot product of vectors $(\acute{A} \cdot R)$

$W \leftarrow \acute{A} \cdot R$

7) summation each $\boldsymbol{w_j}$ in $\boldsymbol{W}$ to produce the context vector $\overline{\boldsymbol{c}}$

$\boldsymbol{c_j} \leftarrow \sum_{i=1}^{n} \boldsymbol{w_{ij}} \ , \forall \boldsymbol{c_j} \in \overline{\boldsymbol{c}}$

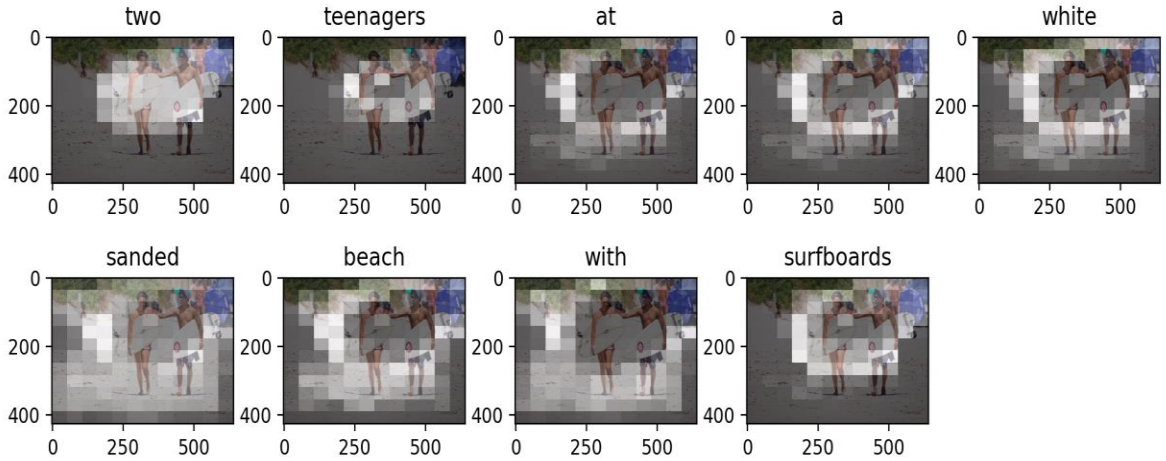


Figure 3.10: Visual Representation of Attention Results.

c) Language Block

The language block consists of four layers: Embedding Layer, LSTM Layer, Computation Layer, and Output Layer. The first layer takes a sequence of words from the input caption, and returns one dimension vector of 256 length. The LSTM layer is responsible for generating two outputs: hidden state and prediction vectors. We rely on LSTM due to its beneficial performance with long dependencies. It can remember information for long periods and provides wide range of parameters, such as learning rates and biases, thus eliminating the need for finetuning. LSTM produces a caption by generating one word relative to the context vector in the previous hidden state as well as the previously generated words. The remaining layers are FC layers where the computation layer is of size 512, and the second is of length equals the vocabulary dimensions.

Algorithm 3. 4 depicts the details of language block workflow. The algorithm takes two inputs:

- The word sequence w . Empirically, w is the key of the word in the dictionary.
- $\overline{c}$ is 1D vector (1×256) which represents the context vector obtained from the attention block.

In the step (1) the algorithm passes w to the embedding layer $EMBED()$ to transform it into a vector $\overline{w}$ of 256 length. After that concatenates $(\overline{c} \oplus \overline{w})$ to prepare LSTM input which in turn returns two vectors $\overline{h}$, $\overline{p}$ of size 512. Next step the algorithm processes $\overline{p}$ using $FC_1()$ layer for computation purposes. The result is 1D vector $\overline{\acute{p}}$ with size (1×512). Finally, $\overline{\acute{p}}$ passes to $FC_2()$ layer to extend its dimensionality with the vocabulary size. The output is $\overline{\acute{P}}$ 1D vector with size (1×10000).

Algorithm 3. 4: Language Block

input*:*

w : $word\ sequnce$
$\overline{c}$*: context vector*
output*:*

$\overline{\acute{P}}$*: 1D prediction vector*
$\overline{h}$*: hidden state vector*

1) w is transformed into vector using embedding layer

$\overline{w} \leftarrow EMBED(w)$

2) preparing LSTM input by concatenating ($\overline{c} \oplus \overline{w}$)

$\overline{h}, \overline{p} \leftarrow LSTM(\overline{c} \oplus \overline{w})$

3) $\overline{p}$ undergoes a processing using $FC_1()$ layer

$\overline{\acute{p}} \leftarrow FC_1(\overline{p})$

4) $\overline{\acute{p}}$ undergoes a processing using $FC_2()$ layer

$\overline{\acute{P}} \leftarrow FC_2(\overline{\acute{p}})$

3.3 Training of The Proposed System

Before the training phase, the dataset undergoes a preparatory stage. The raw dataset consists of two files: the first file includes images, while the second file is a JSON file containing captions for all images in the first file. The features corresponding to each image in the first file are extracted and archived within a file with the "npy" extension using the features extraction phase, but before this phase, all the images and captions undergo the preprocessing phase. The final result is the generation of database *D* which comprises a set of features and a set of captions as depicted in Figure 3.11.

Algorithm 3. 5 depicts the main training workflow of the proposed system. The input of this algorithm is as follows:

- A set of image features denoted by V, each element in V is a 2D array (145 × 1280) which represents the features of a single image in the training dataset.
- A set of textual phrases donated by S. Each phrase represents the description of various objects appearing in an image, and assigned by an expert.
- $\overline{h}$ is 1D vector (1×512) which represents the hidden state, and is initialized with 0.
- The desired number of training epochs.

The processing workflow for any given training epoch is as follows:

Step-2: For each image in V, i.e. V_i

Step-3: The image's features set V_i must be sent to the routine $Reduction\ Block$ in order to reduce its dimensionality from 145 X 1280 to 145 X 256.

Step-4: The reduced feature set, donated by r_i, is passed together with the hidden state vector ($\overline{h}$) to the routine $Attention\ Block$ for learning the relevance between r_i features and their corresponding caption words as discussed in Step-8. The result is an initial context vector $\overline{c_i}$

Step-5: The loss function in initialized to zero.

Step-6: In this step, the algorithm loops into each word s_{ij} in the caption $s_i, s_i \in S$ which corresponds to the image V_i in the dataset V.

Step-7: Each s_{ij} is passed with $\overline{c_i}$ to the $Language\ Block$ routine. The latter returns two results:

- The prediction vector p_{ij} with length that equals the vocabulary size (in the context of this thesis the size is 1×10000).
- The updated hidden state $\overline{h}$ which reflects the learning of relevance.

Step-8: In this step, we call the $Attention\ Block$ routine again with two parameters: the original image features set, and the updated $\overline{h}$ in Step-7. The result is an updated context vector that will be used for prediction calculation for the word s_{ij+1}.

Step-9: The algorithm uses the function $LOSS$ to compute the loss between the predication of the word s_{ij} and the next word s_{ij+1} in the caption set. The loss values are accumulated in the variable l_i. Steps (7,8, and 9) continues looping for all the images and their corresponding captions.

Step-11: In this step, the average loss is computed for the feature set V_i and its corresponding caption s_i. The result is stored in av_i

Step-13: Finally, av represents the average loss of the entire training epoch n, and is computed by divining the sum of loss averages for all images by the size of the dataset.

Algorithm 3. 5: Training Iteration

---

input*:*

a set of image features V, a set of captions S, $\overline{h}$ is an initial hidden state vector, and epoch initialization N=25

output*:*

av: represents the average loss of the entire training epoch n

1) **FOR** $epoch = 1\ to\ N$ **DO**

2) **FOR** i =1 to |V| **DO**

3) $r_i \leftarrow Reduction\ Block\ (V_i)$

4) $\overline{c_i} \leftarrow Attention\ Block\ \left(r_i, \overline{h}\right)$

5) $l_i \leftarrow 0$

6) **FOR** j = 1 to $|s_i|$ **DO**

7) $(\overline{p_{ij}}, \overline{h}) \leftarrow Language\ Block\ (\overline{c_i}, s_{ij})$

8) $\overline{c_i} \leftarrow Attention\ Block\ \left(r_i, \overline{h}\right)$

9) $l_i \leftarrow l_i + LOSS(\overline{p_{ij}}, s_{ij+1})$

10) **END**

11) $av_i \leftarrow \frac{l_i}{|s_i|}$

12) **END**

13) $av \leftarrow \frac{\sum_{i=1}^{|V|} av_i}{|V|}$

14) **END**

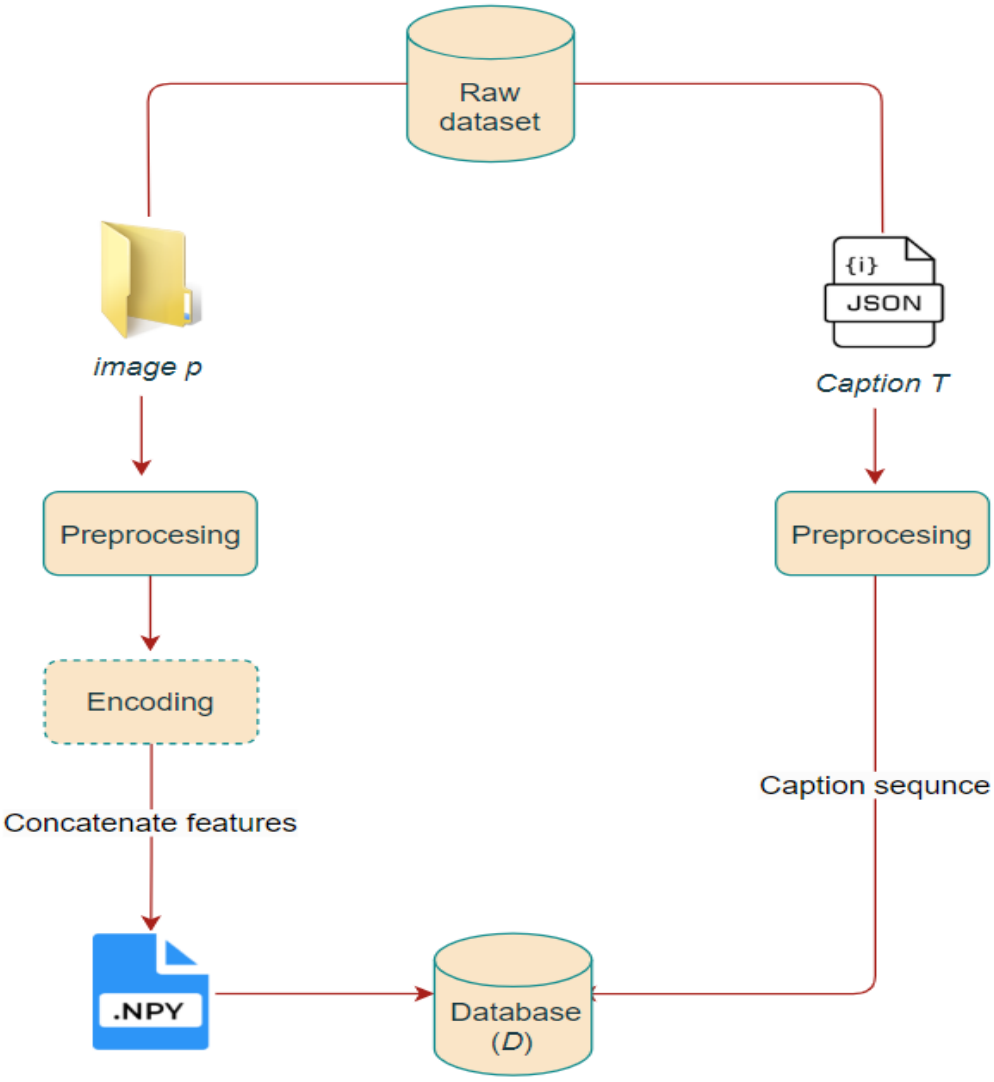


Figure 3.11: Preparing Dataset for Training

Chapter Four
Experimental Results and Discussion

Chapter Four: Experimental Results and Discussion

4.1 Overview

The proposed system explained in Chapter Three is applied to achieve the thesis aims explained in Chapter One. The findings of all phases are ordered according to their appearance in Chapter Three. However, this Chapter begins with the software and hardware configurations along with explaining the dataset for implementing the proposed system.

4.2 Software and Hardware Configurations

Implementing image processing systems using machine learning with a deep learning approach requires high computer resources to be applied flexibly, especially with huge datasets. Therefore, the proposed system was implemented using two main sources:

A. Local Software and Hardware

- Central Processing Unit (CPU): Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz.
- RAM: 16 GB.
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX 1060 6GB.
- Hard Disk: 256 GB SSD.
- Operating system: Windows10, 64 bit.
- Programing language: Python 3.9 Anaconda with PyCharm 2022 IDE.

B. Cloud GPU

- Google Colab Pro subscription for 1 month.
- Google Colab Pro+ subscription for 3 months.

4.3 Implementation Details

In this thesis, the loss function used during training is sparse categorical cross entropy. Adam optimizer was used as the optimizer. Therefore, we adopt the following configurations:

- Learning rate = 0.001
- Epoch = 25
- Batch size = 32, and early stop if the metric scores discontinue improving.

4.4 Results of Data Preprocessing

The outcomes of the preprocessing phase involving both images and captions are displayed after conducting the following: image resizing, adding unique tokens for each caption, tokenization of captions, dictionary construction, transformation of each caption into a vector, and finally padding.

a) Images Resizing

It is essential to resize the input images to 384 × 384 in order to be compatible with the architectural requirements of the EfficientNetV2 CNN model due to the dataset's variable image dimensions make this modification necessary (See Figure 4.1).

Figure 4.1: Resize Sample of the Dataset Images

b) Adding Unique Tokens

For each caption in the dataset, unique tokens should be added which are <start> and <end> where <start> is added at the beginning of the caption and <end> is added at the caption end as shown in Table 4.1.

Table 4.1: Sample of the dataset captions before and after adding unique tokens

Before Adding Unique Tokens	"Three large dollies with scattered luggage place on top " "A person walking with an umbrella against the wind" "A dog is laying down and getting his teeth brushed" "A teddy bear is sitting on a chair"
After Adding Unique Tokens	"<start> Three large dollies with scattered luggage place on top <end> " "<start> A person walking with an umbrella against the wind <end>" "<start> A dog is laying down and getting his teeth brushed <end>" "<start> A teddy bear is sitting on a chair <end>"

c) Tokenization of Caption

All captions in the dataset are converted into separated words based on punctuations, special characters, and white spaces by tokenizing the textual content (See Table 4.2)

Table 4.2: The captions sample before and after the tokenization process

Before Tokenization	"<start> Three large dollies with scattered luggage place on top <end> " "<start> A person walking with an umbrella against the wind <end>" "<start> A dog is laying down and getting his teeth brushed <end>" "<start> A teddy bear is sitting on a chair <end>"
After Tokenization	[' <start>',' Three',' large', 'dollies', 'with', 'scattered', 'luggage', 'place',' on', 'top','<end>'] ['<start>', 'A', 'person', 'walking', 'with', 'an', 'umbrella', 'against', 'the', 'wind','<end>'] ['<start>', 'A', 'dog', 'is', 'laying', 'down', 'and', 'getting', 'his', 'teeth', 'brushed','<end>'] ['<start>', 'A', 'teddy', 'bear', 'is', 'sitting', 'on', 'a', 'chair','<end>']

d) Dictionary Construction

The resulting tokens after the tokenization process are then counted and sorted based on their frequencies. Only the tokens within top 10,000 ranks are chosen as our vocabulary (i.e. dictionary) to avoid overfitting as depicted in Table 4.3.

Table 4.3: The captions sample before and after the dictionary construction

Before Dictionary Construction	[’ <start>’,’ Three’,’ large’, ’dollies’, ’with’, ’scattered’, ’luggage’, ’place’,’ on’, ’top’,’<end>’] [’<start>’, ’A’, ’person’, ’walking’, ’with’, ’an’, ’umbrella’, ’against’, ’the’, ’wind’,’<end>’] [’<start>’, ’A’, ’dog’, ’is’, ’laying’, ’down’, ’and’, ’getting’, ’his’, ’teeth’, ’brushed’,’<end>’] [’<start>’, ’A’, ’teddy’, ’bear’, ’is’, ’sitting’, ’on’, ’a’, ’chair’,’<end>’]
After Dictionary Construction	{1: '<start>', 2: '<end>', 3: 'a', 4: 'with', 5: 'on', 6: 'is', 7: 'three', 8: 'large', 9: 'dollies', 10: 'scattered', 11: 'luggage', 12: 'place', 13: 'top', 14: 'person', 15: 'walking', 16: 'an', 17: 'umbrella', 18: 'against', 19: 'the', 20: 'wind', 21: 'dog', 22: 'laying', 23: 'down', 24: 'and', 25: 'getting', 26: 'his', 27: 'teeth', 28: 'brushed', 29: 'teddy', 30: 'bear', 31: 'sitting', 32: 'chair'}

e) Caption Transformation to Vector

After dictionary construction, each caption after the tokenization process is converted into a vector by comparing each token with the dictionary. Table 4.4 shows a sample of the captions before and after transforming them into vectors.

Table 4.4: Caption Transformation to Vector

Before	[’ <start>’,’ Three’,’ large’, ’dollies’, ’with’, ’scattered’, ’luggage’, ’place’,’ on’, ’top’,’<end>’] [’<start>’, ’A’, ’person’, ’walking’, ’with’, ’an’, ’umbrella’, ’against’, ’the’, ’wind’,’<end>’] [’<start>’, ’A’, ’dog’, ’is’, ’laying’, ’down’, ’and’, ’getting’, ’his’, ’teeth’, ’brushed’,’<end>’] [’<start>’, ’A’, ’teddy’, ’bear’, ’is’, ’sitting’, ’on’, ’a’, ’chair’,’<end>’]

After	[[1, 7, 8, 9, 4, 10, 11, 12, 5, 13, 2], [1, 3, 14, 15, 4, 16, 17, 18, 19, 20, 2], [1, 3, 21, 6, 22, 23, 24, 25, 26, 27, 28, 2], [1, 3, 29, 30, 6, 31, 5, 3, 32, 2]]

f) Padding

This means captions need to be padded with zeros to make them uniform in length since the length of features vary. Table 4.5 depicts a sample of the captions vectors before and after padding.

Table 4.5 The use of padding

Before	[[1, 7, 8, 9, 4, 10, 11, 12, 5, 13, 2], [1, 3, 14, 15, 4, 16, 17, 18, 19, 20, 2], [1, 3, 21, 6, 22, 23, 24, 25, 26, 27, 28, 2], [1, 3, 29, 30, 6, 31, 5, 3, 32, 2]]
After	[[1, 7, 8, 9, 4, 10, 11, 12, 5, 13, 2, 0], [1, 3, 14, 15, 4, 16, 17, 18, 19, 20, 2, 0], [1, 3, 21, 6, 22, 23, 24, 25, 26, 27, 28, 2], [1, 3, 29, 30, 6, 31, 5, 3, 32, 2, 0, 0]]

4.5 Experiments

In this thesis, four experiments were conducted on the proposed system, especially the features extraction phase (Encoding):

- Experiment 1 compared the proposed system before and after adding object detection.
- Experiment 2 compared the proposed system before and after adding a *pf* feature to features of object detection.

- In experiment 3, the proposed system as whole (with object detection & *pf*) was compared with the reduction process before and after the concatenation step.
- In experiment 4 our proposed system as whole is compared with state of the art approaches in terms of Precision(P) and Recall(R) of the reported caption.

4.5.1 Results of the First Experiment

Table 4.6 shows the quantitative comparison of experiment 1. The inclusion of object detection results in a significant improvement in assessment scores, especially in CIDEr and SPICE scores, which increased from 83.1-89.1 and 14.2-15, respectively. This increase is considered good compared to the results of the Al-Malla [20], indicating a high level of concurrence with human judgment. Additionally, this improvement underscores the efficacy of incorporating object features to enhance the overall quality and relevance of generated descriptions, thereby bridging the gap between machine generated outputs and human perception and comprehension. Figure 4.2 depicts a comparison of the results of Experiment 1 before and after adding object detection.

Table 4.6: Results of experiment 1. Bold indicates the highest score, and OD denotes Object Detection.

Proposed system	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr	ROUGH-L	SPICE
Without OD	72.5	57.2	44.9	32.3	25.4	83.1	50.1	14.2
With OD	**75**	**59.5**	**46.9**	**33.8**	**26.4**	**89.1**	**51.6**	**15**

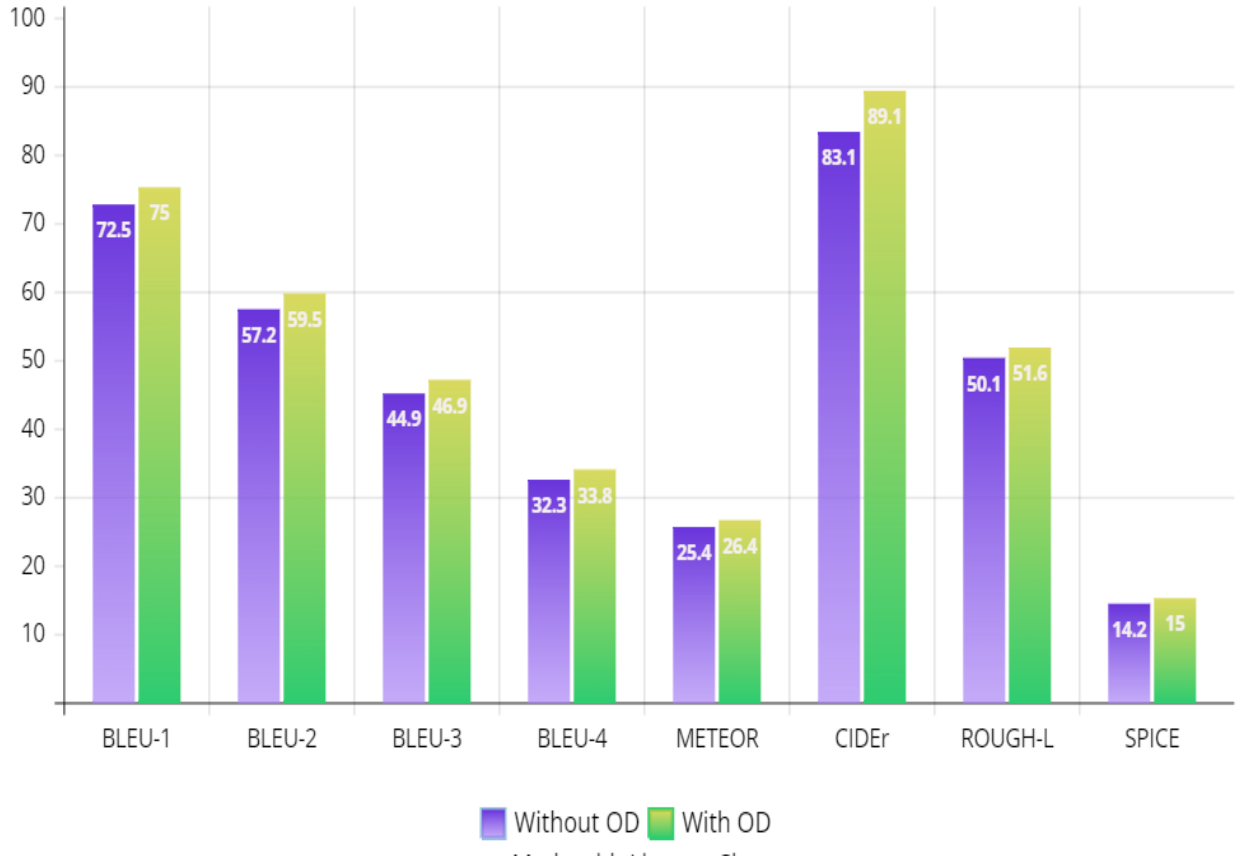


Figure 4.2: Comparison of the results of experiment 1. OD denotes Object Detection.

Figure 4.3 depicts the qualitative results of experiment 1. It is obvious that the proposed system with object detection features generates captions with higher quality compared to those produced by this same system but without using these features. In Figure 4.3(a), our system without object detection features can identify a "two sheep ". However, by adding object detection, the system identifies "group of sheep".

In Figure 4.3(b), the proposed system without object detection features is unable to generate accurate captions compared to the improved caption generation after adding these features. As shown in Figure 4.3(c), the proposed system generates the caption “a group of young people sitting around a table.”, and the object detection variation gives the caption “a group of people that are sitting at a birthday party.”. In Figure 4.3(d), our system without adding the object detection model mixes

animals and people. In Figure 4.3(e), the proposed system identifies the "sink" of a bathroom after adding an object detection model.

Figure 4.3: Examples of captions generated by experiment 1. OD denotes Object Detection

In Figure 4.3(f), the proposed system is capable of describing a group of motorcycles rather than one motorcycle after adding object detection. In Figure 4.3(g), without object detection, the system was unable to identify the 'tree' and

'rocks' behind the giraffes. Finally, in Figure 4.3(h), we are able to identify the "glass vases" holding flowers.

4.5.2 Results of the Second Experiment

Table 4.7 shows the results of second experiment based on a quantitative comparison. We observe that our proposed scheme (i.e. *pf*) when it was combined with object features contributes in increasing BLEU metrics, while other metric values remain unaffected. In contrast to the results of Al-Malla [20], our scheme does not result in a decrease in the METEOR score. Figure 4.4 depicts the minor increase in BLEU metrics of the results of Experiment 2 before and after adding *pf*.

Table 4.7: Results of experiment 2. Bold indicates the highest score

Proposed system	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr	ROUGH-L	SPICE
With OD	75	59.5	46.9	33.8	26.4	89.1	51.6	15
With OD & *pf*	**76**	**60**	**47.1**	**34**	26.4	89.1	51.6	15

Experiment 2 is qualitatively evaluated, and the results are depicted in Figure 4.5. For example, in Figure 4.5(a), the proposed system generates the caption “two sheep grazing in field”, and the object features variation gives the caption “group of sheep together in a flooring.”, whereas the object features with the added *pf* feature, the system generates the caption “a very herd of sheep with different features of sheep”. From semantic point of view, it can be readily observed that the captions generated with the *pf* approach have better semantics with respect to the objects appearing in the input image, compared to the captions generated by other approaches. For example, the object features approach (without *pf*) can identify a "group of sheep". However, when adding *pf*, it identifies "sheep features".

Similarly, in Figure 4.5(b), the proposed system identifies "glasses" in the image along with “sitting woman” due to the effect of *pf* addition. In Figure 4.5(c), the object detection approach (without *pf*) was unable to generate the "dessert" eaten by individuals at the table compared to the improved caption generation after adding *pf*. In Figure 4.5(d), our system with the object detection model (without *pf*) mixes animals and people, whereas with adding *pf* the system can identify the 'cows'.

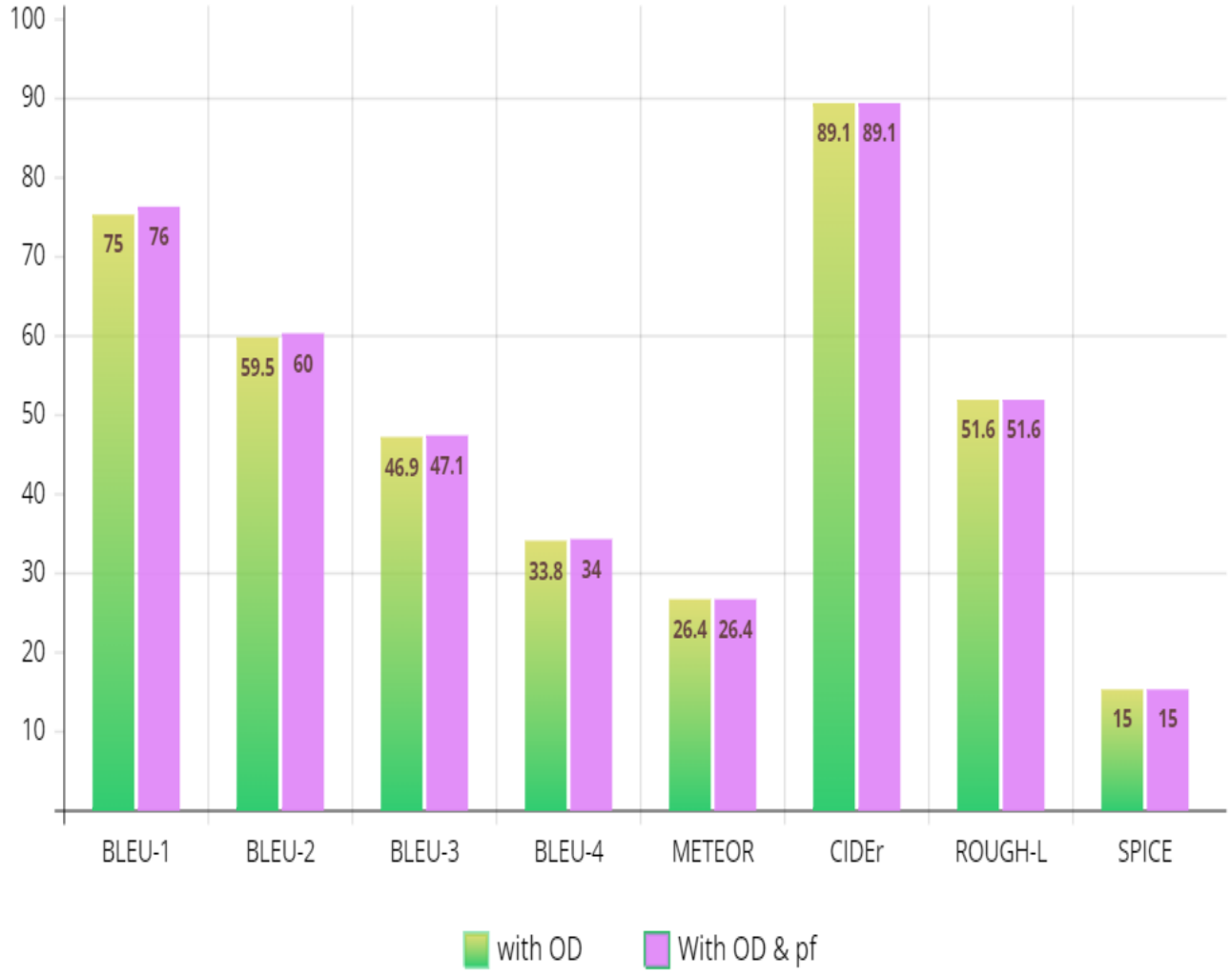


Figure 4.4: Comparison of the results of experiment 2

Figure 4.5: Examples of captions generated by experiment 2

4.5.3 Results of the Third Experiment

Table 4.8 demonstrates the results of experiment 3. A slight improvement is observed in the evaluation scores, particularly in the BLEU 1, BLEU 2, and BLEU 3 metrics, where a 0.63 % increase is evident when conducting the reduction process (i.e. Reduction Block) of $\mathbb{V}$ features after the concatenation step. This experiment

illustrates the effectiveness of this method in improving the accuracy, especially the BLUE metric.

Table 4.8: Results of experiment 3. Bold indicates the highest score

Proposed system	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr	ROUGH-L	SPICE
With (OD & *pf* and reduction before concatenation)	76	60	47.1	34	26.4	89.1	51.6	15
With (OD & *pf* and reduction after concatenation)	**76.2**	**60.1**	**47.2**	34	26.4	89.1	51.6	15

4.5.4 Results of the Fourth Experiment

Table 4.9 presents the results of the fourth experiment. These results show that the proposed system outperforms the selected counterparts in terms of *BLEU-1*, *BLEU-2*, *BLEU-3*, and *SPICE* metrics. Although previous experiments prove that *SPICE* score is hard-to-improve measure, our method achieves 2nd of *BLEU-4* with only a 1.2 difference from the best score. Similarly, w.r.t *METEOR*, we achieve second rank whereas Iwamura et al. [19] achieved first rank, w.r.t *CIDEr* and *ROUGH-L* we achieve third place whereas they achieved first rank. Finally, we achieve best scoring w.r.t the *BLEU-1*, *BLEU-2*, *BLEU-3*, and *SPICE* metrics.

Table 4.9: Results of experiment 4. Bold indicates the highest score

State of the art approaches	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr	ROUGH-L	SPICE
Yang et al. [16]	70.4	53.1	39.2	29	23.8	85	52.1	-
Yin and Ordonez [17]	-	-	-	25.3	23.8	92.2	50.7	-
Iwamura et al. [19]	75.9	59.9	46	**35.2**	**26.7**	**109.9**	**55.8**	-
Al-Malla et al. [20]	49.2	29.6	17.4	10.1	16.3	39	35.8	10.8
Ours	**76.2**	**60.1**	**47.2**	34	26.4	89.1	51.6	**15**

Figure 4.6 displays a comparison between state of the art approaches and our system. We see a clear increase in the results on evaluation metrics, especially in BLEU-1, BLEU-2, BLEU-3, and SPICE, where we achieved scores of 76.2, 60.1, 47.2, and 15, respectively. The acceptable ranges for BLEU, METEOR, CIDEr, ROUGH-L, and SPICE are 60-80, 20-30, 50-100, 40-60, and 10-20, respectively. Accordingly, the results of our system are good relative to these ranges and lead to improving the performance of captioning on images.

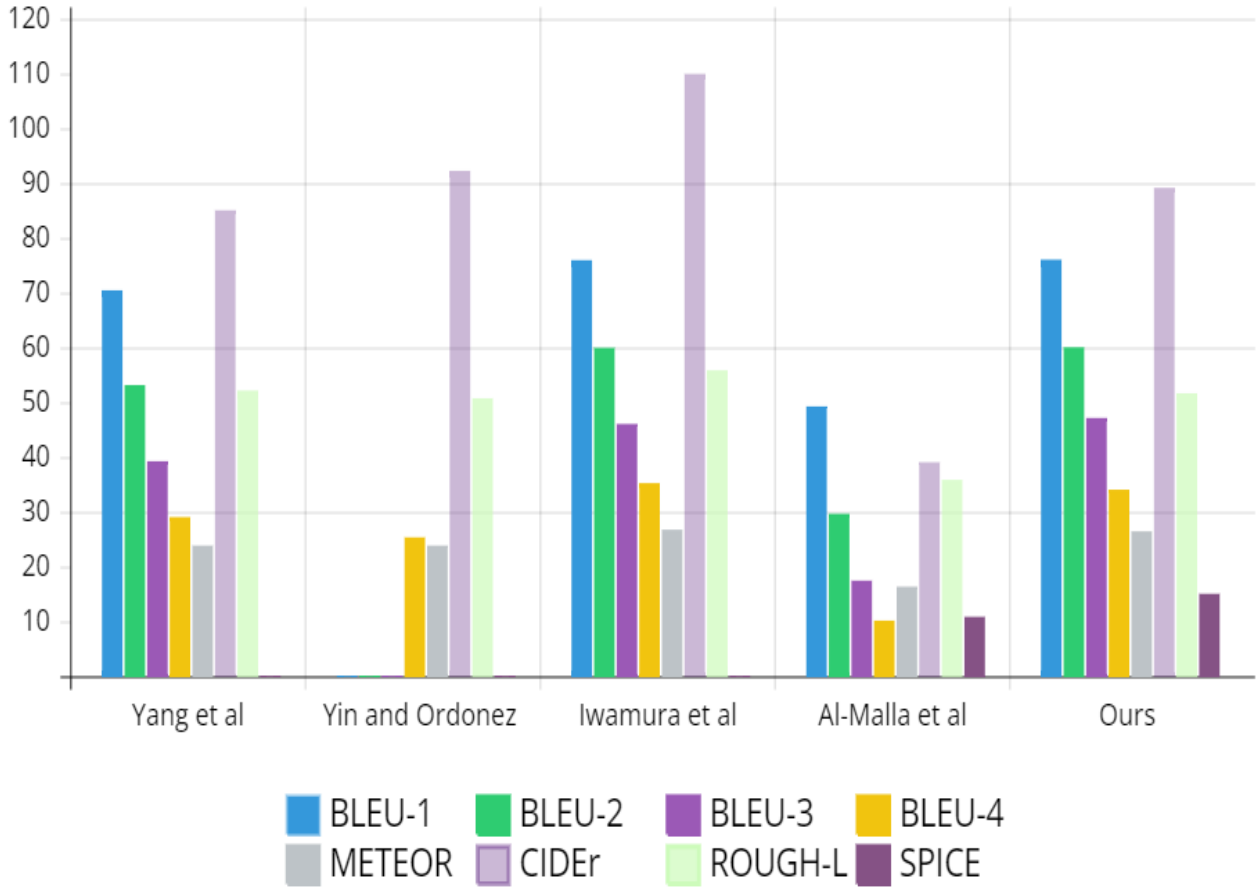


Figure 4.6: Comparison of the experiment 4 results between our system and state of the art approaches

Chapter Five
Conclusions and Future Works

Chapter Five: Conclusions and Future Work

5.1 Conclusions

Several conclusions throughout the design and implementation of the proposed system are drawn.

- Utilizing a combination of diverse features, both generic (deep visual features) and specific (object detection features), has been demonstrated to enhance caption accuracy, as shown by the results of experiment 1.
- The attention mechanism takes an important role in the proposed system architecture, especially in the features decoding phase (Decoding). This mechanism directs the decoder to focus on specific features derived from diverse feature combinations. Empirically, the attention mechanism prioritizes object detection features known for their semantic richness in describing the input image. Additionally, it feeds the decoder with relevant features that bridge the gap between text and image features.
- A new feature called priority factor (pf) was introduced in addition to the features of the object detection. pf is a score derived from the features of an object in order to rank its salience with respect to other objects within that image's landscape. This schema shows the empirical effectiveness of the captioning task from semantic point of view as demonstrated in the analysis of experiment 2.
- Unlike previous works, this thesis shows that it is more effective to combine all raw features (Generic and Specific) in the first step of downstream processing workflow, and only then the feature reduction can take place. This

method demonstrates the observed improvement in the evaluation metrics especially the BLUE metric as shown in the results of experiment 3.

- The proposed system produced the best results for almost all considered metrics. Therefore, it demonstrably improved the image captioning performance compared with other state of the art approaches as demonstrated in the analysis of experiment 4.

5.2 Future Works

Several future directions could be highlighted based on the findings of this present thesis. A basic foundation can be provided upon which various image collections can be annotated with semantics supported captions drawn from (semi) structured knowledge sources (e.g. WordNet Ontology [40]). This can help facilitate various information retrievals tasks (e.g. semantic search, structural querying with machine readable metadata like RDF [41] or OWL [42]). Moreover, it is possible to conduct experiments on larger and more diverse datasets to validate generalization and scalability.

Regardless of such important outcomes, this thesis is not without limitations such as:

- Ensuring the generation of captions of excellent quality that align with human judgments.
- Dealing with potential biases that may occur when evaluating the system using datasets sourced from diverse domains, aiming to achieve unbiased and representative performance assessments.
- Training such a large-scale model requires high computation power which may not always be readily available.

References

[1] M. D. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, 2019.

[2] P. Anderson *et al.*, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6077–6086. doi: 10.1109/CVPR.2018.00636.

[3] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[4] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3128–3137. doi: 10.1109/CVPR.2015.7298932.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, pp. 3104–3112.

[6] A. Graves, "Long Short-Term Memory," in *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin Heidelberg, 2012, pp. 37–45. doi: 10.1007/978-3-642-24797-2_4.

[7] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[8] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *32nd International Conference on Machine Learning, ICML 2015*, in Proceedings of Machine Learning Research. 2015, pp. 2048–2057.

[9] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018.

[10] J. Johnson, A. Karpathy, and L. Fei-Fei, "DenseCap: Fully convolutional localization networks for dense captioning," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4565–4574. doi: 10.1109/CVPR.2016.494.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv Prepr. arXiv1804.02767*, 2018.

[12] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6517–

6525. doi: 10.1109/CVPR.2017.690.

[14] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars, “Guiding the long-short term memory model for image caption generation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2407–2415. doi: 10.1109/ICCV.2015.277.

[15] C. Wang, H. Yang, C. Bartz, and C. Meinel, “Image captioning with deep bidirectional LSTMs,” in *MM 2016 - Proceedings of the 2016 ACM Multimedia Conference*, 2016, pp. 988–997. doi: 10.1145/2964284.2964299.

[16] Z. Yang, Y.-J. Zhang, S. ur Rehman, and Y. Huang, “Image captioning with object detection and localization,” in *Image and Graphics: 9th International Conference, ICIG 2017*, 2017, pp. 109–118. doi: 10.1007/978-3-319-71589-6_10.

[17] X. Yin and V. Ordonez, “OBJ2TEXT: Generating visually descriptive language from object layouts,” in *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2017, pp. 177–187. doi: 10.18653/v1/d17-1017.

[18] V.-K. Vo-Ho, Q.-A. Luong, D.-T. Nguyen, M.-K. Tran, and M.-T. Tran, “A smart system for text-lifelog generation from wearable cameras in smart environment using concept-augmented image captioning with modified beam search strategy,” *Appl. Sci.*, vol. 9, no. 9, p. 1886, 2019.

[19] K. Iwamura, J. Y. L. Kasahara, A. Moro, A. Yamashita, and H. Asama, “Potential of Incorporating Motion Estimation for Image Captioning,” in *2021 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, Jan. 2021, pp. 23–28. doi: 10.1109/IEEECONF49454.2021.9382725.

[20] M. A. Al-Malla, A. Jafar, and N. Ghneim, “Image captioning model using attention and object features to mimic human image understanding,” *J. Big Data*, vol. 9, no. 1, p. 20, Dec. 2022, doi: 10.1186/s40537-022-00571-w.

[21] S. Liu, L. Bai, Y. Hu, and H. Wang, “Image captioning based on deep neural networks,” in *MATEC web of conferences*, EDP Sciences, 2018, p. 1052.

[22] B. Makav and V. Kılıç, “A new image captioning approach for visually impaired people,” in *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, IEEE, 2019, pp. 945–949.

[23] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: Lessons learned from the 2015 mscoco image captioning challenge,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 652–663, 2016.

[24] P. Bell, “Content analysis of visual images,” *SAGE Vis. methods Interpret. Classif.*, vol. 3, pp. 31–57, 2012.

[25] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[26] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, Ieee, 2005, pp. 886–893.

[27] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Gray scale and rotation invariant texture classification with local binary patterns," in *Computer Vision-ECCV 2000: 6th European Conference on Computer Vision Dublin, Ireland, June 26–July 1, 2000 Proceedings, Part I 6*, Springer, 2000, pp. 404–420.

[28] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[29] M. Stefanini, M. Cornia, L. Baraldi, S. Cascianelli, G. Fiameni, and R. Cucchiara, "From show to tell: A survey on deep learning-based image captioning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 539–559, 2022.

[30] S. Ghosh and S. Anupam, "CapText: Large Language Model-based Caption Generation From Image Context and Description," *arXiv Prepr. arXiv2306.00301*, 2023.

[31] D. E. Cahyani and I. Patasik, "Performance comparison of tf-idf and word2vec models for emotion text classification," *Bull. Electr. Eng. Informatics*, vol. 10, no. 5, pp. 2780–2788, 2021.

[32] E. M. Dharma, F. L. Gaol, H. Warnars, and B. Soewito, "The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification," *J Theor Appl Inf Technol*, vol. 100, no. 2, p. 31, 2022.

[33] A. Subakti, H. Murfi, and N. Hariadi, "The performance of BERT as data representation of text clustering," *J. big Data*, vol. 9, no. 1, pp. 1–21, 2022.

[34] V. Atliha and D. Šešok, "Text augmentation using BERT for image captioning," *Appl. Sci.*, vol. 10, no. 17, p. 5978, 2020.

[35] Y. Hu, H. Hua, Z. Yang, W. Shi, N. A. Smith, and J. Luo, "Promptcap: Prompt-guided image captioning for vqa with gpt-3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2963–2975.

[36] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, 2018.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[38] C. C. Aggarwal, "Neural networks and deep learning," *Springer*, vol. 10, no. 978, p. 3, 2018.

[39] R. W. Pettit, R. Fullem, C. Cheng, and C. I. Amos, "Artificial intelligence, machine learning, and deep learning for clinical outcome prediction," *Emerg. Top. life Sci.*, vol. 5, no. 6, pp. 729–745, 2021.

[40] J. Karhunen, T. Raiko, and K. Cho, "Unsupervised deep learning: A short review," *Adv. Indep. Compon. Anal. Learn. Mach.*, pp. 125–142, 2015.

[41] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC*

conference on computer and communications security, 2017, pp. 603–618.

[42] V. Avinash Sharma, "Understanding activation functions in neural networks," *Mach. Learn. Mastery. Available https//medium. com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0*, 2017.

[43] G. Lin and W. Shen, "Research on convolutional neural network based on improved Relu piecewise activation function," *Procedia Comput. Sci.*, vol. 131, pp. 977–984, 2018.

[44] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towar. Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[45] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Ann. Data Sci.*, pp. 1–26, 2020.

[46] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA, 2015.

[47] S. P. Siregar and A. Wanto, "Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (forecasting)," *IJISTECH (International J. Inf. Syst. Technol.*, vol. 1, no. 1, pp. 34–42, 2017.

[48] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification.," *Schedae Informaticae*, vol. 25, 2016.

[49] L. Li, M. Doroslovački, and M. H. Loew, "Approximating the gradient of cross-entropy loss function," *IEEE access*, vol. 8, pp. 111626–111635, 2020.

[50] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv Prepr. arXiv1609.04747*, 2016.

[51] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*, Springer, 2012, pp. 421–436.

[52] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 1–15.

[53] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[54] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, pp. 611–629, 2018.

[55] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9*, Springer, 2014, pp. 364–375.

[56] W. Ma and J. Lu, "An equivalence of fully connected layer and convolutional layer," *arXiv Prepr. arXiv1712.01252*, 2017.

[57] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural*

Networks, vol. 71, pp. 1–10, 2015.

[58] J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin, "Understanding and improving layer normalization," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[60] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[62] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[63] N. R. Gavai, Y. A. Jakhade, S. A. Tribhuvan, and R. Bhattad, "MobileNets for flower classification using TensorFlow," in *2017 international conference on big data, IoT and data science (BID)*, IEEE, 2017, pp. 154–158.

[64] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[65] M. S. Hasan, "An application of pre-trained CNN for image classification," in *2017 20th international conference of computer and information technology (ICCIT)*, IEEE, 2017, pp. 1–6.

[66] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 129–146, 2020.

[67] J. Liu, M. Wang, L. Bao, and X. Li, "EfficientNet based recognition of maize diseases by leaf image classification," in *Journal of Physics: Conference Series*, IOP Publishing, 2020, p. 12148.

[68] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE access*, vol. 9, pp. 14078–14094, 2021.

[69] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," in *International conference on machine learning*, 2021, pp. 10096–10106.

[70] C. K. Sunil, C. D. Jaidhar, and N. Patil, "Cardamom plant disease detection approach using EfficientNetV2," *IEEE Access*, vol. 10, pp. 789–804, 2021.

[71] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[72] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for

deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[73] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness," in *International conference on artificial intelligence and statistics*, PMLR, 2020, pp. 2370–2380.

[74] S. Nosouhian, F. Nosouhian, and A. K. Khoshouei, "A review of recurrent neural network architecture for sequence learning: Comparison between LSTM and GRU," 2021.

[75] M. Al-Fawa'reh, Z. Ashi, and M. T. Jafar, "Detecting malicious dns queries over encrypted tunnels using statistical analysis and bi-directional recurrent neural networks," *Karbala Int. J. Mod. Sci.*, vol. 7, no. 4, p. 4, 2021.

[76] S. Gao *et al.*, "Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation," *J. Hydrol.*, vol. 589, p. 125188, 2020.

[77] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis," *Futur. Gener. Comput. Syst.*, vol. 115, pp. 279–294, 2021.

[78] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 235–245, 2019.

[79] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 1–15.

[80] H. Parikh, H. Sawant, B. Parmar, R. Shah, S. Chapaneri, and D. Jayaswal, "Encoder-decoder architecture for image caption generation," in *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)*, IEEE, 2020, pp. 174–179.

[81] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[82] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in NLP," *arXiv Prepr. arXiv1506.01066*, 2015.

[83] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv Prepr. arXiv1508.04025*, 2015.

[84] B. Goodrich and I. Arel, "Reinforcement learning based visual attention with application to face detection," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2012, pp. 19–24.

[85] S. Ayoub, Y. Gulzar, F. A. Reegu, and S. Turaev, "Generating Image Captions Using Bahdanau Attention Mechanism and Transfer Learning," *Symmetry (Basel).*, vol. 14, no. 12, p. 2681, 2022.

[86] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, 2023.

[87] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," *J. Phys. Conf. Ser.*, vol. 1544, no. 1, p. 12033, 2020, doi: 10.1088/1742-6596/1544/1/012033.

[88] P. Rajeshwari, P. Abhishek, and P. S. | T. Vinod, "Object Detection: An Overview," *International Journal of Trend in Scientific Research and Development*, vol. Volume-3, no. Issue-3. pp. 1663–1665, 2019. doi: 10.31142/ijtsrd23422.

[89] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," *Procedia Comput. Sci.*, vol. 199, pp. 1066–1073, 2022, doi: https://doi.org/10.1016/j.procs.2022.01.135.

[90] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 7464–7475.

[91] Gillani and I. Saira, "Yolov5, yolo-x, yolo-r, yolov7 performance comparison: A survey," *Artif. Intell. Fuzzy Log. Syst.*, pp. 17–28, 2022.

[92] Z. Chen, C. Liu, V. F. Filaretov, and D. A. Yukhimets, "Multi-Scale Ship Detection Algorithm Based on YOLOv7 for Complex Scene SAR Images," *Remote Sensing*, vol. 15, no. 8. 2023. doi: 10.3390/rs15082071.

[93] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, "Natural language processing (NLP) in management research: A literature review," *J. Manag. Anal.*, vol. 7, no. 2, pp. 139–172, Apr. 2020, doi: 10.1080/23270012.2020.1756939.

[94] J. J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in *The 14th international conference on computational linguistics*, 1992, p. 1106. doi: 10.3115/992424.992434.

[95] A. Jakhotiya *et al.*, "Text Pre-Processing Techniques in Natural Language Processing: A Review," *Int. Res. J. Eng. Technol.*, vol. 09, no. 02, pp. 878–880, 2022, [Online]. Available: www.irjet.net

[96] A. S. Khan, H. Ahmad, M. Z. Asghar, F. K. Saddozai, A. Arif, and H. A. Khalid, "Personality classification from online text using machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 3, pp. 460–476, 2020, doi: 10.14569/ijacsa.2020.0110358.

[97] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review," *EURASIP J. Wirel. Commun. Netw.*, no. 1, p. 211, 2017, doi: 10.1186/s13638-017-0993-1.

[98] X. Zhou, L. Liu, X. Luo, H. Chen, L. Qing, and X. He, "Joint Entity and Relation Extraction Based on Reinforcement Learning," in *IEEE Access*, 2019, pp. 125688–125699. doi: 10.1109/ACCESS.2019.2938986.

[99] S. Li and B. Gong, "Word embedding and text classification based on deep learning methods," in *MATEC Web of Conferences*, 2021, p. 06022. doi:

10.1051/matecconf/202133606022.

[100] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, “BLEU: A method for automatic evaluation of machine translation,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318. doi: 10.3115/1073083.1073135.

[101] A. Lavie and A. Agarwal, “METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2007, pp. 228–231.

[102] R. Vedantam, C. L. Zitnick, and D. Parikh, “CIDEr: Consensus-based image description evaluation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575. doi: 10.1109/CVPR.2015.7299087.

[103] L. CY, “Rouge: a package for automatic evaluation of summaries,” *Text Summ. branches out*, pp. 74–81, 2004, doi: 10.1253/jcj.34.1213.

[104] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “Spice: Semantic propositional image caption evaluation,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, Springer, 2016, pp. 382–398.

[105] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Computer Vision--ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

Appendix thesis-related publications

The second International Conference on Scientific Research and Innovation (2ICSRI 2023)

Organized by

Oakland Publishing and Quality Conferences

Ohio Publishing and Academic Services

Cincinnati, OH, United States, August 25-26, 2023

Acceptance Letter

Date: 24/08/2023

Manuscript Title: A Deep Learning Based Approach for Image Captioning: Exploiting Priority Factor to Enhance Accuracy and Relevance

Manuscript ID: 2ICSRI-5443

Dear Ali Saleem and Israa Hadi,

We are pleased to inform you that the manuscript with the above title has been accepted for oral presentation in the **Second International Conference on Scientific Research & Innovation (2ICSRI 2023)**, organized by ***Oakland Publishing and Quality Conferences***, **and** ***Ohio Publishing and Academic Services***, which will be held in **Cincinnati, OH, United States, August 25-26, 2023**, and the publication will be in **AIP Conference Proceedings (E-ISSN:1551-7616).** The concept note and draft agenda will be sent to you at a later time.

We look forward to your attendance.

Thank you very much for your participation.

Prof. Dr. Justin Link	Assoc. Prof. Dr. Haider Raad	Assoc. Prof. Dr. Thaer Al-Jadir
Scientific Chairman	Publication Chairman	Organizing Chairman


The Second International Conference on Scientific Research & Innovation 2023
CERTIFICATE OF PRESENTATION
This certificate is awarded to
Ali Saleem and Israa Hadi
Presenter of the paper
A Deep Learning Based Approach for Image Captioning: Exploiting Priority Factor to Enhance Accuracy and Relevance
At the Second International Conference on Scientific Research & Innovation (2ICSRI 2022), Cincinnati, OH, United States, August 25-26, 2023.
Prof. Dr. Justin Link
SCIENTIFIC CHAIRMAN
Assoc. Prof. Dr. Haider Raad
PUBLICATION CHAIRMAN
Assoc. Prof. Dr. Thaer Al-Jadir
ORGANIZING CHAIRMAN

OFFICIAL ACCEPTANCE LATTER

International Conference on Engineering, Applied and Nano Sciences (ICEANS-2023)
October 25-27, 2023

Dears: Ali Saleem and Israa Hadi Ali

Congratulations, it is my pleasure to inform you that after the blind peer review, your paper entitled "Feature Selection Approach for Image Caption Generation Using an Attention Mechanism" has been accepted for presentation at the International Conference on Engineering, Applied and Nano Sciences (ICEANS-2023) conference which was taking place from October 25-27, 2023 in Soran City, Kurdistan Region- Iraq. Accordingly, your manuscript has been selected to be published in a Special Issue in journal of IEEE Explorer Digital Library indexed in Scopus.

Thank you for considering submitting your research with ICEANS-2023, please do not hesitate to contact us if you have any further questions.

Respectfully Yours,
Prof. Dr. Samir Mustafa Hamad

المستخلص

التعليق التوضحي للصور هو تحويل صورة مدخلة إلى وصف نصي (تعليق)، مما يسهم في سد الفجوة بين الرؤية واللغة بطريقة إبداعية. يمكن توثيق مجموعات متنوعة من البيانات يدويًا بتسميات مناسبة لكل صورة، ومع ذلك، يمكن أن تصبح مثل هذه العمليات مرهقة بسرعة بالنسبة لحجم المجموعة البيانية أو عدد المجموعات البيانية. لذلك، يوجد حاجة ماسة لتطوير نظام توليد تلقائي لتسمية الصور بشكل كاف. الهدف الرئيسي لهذه المهمة هو إنشاء وصف شبيه بوصف الإنسان للمحتوى والسياق (الدلالي) للصورة، والذي يشمل في كثير من الأحيان تفاصيل حول الكائنات والأفعال والعلاقات داخل المشهد.

تقترح هذه الرسالة نظامًا للتعليق على الصور بناءً على الدلالة لتعظيم دقة توليد الوصف. يستخدم النظام معمارية (encoder-decoder) التي تعتمد على آلية الانتباه لتحفيز المراقبة الانتقائية لميزات الصورة المهمة من مجموعة من الميزات المتاحة. يتكون النظام من ثلاث مراحل رئيسية: المعالجة المسبقة، استخراج الميزات (التشفير)، وفك تشفير الميزات. في المرحلة الأولى، تخضع كل من الصورة والبيانات لعمليات معالجة مسبقة. في المرحلة الثانية، يتم استخراج نوعين من الميزات: الميزات العامة (deep visual features) المستخرجة من نموذج مدرب مسبقًا والميزات الخاصة (object features) المستخرجة من نموذج اكتشاف الكائنات المدرب مسبقًا ايضاً.

لتحسين دقة النظام المقترح، تستغل هذه الرسالة نموذج اكتشاف الكائنات لحساب ميزة جديدة تسمى "Priority Factor" الذي يعظم تنشيط الكائنات الهامة دلاليًا في صورة معينة. وأخيرًا، يتم دمج هذه الميزات المتنوعة معًا في خطوة الدمج. على عكس الطرق السابقة، يحدث الدمج قبل تقليل الميزات، مما يؤدي إلى مصفوفة ميزات غير معالجة. المرحلة الثالثة هي شبكة عصبية عميقة تتألف من ثلاث كتل معالجة: التقليل، الانتباه، واللغة. خلال هذه المرحلة، يتم تمرير مجموعة الميزات المدخلة من خلال طبقات إضافية (FC layer، LSTM) للمعالجة المستندة إلى المتجهات لتعلم ارتباطات مفردات الميزات وبالتالي إنشاء نموذج مدرب.

تم إجراء التجارب باستخدام مجموعة بيانات شائعة في computer vision (MSCOCO)، وتم تقييم أدائها باستخدام ثماني مقاييس. تثبت النتائج فعالية دمج الميزات الخاصة، مما يؤدي إلى تحسينات في مقاييس التقييم المختلفة، بما في ذلك BLEU-1، BLEU-2، BLEU-3، BLEU-4، METEOR، CIDEr، ROUGH-L، وSPICE، بشكل خاص بعد إضافة ميزة "Priority Factor" المقترحة إلى object

features ، مما يؤدي إلى إنشاء وصف يتناسب مع المحتوى الدلالي الرئيسي للصورة. بالإضافة إلى ذلك، يتفوق هذا النظام على أربعة من أحدث الطرق في (76.2) BLEU-1، (60.1) BLEU-2، BLEU-3 (47.2)، وSPICE (15). ونتيجة لذلك، يمكن أن تساعد نتائج هذه الرسالة الأفراد المكفوفين أو ضعاف البصر من خلال إنشاء أوصاف للصور الموجودة على الويب أو في المستندات أو في منشورات وسائل التواصل الاجتماعي، مما يسمح للأفراد ضعاف البصر بفهم محتوى الصور التي لا يمكنهم رؤيتها.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعــة بابـل

كلية تكنولوجيا المعلومات

قسم البرمجيات

التعليق التوضيحي للصور المستند على الدلالة باستخدام التعلم العميق مع آلية الانتباه المرئي

رسالة مقدمه الى

مجلس كلية تكنولوجيا المعلومات – جامعة بابل كجزء من متطلبات نيل درجة

الماجستير في تكنولوجيا المعلومات / البرمجيات

من قبل

علي سليم حليم جابر

بإشراف

أ. د. اسـراء هـادي علـي

1445هـ

2024 م