

Automated Arabic Image Caption Generator for assisting disabled Arabic Speakers

Project draft

Course Name:

Programming in Python

Submitted by:

Leen Alnajjar

Abstract

The process of generating content from images is considered to be a very difficult and challenging task, but at the same time, it is very impressive because it helps blind people to have a better understanding of their surroundings. The importance of automatic image caption generator for children with learning difficulties come from the visibility of transferring the captions later on to voice by using text-to-speech technology in order to describe the whole image for those people. Although it is considered a critical mission, the resources and projects regarding automatic Arabic image captioning are rare and need further improvements. According to that, this work aims to build an Automated Arabic Image Caption Generator model to give a short explanation of the images in the Arabic language. The model will be constructed from both CNN which aims to figure out features from figures and the RNN which will take the features from the CNN and provide suitable descriptions for these features. The data itself is constructed from images and text or description for each image.

Contents

Abstract	ii
List of Figures	v
List of Tables	vii
Introduction.....	8
1.1 Background	8
1.2 Aim and objectives	10
Chapter two: Literature review	11
2.1 Arabic image captioning	11
2.2 Main challenges in Arabic image captioning.....	13
2.3 Related work	15
2.3.1 Find out the massive work on Arabic image captioning.....	16
2.3.2 Conclusion of the related work	17
Methodology	19
3.1 Methodology structure	19
3.2 Data gathering	20
3.2.1 Find out all available data source and their limitations	21
3.2.2 Data selection and understanding	22
3.3 Data importing and reading	23
3.4 Image pre-processing	24
3.4.1 Select the architecture of convolution Neural Network.....	25
3.4.2 Import the libraries for the CNN model.....	26
3.4.3 Build the CNN model	27
3.5 Text pre-processing using NLP techniques	31
3.5.1 Load and read the document caption file	31

3.5.2 Prepare the text	31
3.5.3 Visualize images with their corresponding captions	33
3.5.4 Text cleaning.....	34
3.6 Building the LSTM model	38
3.7 Scoring method	41
Discussion of results	42
4.1 Encoder model discussion.....	42
4.2 The decoder model discussion	43
Conclusion and future work.....	46
Bibliography	47

List of Figures

Figure 1: An example of image captioning model and its results (Alazzam, 2022).....	9
Figure 2: The expected stages of the project	10
Figure 3: The general layout of the image captioning model	11
Figure 4: Different attention based model in the encoder of the image processing (Pedersoli, et al., 2021)	12
Figure 5: Different model and techniques type of image captioning (Hossain, et al., 2019)	13
Figure 6: An example of the lexical/morphological sparsity (Obeida ElJundi, 2020)	15
Figure 7: The steps of finding out the research gap.....	16
Figure 8: The structure of the project methodology for encoder-decoder	20
Figure 9: Steps of data gathering	20
Figure 10: An example of images with their related captions using Pexels (pexels, 2023)	21
Figure 11: An example of image with its caption from Shutterstock (shutterstock, 2022)	22
Figure 12: The differences between the Arabic caption using COCO and Flickr8k for the same figure (Emami, et al., 2022)	23
Figure 13: The required libraries for the project.....	24
Figure 14: The required libraries for the project (NLP part)	24
Figure 15: The construction of the CNN model (upgrad, 2022).....	25
Figure 16: the flow chart for selecting the appropriate CNN model	26
Figure 17: All the libraries that required to build the MobileNet_v2 CNN model.....	26
Figure 18: The dropped part of the MobileNetV2 per-trained model (Akay, et al., 2021)	28
Figure 19: The method of checking the images size.....	28
Figure 20: Loading the MobileNet_V2 model.....	29
Figure 21: The extracted features before moving it to external file	30
Figure 22: The extracted features of the images.....	30
Figure 23: Load and read the document caption file	31
Figure 24: Load the text to python environment	31
Figure 25: The text variable in which the function will loop through.....	32
Figure 26: The return captions file after applying the function	32
Figure 27: The step of visualize images with their corresponding captions.....	33
Figure 28: The results of images visualization with their captions	33

Figure 29: Text cleaning steps	34
Figure 30: Remove the punctuations	34
Figure 31: The text after applying the stop words and stemmer.....	35
Figure 32: The meaning of the region feature	35
Figure 33: The rich morphology of Arabic language	36
Figure 34: The step of removing the single characters and definition words	36
Figure 35: The step of converting the numeric to words	37
Figure 36: Harakat and their English meaning	37
Figure 37: Remove the harakat from the text	37
Figure 38: The step of adding the start and end point to the model.....	38
Figure 39: Convert the cleaned captions into list.....	38
Figure 40: Create a tokenizer	38
Figure 41: Test the vocabulary size	39
Figure 42: Create sequence of images, input sequences and output words for image.....	39
Figure 43: Define the LSTM model.....	40
Figure 44: Scoring method.....	41
Figure 45: Arabic image captioning consists of the encoder (on the left) and the decoder (right side) (margotwagner, 2023)	42
Figure 46: A sample of the results of the pre-training CNN model.....	43
Figure 47: The most frequent 50 words in the captions.....	44
Figure 48: Sample results one.....	45
Figure 49: Sample results two.....	45

List of Tables

Table 1: Conclusion of the related work.....	17
Table 2: The advantages and disadvantages of using Flickr8k_Dataset over the other datasets (medium, 2022).....	23
Table 3: The differences between the pre-trained models (keras, 2023).....	25
Table 4: The importance of each library in the MobileNet_v2 model and images feature extraction	27
Table 5: The results of text cleaning.....	43
Table 6: The images captions with their boundary	44

Introduction

This chapter provide a brief introduction for automatic image captioning, this include but not limited to a brief description of the process, the definition of image captioning, and its important. Furthermore, the project opportunity and the reasons behind working on the project were also mentioned. Lastly, the project aim and objectives were also discussed.

1.1 Background

As mentioned by [1], the percentage of children with learning difficulties are increasing rapidly in the Arabic world. As recorded by [2], in 2019, 6% of the children aged between 5-17 in Jordan are suffering from mild to severe learning difficulties, on the other hand , 1% suffering from an acute disability. Although this high percentage, these children are often struggling to go to school as a result of a lack of awareness and deficiency of accessible learning resources for children with . The main issue is these children require special treatment and especially in their early childhood years. [1] claimed that Children with a verity type of learning difficulties are unable to distinguish between the different words or form sentences. In addition to this, it is hard for them to understand the correct meaning of context or either create relations between similar words.

According to the previously mentioned facts, the project aims to help these children and reduce the gap between them and society by proposing a new learning technique that is basically dependent on technology engagement in the learning process. This will be done by using the image captioning technique, with the aim of using image captioning is to increase the student's interaction with their teachers and enhance their communication during their rehabilitation process by helping them to create a relationship between images and the related context. Image captioning is the process of generating description automatically from figures using natural language processing. Furthermore, it is known to be one of the most attractive topics in AI because it can be used in different applications such as disease diagnosis, in education such as teaching children words and describing images for visually impaired people [3]. Figure 1 presents an example of image captioning model and its results.

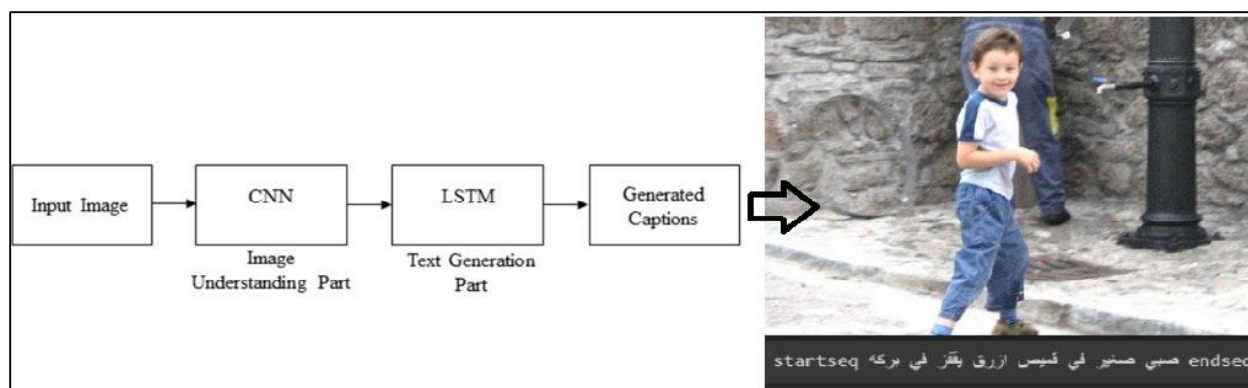


Figure 1: An example of image captioning model and its results [4]

A study done by [5] used the image captioning in Chinese language to enhance the abilities of children with Autism Disorder during their rehabilitation process, the results show that the children attention and communication skills and their ability of describing object were increased.

This study was the base point of the project, as stated on the study, the image captioning technique, is one of the best options that can be lean on in order to improve the performance of the children with learning difficulties. Based on the above fact, the expected outcomes of implementing Arabic image captioning in the rehabilitation process of the children with learning difficulties can be listed below:

1. Arabic image captioning can be a useful tool for children with learning difficulties because it can provide them with a visual representation of the text they are reading. This can help to improve their understanding and comprehension of the material, as well as help them to make connections between the words and the corresponding images.
2. For children with learning difficulties, the use of visual aids can be particularly helpful in facilitating learning and comprehension. By providing both visual and textual information about the subject matter, Arabic image captioning can help to make the material more accessible and engaging for children with learning difficulties.
3. In addition, the use of Arabic image captioning can also help to improve children's vocabulary and language skills, as they are exposed to new words and concepts through the captions. This can help to enhance their understanding of the material and improve their overall language skills.
4. Overall, Arabic image captioning can be a valuable resource for children with learning difficulties, as it can help to improve their understanding and comprehension of the material, as well as enhance their language skills.

Figure 2 illustrates the expected stages of the project. As shown in the figure, the current of the proposed project aims to build up an accurate and useable Arabic image captioning model that can be implemented in the future at the centres that are specified in providing learning services to children with learning difficulties.

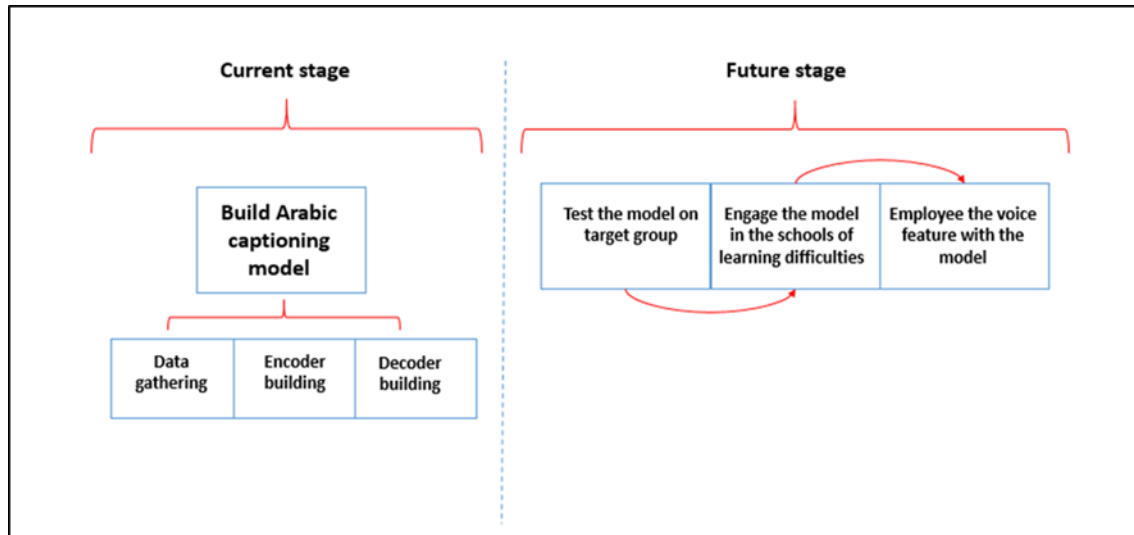


Figure 2: The expected stages of the project

1.2 Aim and objectives

The aim of this project was to develop an Arabic image captioning model for assisting Arabic Speakers with learning difficulties using transformer. This aim was achieved by following the below objectives:

- Review the previously developed Arabic image captioning models through the literature review.
- Investigate the limitation of the limitations of the image captioning models (CNN-LSTM models).
- Build transformer model for image captioning model.
- Set a recommendations for future work.

Chapter two: Literature review

Since it was used back to the early 2000s, huge effort has been done to improve the image captioning techniques by enhancing its capabilities of the feature extraction in order to produce appropriate captions that imitate the human capabilities of describing image. Although automatic image captioning has become an active area of research but the research on Arabic image captioning was very rare, this is due to the fact that the Arabic language is very difficult and highly inflected language when compared to other languages, as a consequences of this, it is a little bit challenging to generate natural language processing (NLP) technologies that can accurately understand and generate Arabic text. This chapter spot the light on the definition of image captioning and the architecture of the model, then it discuss the main challenges that are associated with the Arabic image captioning models, lastly it review the previous literature regarding the Arabic image captioning.

2.1 Arabic image captioning

As defined by [6], the goal of image captioning is to generate a natural language description of the contents of an image using both a system that can understand visual information and a language model that can generate grammatically correct sentences. This requires the use of a visual understanding system and a language model. Figure 3 shows the general layout of the image captioning model.

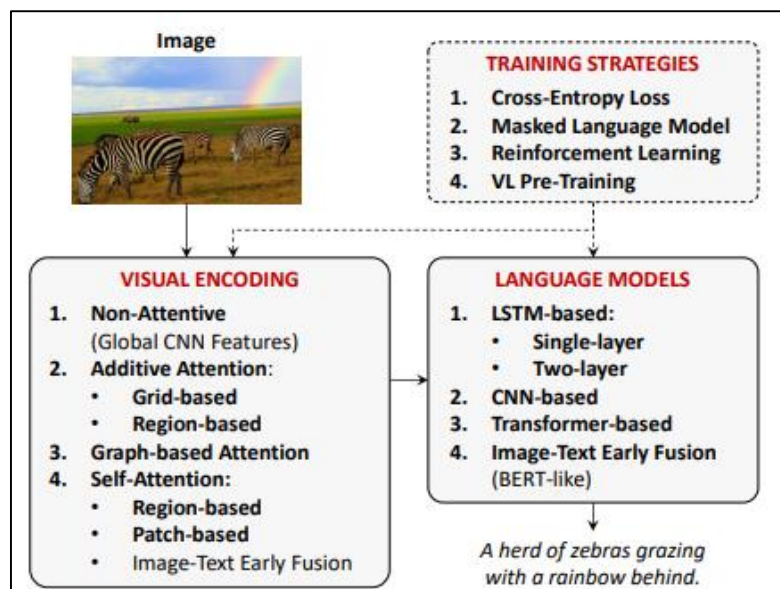


Figure 3: The general layout of the image captioning model

As seen in Figure 3, the image captioning model is usually constructed from two parts, these are the encoder and the decoder.

CNNs are commonly used in image recognition tasks because they are able to learn hierarchical .The encoder is responsible of providing the visual content of the images in the model, by extracting a specific features from the model [7].

One technique that was used to improve the performance of the CNN and image captioning was the usage of the Attention mechanisms, which allow the model to focus on specific parts of the image when generating the caption. The mechanism of attention have become more popular in recent years because they can improve the performance of image captioning models [8]. Figure 4 shows different attention based model in the encoder of the image processing.

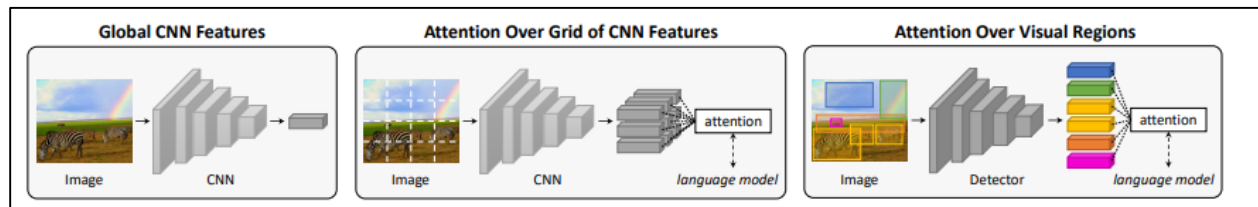


Figure 4: Different attention based model in the encoder of the image processing [8]

In other words, the attention mechanism focuses on the important aspect (pixels) of the image rather than focussing on the global features, this can help in explain the relationship between words in the sentence [9].

The second part of the image captioning model is the decoder or the language model, the aim of the language model is to provide a caption based on the extracted features from the CNN model [10]. As cited by [11], usually, in the decoder part the used model would be RNN -LSTM that the input of the model would be the output from the natural language processing model, the importance of the LSTM is to remember the long sentences. To be rephrased, the use of LSTM with RNN in image captioning models can lead to better performance and more accurate and descriptive image captions.

There are many types of image captioning methods, but all of these techniques are mainly constructed from encoder and decoder in which the encoder extract the main features from images and the decoder convert these features into readable sentences [10]. Figure 5 shows different model and techniques type of image captioning.

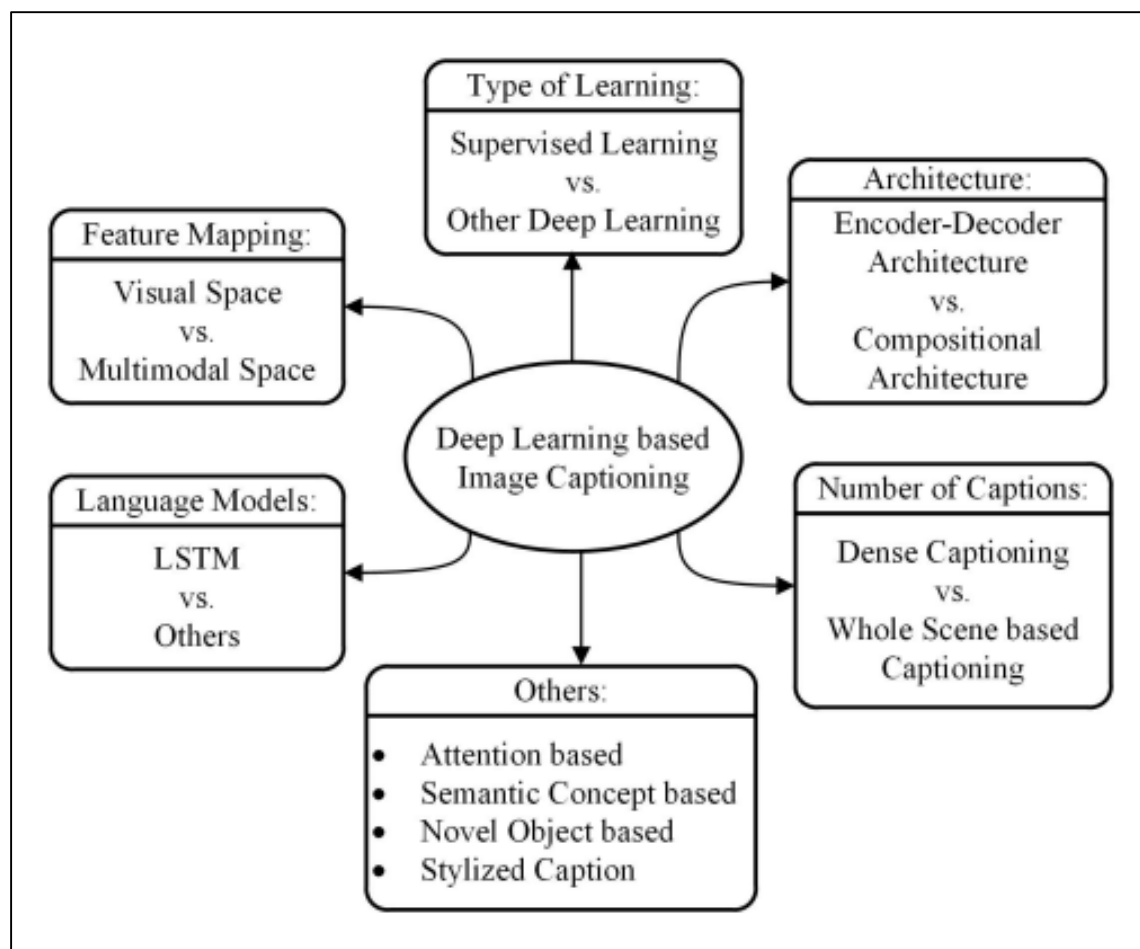


Figure 5: Different model and techniques type of image captioning [10]

The following sections spot the light on the related work and their models, in order to have better understanding of the model architecture types.

2.2 Main challenges in Arabic image captioning

As mentioned earlier, the Arabic image captioning technique in general are considered to be a promising field due to its important in different aspects, the importance and application of Arabic image captioning are mentioned below [12] [13]:

- **Education**

Arabic image captioning is considered one of the significant resources, for Arabic language speakers, this because it can be used to provide them by a visual representation of the text they are reading, accordingly, it can lead to enhance their understanding of the textual material.

- **Accessibility**

Image captioning can also help blind Arabic speakers, by providing those with the text describe the whole images and videos around them, and then the text can be read using text speech tools. By providing text-based descriptions of the content of images and videos, image captioning can help to make this content more accessible to people who are unable to view it directly.

- **Translation**

Some image captioning model can be used to translate the captions on images from one language to other, this might lead to easily access and understand different content that could be not available in their language. An example of this, most of the disease diagnosis are available in English, with Arabic image captioning it allows people to understand the content of the images.

- **Social media**

Arabic image captioning can also be useful for users of social media platforms, who often share images and videos with their followers; in 2019, Meta Company received around 400 million images from Facebook and Instagram. By providing captions for these images and videos, image captioning can help to make them more accessible and engaging for a wider audience. [14]

Although there are around 400 million Arabic speakers globally [15], and significant importance of the Arabic image captioning, but there were only few resources studied it, and they still need for further improvement [14]. In addition to the lack of dataset, there are many reasons behind the deficiency of Arabic image captioning models, and these reasons are listed below [16]:

1. There are less demands on Arabic language with comparison with other languages, this because two main factors, initially, the Arabic language is not popular in the market place such as English language. Secondly, lack of interest of using Arabic image captioning techniques either globally or regionally. For example, in 2020 there were only 3 papers studied the Arabic image captioning, despite the fact that the image captioning was firstly introduced in early 2000s.
2. Regardless the complexity of the image captioning model in general, Arabic language is very difficult, this because unlike most of other languages it requires unique writing system from the right to the left. Furthermore, the grammar and the lexical/morphological sparsity in the Arabic language is considered to be demanding task for those who want to build an Arabic image captioning model. Figure 5 presents an example of the lexical/morphological sparsity.

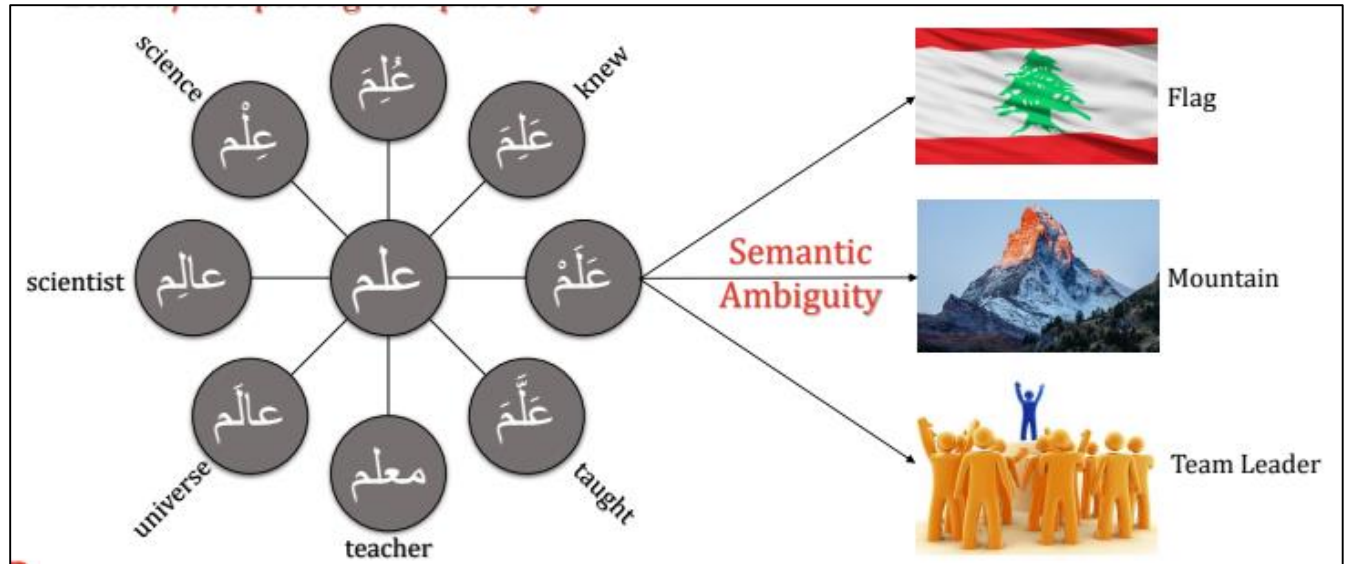


Figure 6: An example of the lexical/morphological sparsity [16]

- In addition, the availability of high-quality Arabic language data may be more limited compared to other languages, which can also make it more difficult to develop and train effective image captioning models [17].
- There may be cultural differences that make it more difficult to develop image captioning systems for the Arabic-speaking market, such as differences in the way that people describe images or in the types of images that are most common [18].

2.3 Related work

Although image captioning is one of the most popular fields today and there is extensive work on it, but Arabic image captioning is very rare and the resources of data set are very limited. At very initial level the Arabic image captioning through the previous literature was reviewed, it was found that among the 122000 results in the search engine only few resources was directly related to Arabic image captioning. Figure 7 shows the steps of finding out the research gap.

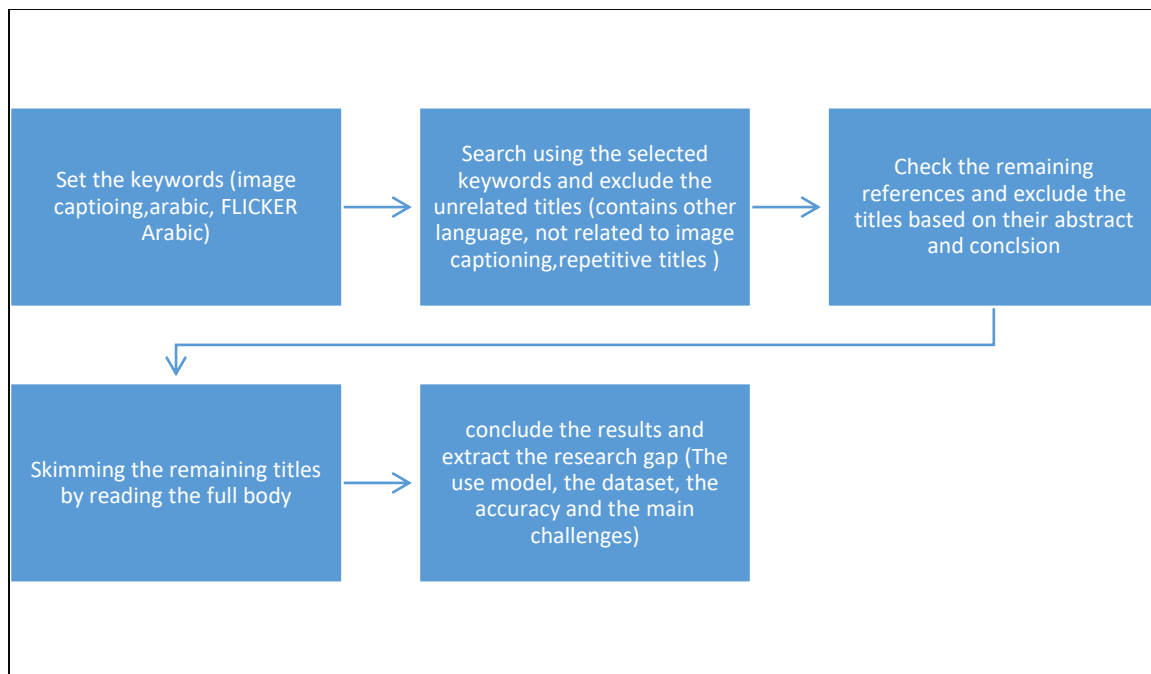


Figure 7: The steps of finding out the research gap

As mentioned earlier, rare work has been done regarding the image captioning since the first time of dealing with the image captioning and its related model. In order to get better understanding of what has been implemented recently, the previous literature of Arabic image captioning was reviewed and checked then it was concluded in the below subsections.

2.3.1 Find out the massive work on Arabic image captioning

Though out the researching process, there were different resources that were followed in order to find out larger amount of research regarding the Arabic image captioning, these are listed below:

1. Search online databases

Different online database were used to find out the most related work regarding the Arabic image captioning such as the ACM Digital Library, IEEE Xplore, and the arXiv preprint server, contain a large number of research papers on a wide range of topics, and many other database to obtain the dataset especially that the required data was not share in public. The desired papers and researches using the previously mentioned keywords.

2. Academic search engines

The following step was to use the reliable scientific search engines such as Google Scholar, and Google data. Those search engine play major role in this work, because it was used to find the related work based on the keywords and number of citation.

3. Search within specific journals

One of the used method was to search over the Journals based on the field of the study, this was done by browse in the journal on the topic, this method was the least effective method because not all journals can be determined based on the topic, and accordingly this led to reduce the results.

4. Use citation searching

Citation searching involves searching for papers that have cited a specific paper or research study. This can help to find other papers that are related to a specific topic or field of study. It can use citation search tools, such as the Web of Science, to find papers that have cited a specific paper.

2.3.2 Conclusion of the related work

In order to provide better understanding of the Arabic image captioning, the previous work was reviewed and checked, the related wok can be concluded as presented in Table 1.

Table 1: Conclusion of the related work

Title	References	Used dataset	Used model	Accuracy(BLEU score)
Automatic Arabic Image Captioning using RNN-LSTM-Based Language Model and CNN	[19]	The dataset was relatively small, the images were taken from the Flickr and COCO but their captions were generated using three different sources as following: 2101 of captions were translated using Google translator, 1170 was described by Arabic proficienals using a CrowdFlower Crowdsourcing service and 150 image captions were translated by human translators.	For image processing stage CNN based on VGG-16 was used. For the language model. A single hidden RNN-LSTM layer was used for the language model.	The recorded BLEU score was 46.2 which is higher than the previous references by 10%.

AraCap:A hybrid deep learning architecture for Arabic Image Captioning	[20]	COCO and Flickr30k dataset	The developed model was basically depends on attention model and multi object detection. With LSTM-RNN text model because the RNN predict words in long-range dependencies	satisfactory accuracy indicates above 60%
Arabic image captioning using ResNet50	[11]	Flickr8k dataset	ResNet50 for images and LSTM-RNN for text. it focuses on the main content of the target image man, girl, dog, or car instead of focusing on the all image features.	minimum loss value in the testingphase is 0.80 at epoch number 29

In conclusion, it can be said that there are a wide number of image captioning models that have been developed in order to enhance the performance of image captioning. However, there has been a rare number of research in the field of Arabic image captioning due to the challenges associated with the data and the language itself. Although scientists have tried to improve Arabic image captioning (AIC), it still needs further improvement. According to this, the following sections illustrate the contribution of the project in the enhancement process of AIC.

Methodology

In recent decades, image captioning has undergone a dramatic revolution. However, Arabic image captioning still faces a number of limitations, including a lack of data and challenges associated with natural language processing (NLP). Despite the efforts of scientists over the past few years, the developed models still require further enhancement. The aim of this project was to build a CNN-LSTM Arabic captioning model with attention, and then to enhance the model by replacing the decoder and encoder with a transformer. The model was tested using the BLUE score model. The project was carried out using a set of organized steps and approaches in a Python environment. The main steps and tools used in the project are described in the following subsections.

3.1 Methodology structure

As mentioned earlier, the project was built based on a series of steps in order to achieve the ultimate aim of the project. This section provides a well-structured introduction to each stage of the project's methodology. Figure 8 shows the structure of the methodology.

At the very initial level, the first step was to find the data. The following step was to understand the data and its construction. After that, the data was processed using pre-processing techniques in natural language processing (NLP) and image processing. Then, a CNN-attention based model was built to extract the required features. The output of the CNN model was used as input to the LSTM model. The final step was to test the BLUE score and obtain the model's accuracy. However, the results were not satisfactory, so as a part of the improvement stage, the model was replaced by a transformer. The model was then tested again, and recommendations for future work were obtained. The following sections provide an in-depth review of these steps and the tools, libraries, and methods used during the implementation phase.

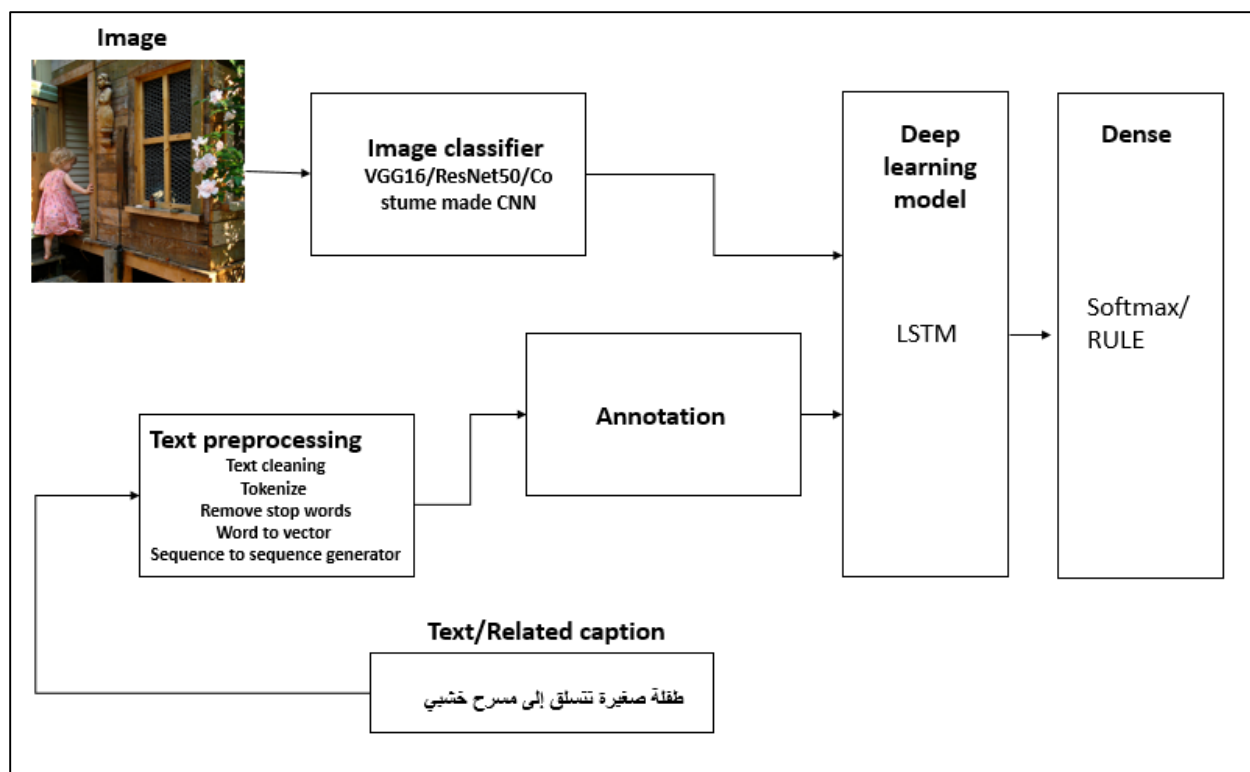


Figure 8: The structure of the project methodology for encoder-decoder

3.2 Data gathering

As mentioned earlier, it was not easy to find out the data, in other words, it was the main challenge of the project. Data selection passed through several steps in order to obtain more accurate results, steps of data selection are presented in Figure 9.

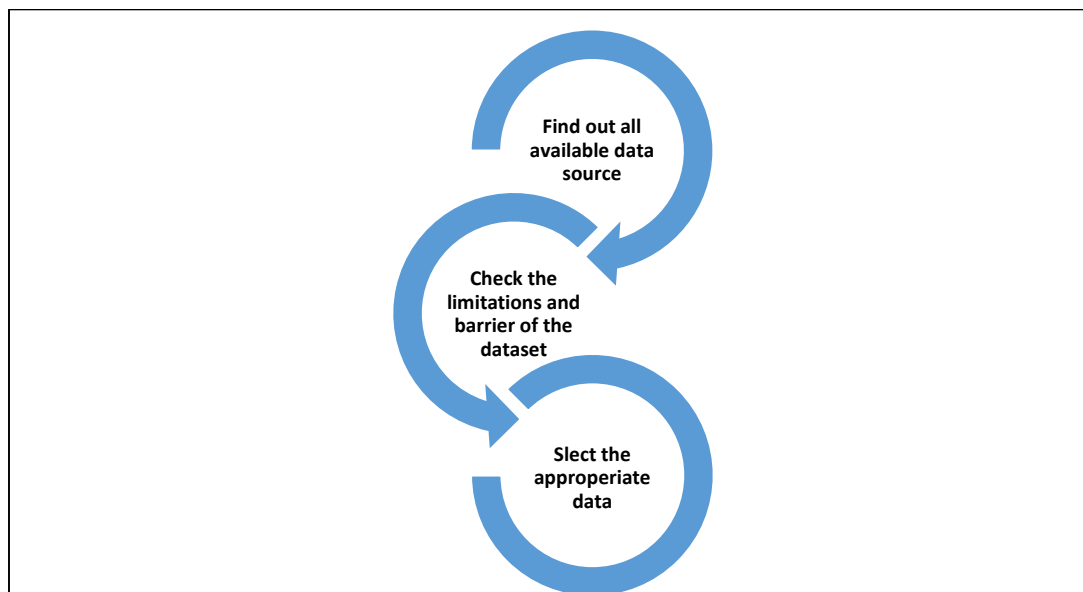


Figure 9: Steps of data gathering

3.2.1 Find out all available data source and their limitations

There are wide sources from which data can be obtained. Some of these sources may not be publicly available. Although this, all data sources are mentioned below [21]. The most appropriate data, along with a detailed justification, is mentioned at the end of this section

- **Open data repositories**

There are different open sources data repositories over the internet that provide access to large data set with their corresponding captions, which are directly related to the project aim. Some examples include the MS-COCO dataset, the Flickr30k dataset, and the Arabic Image Captioning dataset. These datasets can be downloaded for free and used for research and development purposes in most of the previously mentioned literature. It is important to mention that the dataset of Flickr 8 was translated to Arabic based on the contribution of the significant work done by [16].

- **Online image databases**

In addition to the Open data repositories there are online image databases that over millions of images with their captions, unfortunately, these captions are only provided in English and they need to be translated to be used in image captioning project either by using popular translation website. An example of these website is Unsplash, Pexels, and Pixabay, It is important to note that these websites are free and legal to use for public. Figure 10 shows an example of images with their related captions using Pexels.

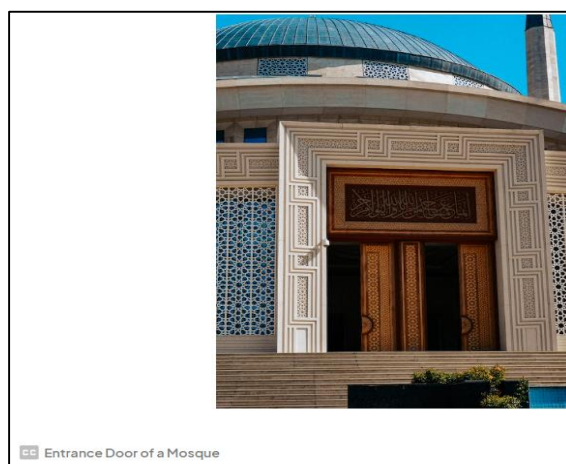


Figure 10: An example of images with their related captions using Pexels [22]

- **Professional image libraries**

An example of these source is Shutterstock and Getty Images, also offer a large collection of images and captions in various languages, but these are not translated to Arabic language.

However, a license is required to use these images and captions Arabic image captioning projects. Figure 11 shows an example of image with its caption from Shutterstock.



Figure 11: An example of image with its caption from Shutterstock [23]

- **Custom dataset**

The last method is custom made dataset based on Arabic experts, although using this method might increase the accuracy, but it is considered a time and effort consuming method, because the experts need to generate at least three captions taking into consideration the previously mentioned challenges associated with Arabic language. As previously mentioned in literature review chapter, there were number of papers mentioned that they use costume made dataset but unfortunately, these data set were not open to public [19].

3.2.2 Data selection and understanding

The selection process of the data is a trade-off multiple things, hence the dataset in the options from 2-4 are not suitable based on the project time frame and the project aim (which is basically aims to increase the accuracy of the results, the remaining solution was the first option which includes the either COCO dataset or the Flickr8k, ultimately, the Flickr8k_Dataset was selected. The data is constructed from two folders, these are [24]:

- Flickr8k_Dataset – Dataset folder which contains 8091 images.
- Flickr_8k_Arabic_text – Dataset folder which contains text files and captions of images.

There are also other large dataset such as Flickr_30K and MSCOCO dataset but due to time constrains and computer limitations the Flickr8k_Dataset will be used initially, the other dataset will be used for further enhancement [24]. It should be noted that the Arabic-COCO dataset is translated from English, and according to study done by [25] there were 46% out of 150 randomly

selected captions of the Arabic COCO are not accurate which might confuse the model. As a result of this conclusion, the COCO dataset was excluded from the analysis and it won't be used as neither as test nor as validation. On the other hand Flickr8k_Dataset was initially translated on Google translate and then checked by Arabic professionals. Figure 12 exhibits the differences between the Arabic caption using COCO and Flickr8k for the same figure. [26].



Figure 12: The differences between the Arabic caption using COCO and Flickr8k for the same figure [26]

Table 1 compare the advantages and disadvantages of using Flickr8k_Dataset over the other datasets.

Table 2: The advantages and disadvantages of using Flickr8k_Dataset over the other datasets [27]

Advantages	Disadvantages
<ul style="list-style-type: none"> It is easy to train the model because the data is small in size. The data is labelled, (It has 3 caption for each image). Free to download. 	<p>The accuracy of the captions might not be high when compare it to the large dataset</p>

3.3 Data importing and reading

To build an Arabic image captioning system, we need to import the necessary libraries and then the data, which consists of images and text. The first step is to create new environment in the anaconda that contains the tensorflow and keras libraries. Figure 13 shows the required libraries in the project. It should be noted that most of these libraries need to be installed separately before importing them to the environment.

```
#Import the required libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle
import os
from os import listdir
from nltk.translate.bleu_score import sentence_bleu, corpus_bleu
from nltk.translate.bleu_score import SmoothingFunction
import bleu
from bleu import Bleu
from bleu_scorer import BleuScorer
from tqdm import tqdm_notebook, trange
import keras
from keras_tqdm import TQDMNotebookCallback
print(keras.__version__)
from bidi import algorithm as bidialg
import arabic_reshaper
from keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.mobilenet_v2 import preprocess_input
from keras.models import Model
import tensorflow as tf
from PIL import Image
```

Figure 13: The required libraries for the project

```
#These for NLP:
import nltk
from nltk import word_tokenize
import pyarabic.araby as ar
import string
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import SnowballStemmer
from pyarabic.araby import strip_harakat
from pyarabic.araby import strip_tashkeel
from pyarabic.araby import strip_diacritics
from pyarabic.araby import strip_tatweel, strip_shadda
from pyarabic.araby import normalize_ligature
import pyarabic.number
```

Figure 14: The required libraries for the project (NLP part)

3.4 Image pre-processing

The data pre-processing was done in two stages, the first was the CNN itself and the second is the text pre-processing on the text which represents the captions of each image. This section presents an overview of the tools which was used in the pre-processing steps.

3.4.1 Select the architecture of convolution Neural Network

The first part is to build the CNN model in order to extract the main features of the images to enable the model of predicting the required captions. Figure 15 presents the construction of the CNN model.

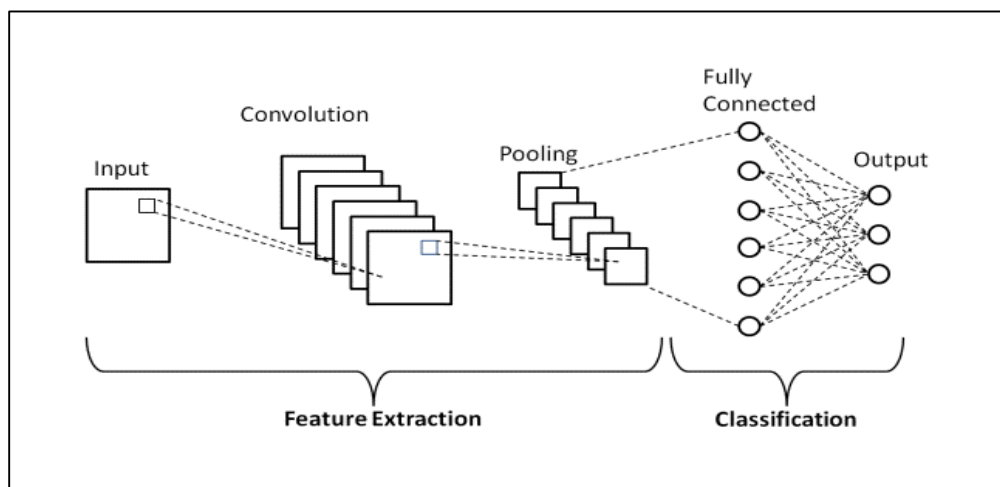


Figure 15: The construction of the CNN model [28]

As seen in Figure 15, the CNN is constructed from three layers these are convolutional layers, pooling layers, and fully-connected layers. The CNN can be built either by using trained CNN model (VGG-16 or ResNet50 or MobileNetV2) or by building costume made CNN. Table x illustrate the differences between the pre-trained models.

Table 3: The differences between the pre-trained models [29]

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9

The decision to select the best CNN was a trade-off between several factors, which are listed below:

- Building a custom CNN model can be considered time-consuming for image captioning, as the goal is to extract features from images rather than classify them. Additionally, there

may be limitations on time and resources, making it more practical to use a pre-trained model.

- When selecting a pre-trained model, factors such as the size of the dataset, computational power, and other considerations were taken into account. Figure 16 presents the flow chart for selecting the appropriate CNN model.

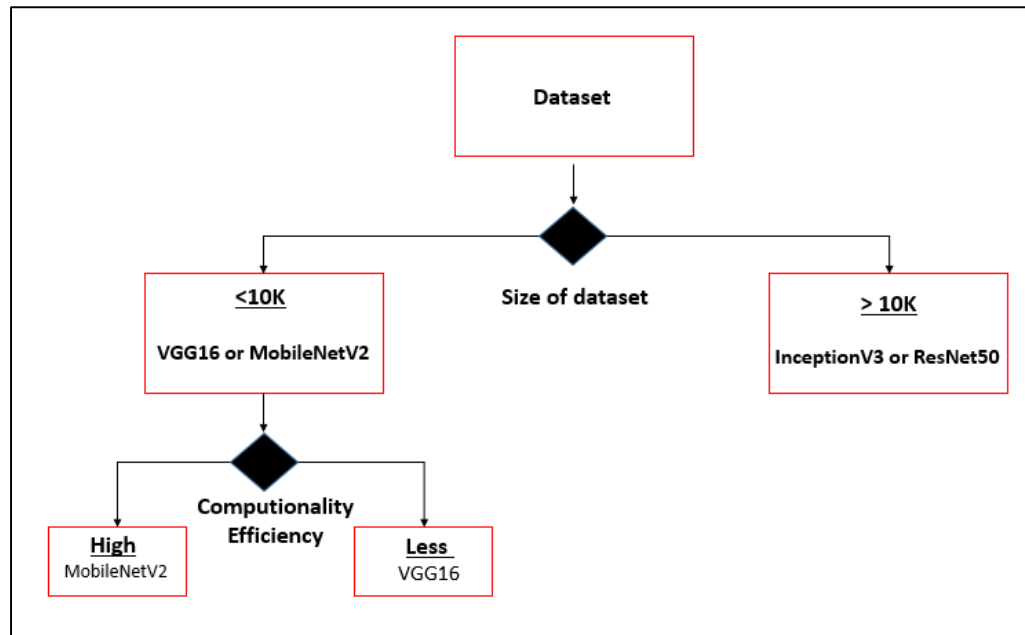


Figure 16: the flow chart for selecting the appropriate CNN model

As both VGG16 and MobileNetV2 had the same accuracy [30], the decision was made to use MobileNetV2 as the pre-defined model for transfer learning.

3.4.2 Import the libraries for the CNN model

There were several libraries that was used in the project to build the pre-trained CNN model, this subsection provided an overview of these libraries and justification of using them. Figure 17 shows all the libraries that required to build the MobileNet_v2 CNN model.

```

import os
from os import listdir
import pickle
from keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.mobilenet_v2 import preprocess_input
from keras.models import Model
  
```

Figure 17: All the libraries that required to build the MobileNet_v2 CNN model

Table 4 concludes the importance of each library in the MobileNet_v2 model and images feature extraction.

Table 4: The importance of each library in the MobileNet_v2 model and images feature extraction

Library	Justification of use	Reference
os.listdir	To get list of all files in the specified directory, in the project, the aim was to get list of all images that located in the ('C:\\Users\\hp\\Desktop\\Personalfiles\\HTU\\python\\project\\working file') directory.	[31]
Pickle.dump	The module pickle will be imported, then the function dump will be used to save an objects from the python to the desktop, this case the extracted features will be saved on the PC using Pickle dump.	[32]
Load image	It is used to load images and return them in PIL format, the PIL format is library in python that provide further images editing capabilities. This is done by providing the images path/file and the target size of this images to the load function.	[33]
Image to array	The image to array is used to convert the tensors of the image into array. This because the model will be trained on the images features.	[34]
Pre-process input	The preprocess_input() function from the keras.applications.mobilenet_v2 module is used to preprocess the input images before they are passed through the MobileNetV2 model.	[35]
Model	This class is part of the Keras functional API, which is used to define, create, and train deep learning models.	[36]

3.4.3 Build the CNN model

As mentioned previously, images are the input of the proposed model and should be provided in the form of a vector. Each image should be converted into a fixed size vector in order to feed it into the neural network model. Hence, the classification process occurs at the last two layers, which were dropped to ensure that the output will only be the features instead of classes. Figure 18 presents the dropped part of the MobileNetV2 pre-trained model.

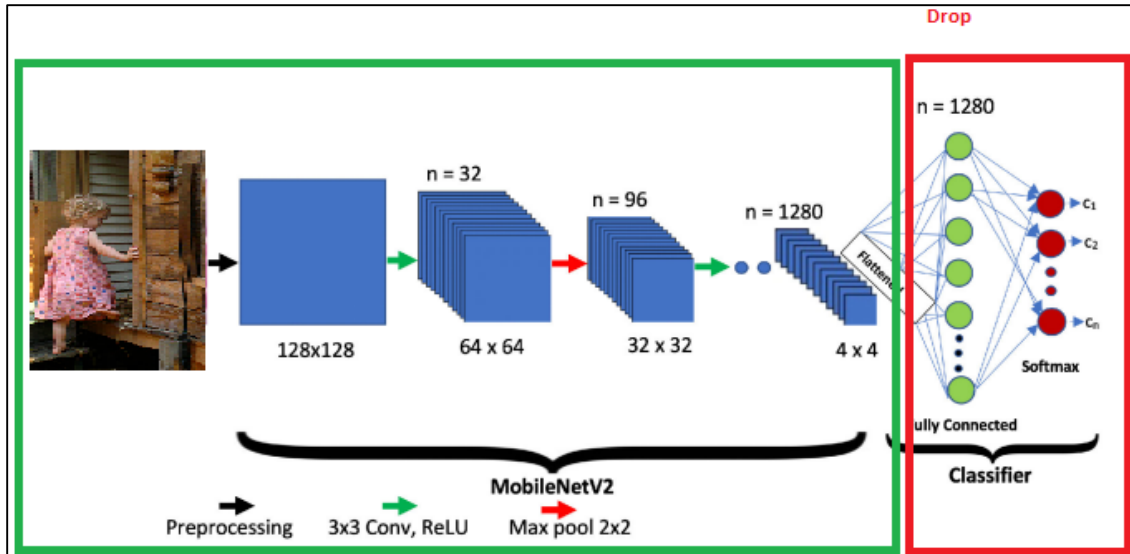


Figure 18: The dropped part of the MobileNetV2 pre-trained model [37]

The first step was to build the load the pre-trained model MobileNet_V2, then the following step was to extract the features from the images.

1. Check the size of the images

The first step before building the model was to check the images size (minimum height and width). Figure 19 shows the method of checking the images size.

```
# Set the directory containing the images
directory = 'C:\\Users\\hp\\Desktop\\Personal files\\HTU\\python\\project\\Dataset\\Flicker8k'
# Initialize the minimum height and width
min_height = float('inf')
min_width = float('inf')
min_img_file = None
# Loop over the images in the directory
for filename in os.listdir(directory):
    # Open the image using PIL
    with Image.open(os.path.join(directory, filename)) as img:
        # Get the image width and height
        width, height = img.size
        # Update the minimum width and height if necessary
        if width < min_width:
            min_width = width
            min_img1_file = os.path.join(directory, filename)
        if height < min_height:
            min_height = height
            min_img_file = os.path.join(directory, filename)
# Open the image with the minimum width and height
min_img = Image.open(min_img_file)
min_img1 = Image.open(min_img1_file)
# Print the minimum width and height
print("Minimum Width:", min_width)
print("Minimum Height:", min_height)
min_img.show() # display the image
min_img1.show()
print(min_img_file)
print(min_img1_file)
```

Figure 19: The method of checking the images size

2. Load the MobileNet_V2

The following step was to load the pre-trained model MobileNet_V2 to train it of the required dataset. Figure 20 shows the step of loading the MobileNet_V2 model.

```
def extract_features(directory):  
    #The directory where the images are stored.  
    # Load the MobileNetV2 model:  
    model = MobileNetV2()  
  
    # Remove the second-to-last layer where the claassification occur:  
    model.layers.pop()  
  
    # Create a new model using the modified MobileNetV2 model  
    #The Model will take the input of the input and the output of the MobileNetV2():  
    model = Model(inputs=model.inputs, outputs=model.layers[-2].output)  
  
    # Print the model summary  
    model.summary()  
    #The above lines will builds/call the pre-trained model.
```

Figure 20: Loading the MobileNet_V2 model

As seen in Figure x, building MobileNet_V2 can be concluded as following, loading the MobileNetV2 model and removing the last layer. The last layer of a neural network is usually the output layer, which is responsible for making the final predictions. In this case, the last layer is removed because the model will not be used for making predictions, but rather, it will be used to extract features from the images. The features will be taken from the second to last layer, which is accessed using the index -2 of the model.layers list, as shown in the following line: `outputs=model.layers[-2].output`.

The new model will take the same inputs as the original MobileNetV2 model, but it will output the features from the second to last layer instead of making predictions. The model summary will give an overview of the new model's architecture and how many parameters each layer has.

2. Feature extraction

After applying the pre-trained model, the following step was to save the extracted features of each image in dynamic file, the reason why to save it externally is to avoid applying the function each single time of testing the model.

The function initializes an empty dictionary called features and loops through all the image files in the provided directory. For each image, it loads the image using the PIL library's `load_img()` function, resizing it to the target size of (224,224) which is required for MobileNetV2 architecture, then it converts the image to an array using the `img_to_array()` function. It then reshapes the image

array to match the required input dimensions for the model, normalizes the image using the `preprocess_input()` function, and then passes it through the model to extract the features from the second-to-last layer. Finally, the function associates the image filename with its extracted feature in the features dictionary, and returns this dictionary after all images have been processed. Figure 21 shows the extracted features before moving it to external file.

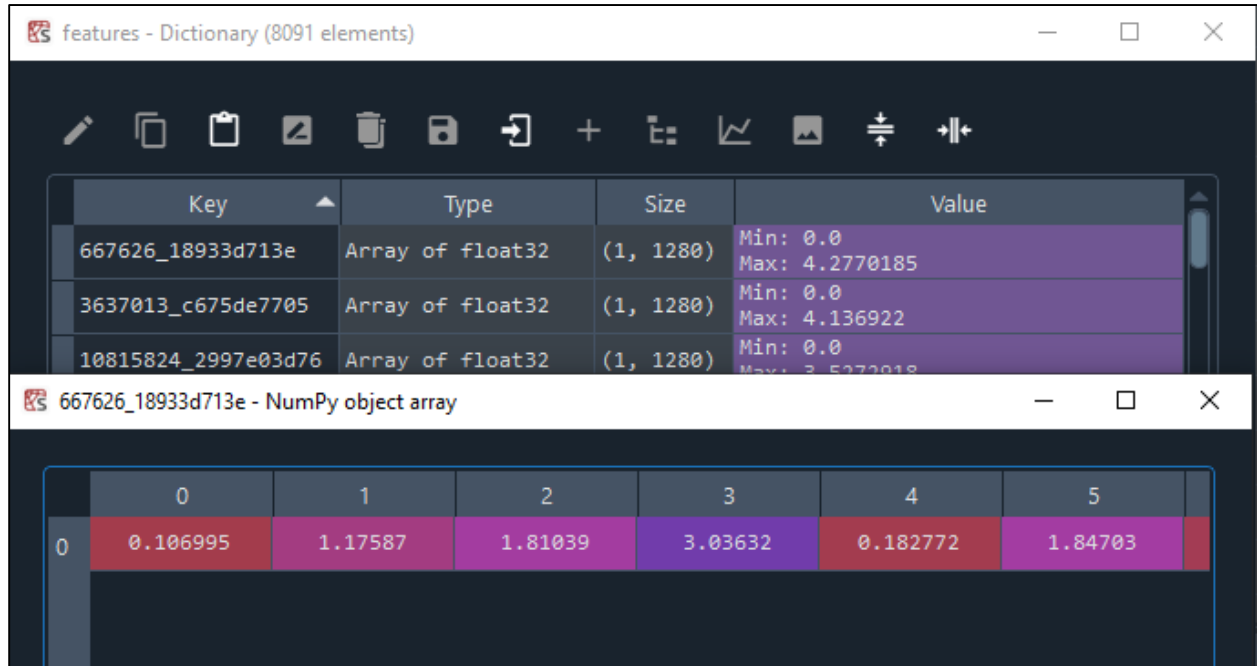


Figure 21: The extracted features before moving it to external file

Figure 22 illustrates how the PKL file will look like on the desktop.


Name	Date modified	Type	Size
 features_MobileNet	1/10/2023 11:50 AM	PKL File	40,933 KB

Figure 22: The extracted features of the images

Ultimately, the CNN model was built using a pre-trained model and the features were extracted and will be passed to the LSTM model. However, it should be mentioned that while the CNN model was being built, the NLP model was also developed. The following section illustrates all the required steps to build Arabic NLP.

3.5 Text pre-processing using NLP techniques

After the images were prepared, the following step is captions pre-processing, the pre-processing of the captions and text is essential step in the project because these captions are the output of the deep learning model.

3.5.1 Load and read the document caption file

The captions file contains the ID of each image with three captions for each image, accordingly the first step was to read and open the captions file. This was done by using the command open, it should be noted that the file contains characters or symbols that was not easy to read using the python, due to this, the file was encoded. Figure 23 show the step of load captions file.

```
def load_document(filename):
    file=open(filename,'r',encoding='utf-8')
    text=file.read()
    file.close()
    return text
```

Figure 23: Load and read the document caption file

3.5.2 Prepare the text

The following step was to split the captions file, this step is important because as previously mentioned, the captions file contains the image ID and the captions , the captions was later on prepared to be passed to the model.

```
def load_captions(caption):
    mapping=dict()
    for line in caption.split('\n'):
        token=line.split('\t') #Split the captions to array of image ID and caoptions
        if len(token)<2: #To skip short captions "في المنطقة. المشجرة"
            continue
        image_id,image_desc=token[0],token[1:]
        image_id=image_id.split('.')[0] #To remove the extension from the image id
        image_desc='.'.join(image_desc) #To convert it to string
        if image_id not in mapping:
            mapping[image_id]=list() #To add the imge id to the list
        mapping[image_id].append(image_desc)
```

Figure 24: Load the text to python environment

As seen in Figure x, the function is loop over the captions the text variable, which is the output from the previous step. Figure 25 illustrates the text variable in which the function will loop through.

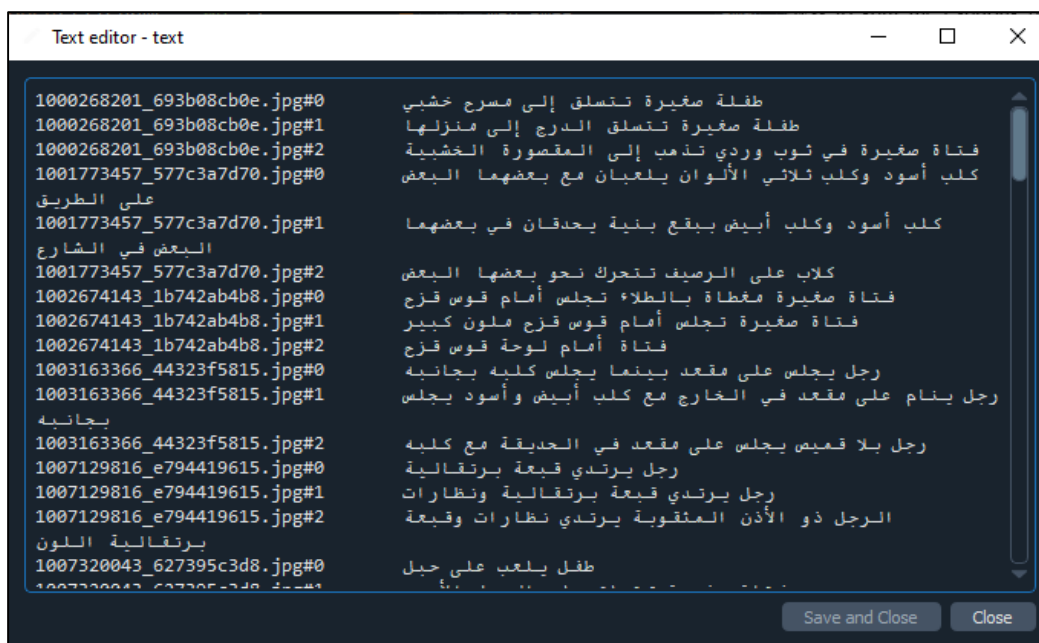


Figure 25: The text variable in which the function will loop through

Through looping process, the function will split each line into two parts, which are the ID and the captions, the captions were initially split from each other using ('\n'), then each line was split using the ('\t'). It should be noted that one of the captions took more than one line in the original file, accordingly, to ensure that it will not be spirited from the min caption, the statement (len(token) < 2) was used to ensure that the code will skip split the line less than two words. Lastly, the captions were saved separately without their ID. Figure 23 shows the return captions file after applying the function.

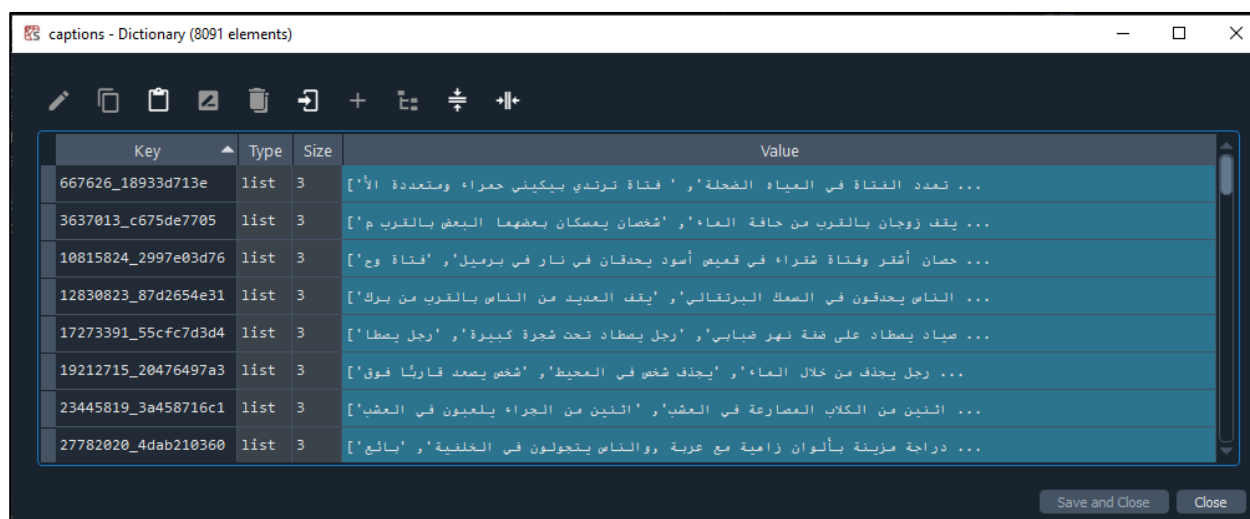


Figure 26: The return captions file after applying the function

3.5.3 Visualize images with their corresponding captions

Before moving to the following step of text cleaning, the images with their corresponding were viewed, Figure 27 present the step of visualize images with their corresponding captions.

```
# In[]:
images_directory='C:\\Users\\hp\\Desktop\\Personal files\\HTU\\python\\project\\Dataset\\Flicker8k_Dataset'
def visualize_images_with_captions(cpts):
    """plot images with their captions"""
    n_pics = len(cpts)
    i = 0
    fig = plt.figure(figsize=(10,n_pics*4))
    #for i,line in enumerate(lines[:n_pics]):
    for img,cpt in captions.items():
        img_file = img + '.jpg'
        img = load_img(images_directory + '\\\\' + img_file)
        cpt_ = '\\n'.join(cpt)
        caption = arabic_resaper.reshape(cpt_)
        caption = bidialg.get_display(caption)
        ax = fig.add_subplot(n_pics,2,2*i+1,xticks=[],yticks=[])
        ax.imshow(img)
        ax = fig.add_subplot(n_pics,2,2*i+2)
        plt.axis('off')
        ax.plot()
        ax.set_xlim(0,1)
        ax.set_ylim(0,1)
        ax.text(0,0.5,caption,fontdict={'color': 'black','weight': 'normal','size': 14})
        i += 1
    if i >= n_pics:
        break
subset = {k:captions[k] for k in list(captions.keys())[:10]}
visualize_images_with_captions(subset)
```

Figure 27: The step of visualize images with their corresponding captions

This code appears to be a function that visualizes a set of images along with their captions. The function takes in one argument, "cpts", which is assumed to be a dictionary containing image file names as keys and their corresponding captions as values. The function loads the images using the Pillow library's "load_img" function and plots them in a grid layout with the captions alongside each image. The script also use python-bidi library to handle the bidirectional text and arabic_resaper to reshape the Arabic text and make it easy to read. The script is showing only 10 images and captions by creating subset of the captions dictionary which contains only 10 items. It uses the Matplotlib library to plot the images and captions.



Figure 28: The results of images visualization with their captions

3.5.4 Text cleaning

Text cleaning is one important step in the project and should be done correctly to ensure that the output captions are correct. Unlike the traditional text pre-processing, in the image captioning, there are few things that should be taken into consideration, such as the verbs and the preposition, this subsection provided an overview of the text cleaning step in the project. Figure 29 shows the steps of text cleaning.

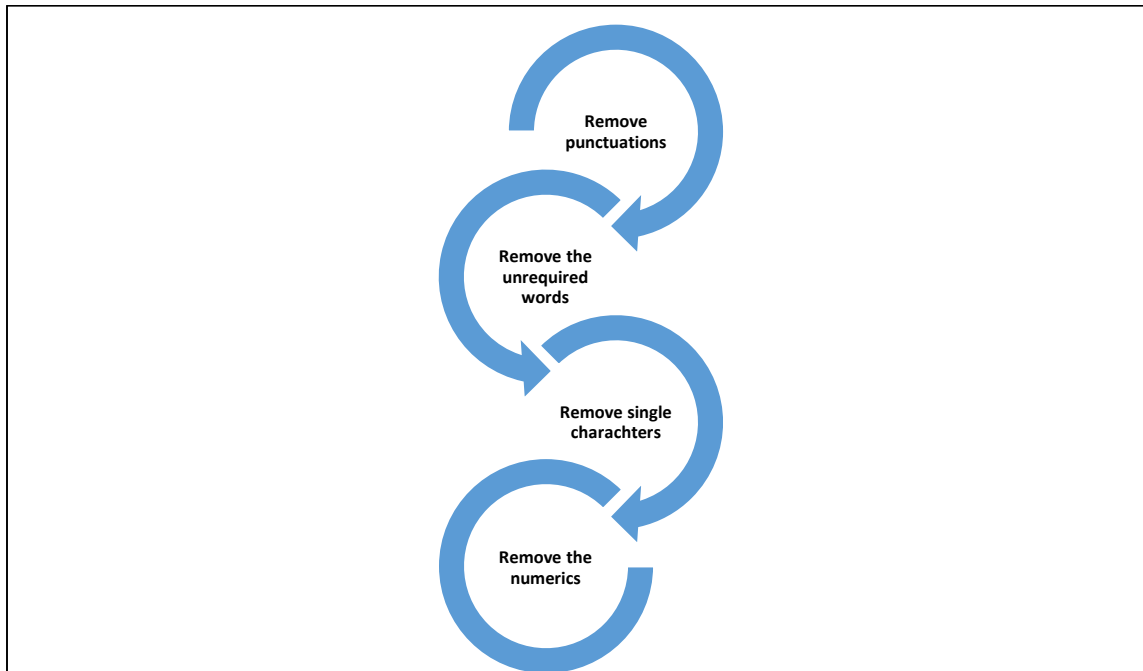


Figure 29: Text cleaning steps

- **Remove the punctuations**

The first step in text cleaning was to remove the punctuation from the text, Figure 30 shows the steps of removing the punctuations from the text.

```

###
nltk.download('punkt') # Download the english punctuation
arabic_punctuations = '""÷×_-"!|+!~{',.?"':/,-][%^&*()_<>:'"
english_punctuations = string.punctuation # Get all the special char
punctuations_list = arabic_punctuations + english_punctuations

def remove_punctuations(data):
    return ''.join([c for c in data if c not in punctuations_list])
#def remove_punctuation(text):
#    #text_no_punctuation = re.sub(r'^\w\s]', '', text)
#    #return text_no_punctuation
  
```

Figure 30: Remove the punctuations

- **Apply the stop words and the stemmer :**

Stop words mean to remove any unrequired words in the sentence, unrequired word means the words that don't add any information to the text. While the stemmer mean to return the word into their roots. Figure 31 presents the text after applying the stop words and stemmer.

Inde ▲	Type	Size	Value
0	str	22	طفل صغر سلق لي سرح خشب
1	str	22	طفل صغر سلق درج لي نزل
2	str	30	فتة صغر ثوب ورد ذهب لي قصر خشب
3	str	35	كلب اسد كلب ثلث الو لعب بعض بعض طرق
4	str	39	كلب اسد كلب ابض بقع بنة حدق بعض بعض شرع
5	str	19	كلب رصف تحر بعض بعض
6	str	27	فتة صغر غطة طلاء جلس قوس قزح
7	str	27	فتة صغر جلس قوس قزح لون كبر

Figure 31: The text after applying the stop words and stemmer

After checking the results of cleaning, it can be concluded that the results are not clear and before applying any kind of text cleaning, the following points should be highlighted in order to know what to keep and to remove from the text:

1. The region features

The region features are extremely important, because it gives an information of the position of the object. Figure x present the meaning of the region feature.

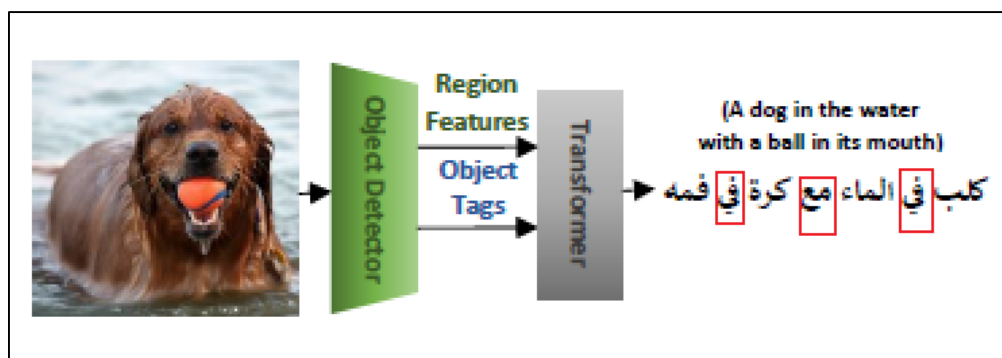


Figure 32: The meaning of the region feature

2. The rich morphology of Arabic language

The Arabic language unlike English language deals with the male and females individually for example in Figure 33 we are going to say ('فتاة تتسلق على الدرج') not ('فتاة يتسلق على الدرج').



Figure 33: The rich morphology of Arabic language

By applying the stop words and the stemmer, these two important characteristics will be removed, accordingly, the appropriate way is to complete the cleaning without applying them.

- **Remove single characters and the definition words**

The following step was to only add the word with length larger than one character and drop the other less than that. Furthermore, the definition Arabic words which called (ال التعريف) was also removed. Figure 34 shows the step of removing the single characters and definition words.

```
def remove_single_character(text):
    text_len_more_than1 = ""
    for word in text.split():
        if len(word) > 1:
            text_len_more_than1 += " " + word
    return(text_len_more_than1)

def remove_defining_words(text):
    text_with_no_defining_chr=re.sub(r'\b_//(\w\w+)', r'\1', text)
    return text_with_no_defining_chr
```

Figure 34: The step of removing the single characters and definition words

- **Remove the numeric**

Then to ensure that the text doesn't include any numeric values, it was cleaned to keep characters only. But instead of removing any number, the numeric values were converted to string values using the pyarabic libraries. Figure 35 show the step of converting the numeric to words.

```
def remove_numeric(text):
    # Use regular expression to extract all Arabic numerals from the text
    numerals = re.findall(r'\d+', text)
    # Convert each numeral to its corresponding word form
    an = pyarabic.number.ArNumbers()
    text_no_numeric = text
    for numeral in numerals:
        word_form = an.int2str((numeral))
        text_no_numeric = text_no_numeric.replace(numeral, word_form)
    return text_no_numeric
```

Figure 35: The step of converting the numeric to words

- **Remove the harakat from the text**

The last step was to remove the harakat (َ ِ ُ) it is called in Arabic fatha, dama, and kasrah, some text may contains these harakat, accordingly they were checked and removed to normalize the text. Figure x shows the harakat and their English meaning and Figure 36 presents the step of removing the harakat from the text.

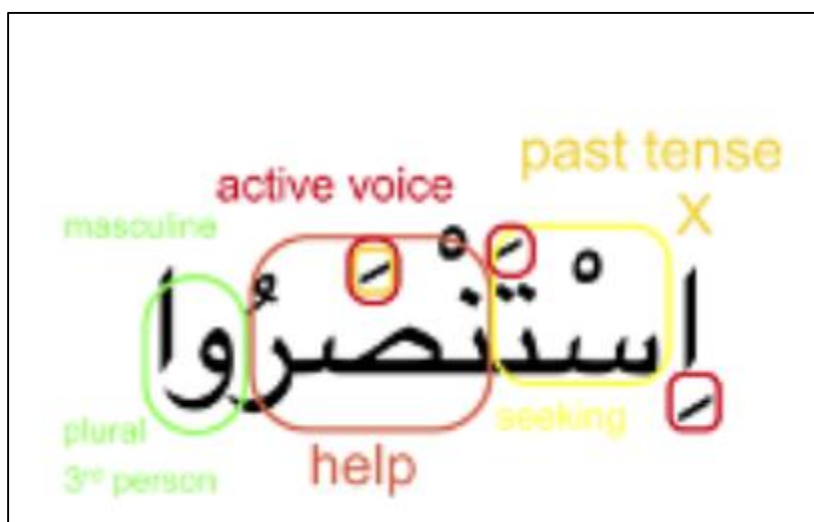


Figure 36: Harakat and their English meaning

```
def remove_harakat(text):
    text_no_harakat=strip_tashkeel(text)
    text_no_diacritics=strip_diacritics(text_no_harakat)
    text_no_tatweel=strip_tatweel(text_no_diacritics)
    text_wihout_english=re.sub(r'[a-zA-Z]+', '',text_no_tatweel)
    return (text_wihout_english)
```

Figure 37: Remove the harakat from the text

3.6 Building the LSTM model

The last and the most important step was building the LSTM model which is responsible of predicting the captions for each image. Building the LSTM model was pass through several stages, these can be listed below:

1. Adding the prefix <start> and suffix <end> to each of our captions

In order to simplify the process of prediction, starting and ending point were added to each caption, in order to enable the model to provide accurate predictions. Figure 38 presents the step of adding the start and end point to the model.

```
def load_clean_descriptions(filename,dataset):
    doc=load_document(filename)
    descriptions=dict()
    for line in doc.split('\n'):
        tokens=line.split()
        image_id,image_desc=tokens[0],tokens[1:]
        #skip images not in set:
        if image_id in dataset:
            #create list:
            if image_id not in descriptions:
                descriptions[image_id]=list()
            desc='startseq ' + ' '.join(image_desc)+' endseq'
            descriptions[image_id].append(desc)
    return descriptions
```

Figure 38: The step of adding the start and end point to the model

2. Convert the cleaned captions into list

Instead of dealing with dictionary, the model will deal only with the captions, in other words, the model won't accept the images id, and accordingly they should be removed. Figure 39 presents the Convert the cleaned captions into list.

```
def to_lines(descriptions):
    all_desc=list()
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc
```

Figure 39: Convert the cleaned captions into list

3. Create a tokenizer

The importance of the tokenizer is to give each single word in the text unique number

```
def create_tokenizer(descriptions):
    lines=to_lines(descriptions)
    tokenizer=Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer
```

Figure 40: Create a tokenizer

4. Test the vocabulary size

The model is initialized by two important elements, these are the vocabulary size and the maximum length of captions, the step of testing the vocabulary size is presented below.

```
tokenizer=create_tokenizer(train_descriptions)
vocab_size=len(tokenizer.word_index)+1
print('vocabulary Size: %d' % vocab_size)
```

Figure 41: Test the vocabulary size

5. Create sequence of images, input sequences and output words for image

The model will pass word by word through the training process, in other words, if the caption was for example 'فتاة برداء أحمر تستلقي على العشب', then the model will start with the prefix word then it will pass the girl, till reach the endseq word. Figure 42 presents the method of create sequence of images, input sequences and output words for image.

```
def create_sequence(tokenizer,max_length,desc_list,photo):
    X1,X2,y=list(),list(),list()
    #Loop over the descriptions:
    for desc in desc_list:
        #Encode the sequence:
        seq=tokenizer.texts_to_sequences([desc])[0]
        #Split the sequence into multiple X,y pairs
        for i in range(1,len(seq)):
            #split into input and output pair:
            in_seq,out_seq=seq[:i],seq[i]
            #pad input sequence:
            in_seq=pad_sequences([in_seq],maxlen=max_length)[0]
            #encode the output sequence:
            out_seq=to_categorical([out_seq],num_classes=vocab_size)[0]
            #Store the variables:
            X1.append(photo)
            X2.append(in_seq)
            y.append(out_seq)
    return array(X1),array(X2),array(y)
```

Figure 42: Create sequence of images, input sequences and output words for image

This function takes in a tokenizer, maximum sequence length, a list of descriptions, and a photo as input. It returns three arrays: X1, X2, and y. The function processes the descriptions by encoding them into sequences using the tokenizer, splitting the sequences into input-output pairs, padding the input sequences to the specified maximum length, and encoding the output sequences using one-hot encoding. The resulting input-output pairs are then stored in the X1, X2, and y arrays respectively, with X1 containing the photo input, X2 containing the padded input sequence, and y containing the one-hot encoded output sequence.

6. Define the model

Defining model step represents the step of building the LSTM model, Figure 43 presents the step of defining the model.

```
def define_model(vocab_size, max_length):
    # Feature Extractor
    inputs1 = Input(shape=(1280,))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)
    # Sequence Model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)

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

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

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

    # summarize model
    print(model.summary())
    plot_model(model, to_file='model.png', show_shapes=True)
    return model
```

Figure 43: Define the LSTM model

This function defines a neural network model for image captioning. It takes in two parameters, the vocabulary size and the maximum length of the input sequences.

The model has two input layers, one for the image feature (inputs1) and one for the input sequence (inputs2).

The image feature extractor takes the image feature (1280-dimensional) as input, applies a dropout layer to prevent over fitting, and a dense layer with 256 neurons and a ReLU activation function. The sequence model takes the input sequence, applies an embedding layer to map the integers to a dense vector representation with 256 dimensions, a dropout layer, and a LSTM layer with 256 units.

The decoder combines the feature extractor and sequence model by concatenating the output of the two layers and applies a dense layer with 256 neurons and a ReLU activation function. The final output layer has as many neurons as the vocabulary size and a softmax activation function.

The model is then compiled with a categorical cross-entropy loss function, Adam optimizer, and accuracy as the evaluation metric. The summary of the model is printed and the model architecture is also plotted in a file named 'model.png'. The function returns the defined model.

3.7 Scoring method

The process of evaluating the image caption generating project is basically based on the Bilingual Evaluation Understudy BLEU score method, which is mainly dependent on n-gram precision. In other words, it compared the number of the correct words that is existed in the predicted caption in comparison with the correct caption [38]. Equation 1 presents the idea of scoring method of the project.

Equation 1: BLUE score equation

$$BLEU \text{ score} = \frac{\# \text{ matches } (\# \text{ words appeared in both generated caption \& ground truth})}{\# \text{ words in generated caption}} \quad (1)$$

It should be mentioned that, the blue score that was used to evaluate the model was obtained from open source. Figure 44 presents the evaluation technique based on blue score.

```
# Evaluate model performance
def evaluate_model(model, descriptions, photos, tokenizer, max_length):
    actual, predicted = list(), list()

    # step over the whole set
    for key, desc_list in descriptions.items():
        # generate description
        yhat = generate_desc(model, tokenizer, photos[key], max_length)

        # store actual and predicted
        references = [d.split() for d in desc_list]
        actual.append(references)
        predicted.append(yhat.split())

    # calculate BLEU score
    print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
    print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
    print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
    print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))
```

Figure 44: Scoring method

Discussion of results

The proposed model architecture shown in Figure 45 for Arabic image captioning consists of the encoder (on the left) and the decoder (right side).

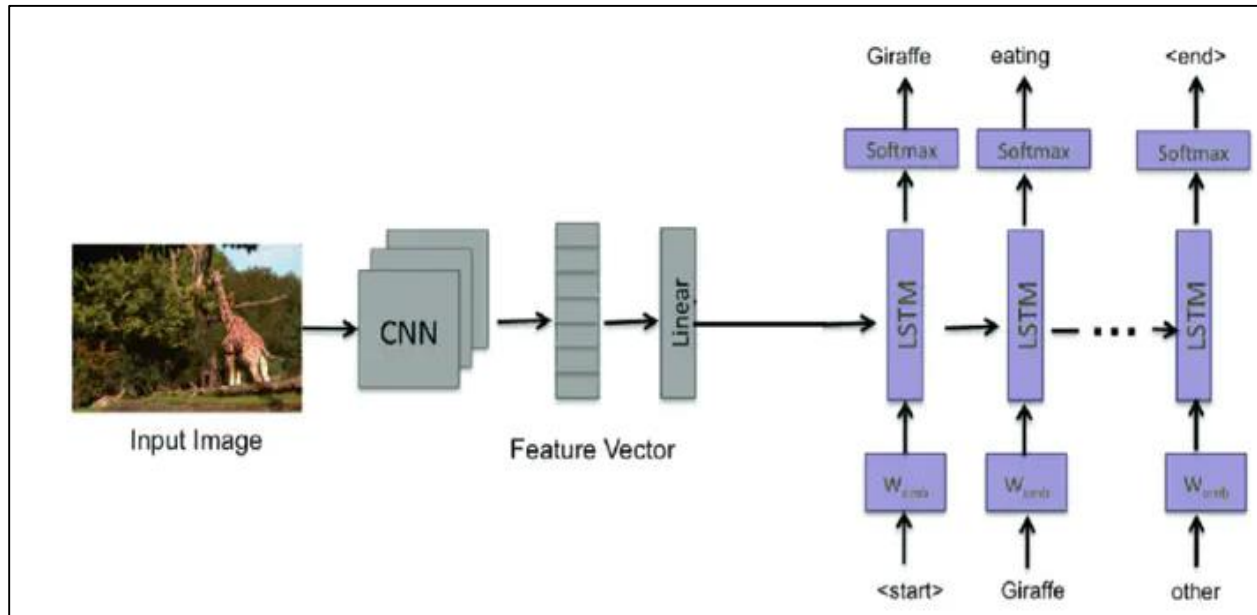


Figure 45: Arabic image captioning consists of the encoder (on the left) and the decoder (right side) [39]

4.1 Encoder model discussion

As mentioned in the methodology chapter, the images size were checked in order to select the target size of the images, however, based on the MobileNet_V2, the images target size is (224,224).

The least image size was:

- Minimum Width: 164
- Minimum Height: 127

The encoder model follows the typical architecture of the pre-trained model based on MobileNet V2 model. The model architecture is mainly constructed from 53 convolution layers, and 1 AvgPool, the convolution layers are constructed from 2 layers these are 1x1 Convolution and 3x3 Depthwise Convolution [40]. As previously mentioned, all these layers are called blocks and they are repeated several times in which they make the CNN deep to elevate the model performance.

Figure 46 shows a sample of the results of the pre-training CNN model.

block_13_project_BN (BatchNormalization)	(None, 7, 7, 160)	640	['block_13_project[0][0]']
block_14_expand (Conv2D)	(None, 7, 7, 960)	153600	['block_13_project_BN[0][0]']
block_14_expand_BN (BatchNormalization)	(None, 7, 7, 960)	3840	['block_14_expand[0][0]']
block_14_expand_relu (ReLU)	(None, 7, 7, 960)	0	['block_14_expand_BN[0][0]']
block_14_depthwise (DepthwiseConv2D)	(None, 7, 7, 960)	8640	['block_14_expand_relu[0][0]']
block_14_depthwise_BN (BatchNormalization)	(None, 7, 7, 960)	3840	['block_14_depthwise[0][0]']
block_14_depthwise_relu (ReLU)	(None, 7, 7, 960)	0	['block_14_depthwise_BN[0][0]']
block_14_project (Conv2D)	(None, 7, 7, 160)	153600	['block_14_depthwise_relu[0][0]']
block_14_project_BN (BatchNormalization)	(None, 7, 7, 160)	640	['block_14_project[0][0]']
block_14_add (Add)	(None, 7, 7, 160)	0	['block_13_project_BN[0][0]', 'block_14_project_BN[0][0]']

Figure 46: A sample of the results of the pre-training CNN model

4.2 The decoder model discussion

The decoder model was initially constructed from the NLP model and the LSTM model, as mentioned earlier, the text and captions were cleaned using different cleaning techniques. Table 5 shows the results of text cleaning.

Table 5: The results of text cleaning

index	file	caption
0	0 1000268201_693b08cb0e.jpg	طفلة صغيرة تتساقط إلى مسرح خشبي
1	1 1000268201_693b08cb0e.jpg	طفلة صغيرة تتساقط درج إلى منزلها
2	2 1000268201_693b08cb0e.jpg	فتاة صغيرة في ثوب وردي تذهب إلى مقصورة خشبية
3	0 1001773457_577c3a7d70.jpg	... كلب أسود و كلب ثنائي ألوان يلعبان مع بعضهما مع
4	1 1001773457_577c3a7d70.jpg	... كلب أسود و كلب أبيض يتبع بنية يحلقان في بعضهما
...
24268	1 997338199_7343367d7f.jpg	امرأة تقف بالقرب من جدار مزخرف تكتب
24269	2 997338199_7343367d7f.jpg	جنران معطاة بالذهب والأنماط
24270	0 997722733_0cb5439472.jpg	رجل يرتدي قميصا ورديا يتساقط وجها صغيرا
24271	1 997722733_0cb5439472.jpg	رجل يتساقط صغير في هواء
24272	2 997722733_0cb5439472.jpg	متساقط صغيرة في قميص أحمر

As seen, the text was cleaned from any unrequired factors and elements that might have an impact on the prediction process. Ultimately, the output of this stage was 3 cleaned captions for each image. Through the process, it was important to check the most frequent words in the captions and the number of the unique vocabulary in the text. Frequent vocabulary size (number of unique words) was 5526, Figure 47 presents the most frequent 50 words in the captions.

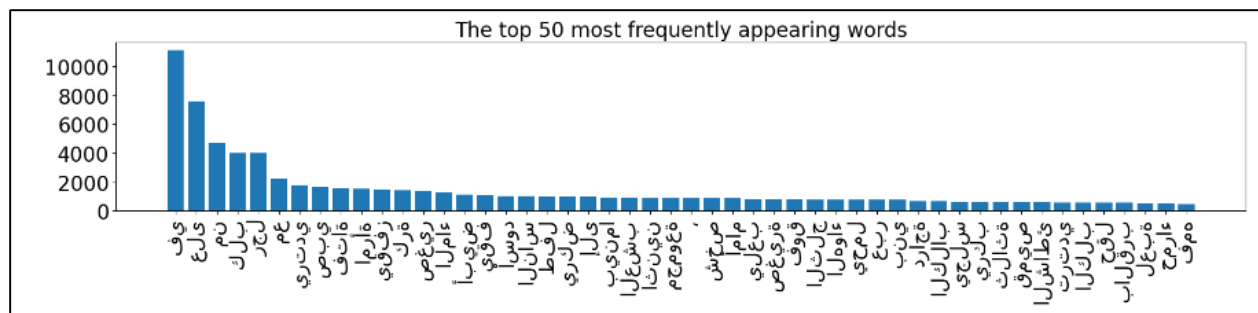


Figure 47: The most frequent 50 words in the captions

As illustrated, the words that give region indication were kept in the text. The Clean Vocabulary size (number of unique words): 11387.

As mentioned earlier, before importing the captions to the RNN-LSTM model, each caption were bounded by prefix and suffix. Table 6 illustrates the images captions with their boundary.

Table 6: The images captions with their boundary

index	file	caption
0	0 1000268201_693b08cb0e.jpg	startseq طفلة صغيرة تتسلق إلى مسرح خشبي endseq
1	1 1000268201_693b08cb0e.jpg	startseq طفلة صغيرة تتسلق درج إلى منزلها endseq
2	2 1000268201_693b08cb0e.jpg	startseq فتاة صغيرة في ثوب وردي تذهب إلى مقصو
3	0 1001773457_577c3a7d70.jpg	startseq ... كلب أسود و كلب ثنائي ألوان يلعبان مع
4	1 1001773457_577c3a7d70.jpg	startseq ... كلب أسود و كلب أبيض يتبع بنية يحذقان
...
24268	1 997338199_7343367d7f.jpg	startseq ... امرأة تقف بالقرب من جدار مزخرف تكتب
24269	2 997338199_7343367d7f.jpg	startseq جدران معطاة بالالاب والألماط endseq
24270	0 997722733_0cb5439472.jpg	startseq ... رجل يرتدى قميصا وريدا يتسلق وجهها صعر
24271	1 997722733_0cb5439472.jpg	startseq رجل يتسلق صخور في هواء endseq
24272	2 997722733_0cb5439472.jpg	startseq متسلق صخرة في قميص أحمر endseq

After applying the model on the training set, and after 15 epochs the accuracy didn't exceed the 36.5% which is similar to what was found in the previous literature. Lastly, the model was tested on test data set and provide relatively good captions, although it needs further improvements. Figure 48,49 presents sample of the provided results.



Figure 48: Sample results one



Figure 49: Sample results two

Conclusion and future work

The percentage of children with learning difficulties are increasing rapidly in the Arabic world, based on the importance and the contribution of new technologies in solving real world problem, the project idea seed was to asset these people to improve the capabilities of making good sentences. The idea was to build Arabic image captioning that is able to produce captions from images based on the features in the images. The data set that was used is constructed from 8091 images with three Arabic captions for each one. The model itself is constructed from the encoder which represents the pre-trained CNN model, and the decoder part which is constructed from the NLP and the LSTM. The overall accuracy was 36.5%, and the model was able to do good prediction. Although this, the model can be improved in order to enable it to extract all important objects in the images.

Bibliography

- [1] M. Saleh and J. M. Alja'am, "Towards Adaptive Multimedia System for Assisting Children with Arabic Learning Difficulties," Amman, Jordan, 2019.
- [2] unicef, "unicef," 2019. [Online]. Available: <https://www.unicef.org/jordan/press-releases/unicef-jordan-raises-awareness-rights-children-disabilities-access-education-zarqa>. [Accessed 27 12 2022].
- [3] K. Iwamura, J. Younes, L. Kasahara, A. Moro, A. Yamashita and H. Asama, "Image Captioning Using Motion-CNN with Object Detection," *Sensors* , vol. 21, no. 4, 2021.
- [4] B. Alazzam, "Arabic image captioning using ResNet50," *Researchgate*, 2022.
- [5] B. Zhang, L. Zhou, S. Song, L. Chen, Z. Jiang and J. Zhang, " Image Captioning in Chinese and Its Application for Children with Autism Spectrum Disorder," 2020.
- [6] M. Stefanini, M. Cornia, L. Baraldi and S. Cascianelli, *From Show to Tell: A Survey on Deep Learning-based Image Captioning*, arXiv preprint , 2021.
- [7] M. E. Za'ter and B. Talafha, "BENCH-MARKING AND IMPROVING ARABIC AUTOMATIC IMAGE CAPTIONING THROUGH THE USE OF MULTI-TASK LEARNING PARADIGM," *arXiv*, 2022.
- [8] M. Pedersoli, T. Lucas and C. Schmid, "Areas of Attention for Image Captioning," *arXiv*, 2021.
- [9] lilianweng, "lilianweng," 2022. [Online]. Available: <https://lilianweng.github.io/posts/2018-06-24-attention/>. [Accessed 1 1 2023].
- [10] Hossain, M. Zakir, F. Sohel, M. F. Shiratuddin and H. Laga, "A Comprehensive Survey of Deep Learning for Image Captioning," *ACM Computing Surveys (CsUR)* , vol. 51, no. 6, pp. 1-36, 2019.
- [11] B. M. Alazzam, "Arabic image captioning using ResNet50," 2022.
- [12] J. ZAKRAOUI, S. E. and J. M. ALJA'AM, "Improving Arabic Text to Image Mapping Using a Robust Machine Learning Technique," *SPECIAL SECTION ON ADVANCED SOFTWARE AND DATA ENGINEERING FOR SECURE SOCIETIES*, 2019.

- [13 J. Monteiro, A. Kitamoto and B. Martins, “Situational Awareness from Social Media
] Photographs Using Automated Image Captioning,” 2017.
- [14 H. Hejazi and K. Shaalan, “Deep Learning for Arabic Image Captioning: A Comparative
] Study of Main Factors and Preprocessing Recommendations,” (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 11, p. 45, 2021.
- [15 worlddata, “worlddata,” 2022. [Online]. Available:
] <https://www.worlddata.info/languages/arabic.php>. [Accessed 1 1 2023].
- [16 Obeida ElJundi, “Resources and End-to-End Neural Network Models for Arabic Image
] Captioning,” 1 1 2020. [Online]. Available:
https://drive.google.com/file/d/1vt1LcE_0xEFHpQ95bLWLNWqrwZ9AbTS8/view.
[Accessed 1 1 2023].
- [17 H. A. Al-muzaini, T. N. Al-yahya and H. Benhidour, “Automatic□Arabic Image Captioning
] using RNN-LSTM-Based Language Model and CNN,” (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 6, 2018.
- [18 V. Jindal, “Generating Image Captions in Arabic using Root-Word Based Recurrent Neural
] Networks and Deep Neural Networks,” New Orleans, Louisiana, 2018.
- [19 H. A. Al-muzaini, T. N. Al-yahya and H. Benhidour, “Automatic Arabic Image Captioning
] using RNN-LSTM-Based Language Model and CNN,” (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 6, pp. 67-74, 2018.
- [20 I. Afyouni, I. Azhara and A. Elnagar, “AraCap: A hybrid deep learning architecture for
] Arabic Image captioning,” 2021.
- [21 towardsdatascience, “towardsdatascience,” 2022. [Online]. Available:
] <https://towardsdatascience.com/a-guide-to-image-captioning-e9fd5517f350>. [Accessed 1 1 2023].
- [22 pexels, “pexels,” 2023. [Online]. Available:
] <https://www.pexels.com/search/arabic%20script/>. [Accessed 1 1 2023].
- [23 shutterstock, “shutterstock,” 2022. [Online]. Available:
] <https://www.shutterstock.com/create/editor>. [Accessed 1 1 2023].

- [24 data-flair.trainin, “data-flair.trainin,” 2022. [Online]. Available: <https://data-flair.training/blogs/python-based-project-image-caption-generator-cnn/>. [Accessed 20 12 2022].
- [25 I. Afyouni, I. Azhara and A. Elnagar, “AraCap: A hybrid deep learning architecture for Arabic Image Captioning,” *Procedia Computer Science*, vol. 189, p. 382–389, 2021.
- [26 J. Emami, P. Nugues, A. Elnagar and I. Afyouni, “In Proceedings of the 15th International Conference on Natural Language Generation,” *Arabic Image Captioning using Pre-training of Deep Bidirectional Transformers*, pp. 40-51, 2022.
- [27 medium, “medium,” 2022. [Online]. Available: <https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926>. [Accessed 19 12 2022].
- [28 upgrad, “upgrad,” 2022. [Online]. Available: <https://www.upgrad.com/blog/basic-cnn-architecture/>. [Accessed 4 1 2023].
- [29 keras, “keras,” 2023. [Online]. Available: <https://keras.io/api/applications/>. [Accessed 6 1 2023].
- [30 S. WANG, C. ZHOU, D. ZHANG, L. CHEN and H. SUN, “A Deep Learning Framework Design for Automatic Blastocyst Evaluation With Multifocal Images,” *EEE Access*, p. 99, 2021.
- [31 geeksforgeeks, “geeksforgeeks,” 2023. [Online]. Available: <https://www.geeksforgeeks.org/python-os-listdir-method/>. [Accessed 9 1 2023].
- [32 python, “python,” 2023. [Online]. Available: <https://docs.python.org/3/library/pickle.html>. [Accessed 9 1 2023].
- [33 geeksforgeeks, “geeksforgeeks,” 2023. [Online]. Available: <https://www.geeksforgeeks.org/load-images-in-tensorflow-python/>. [Accessed 10 1 2023].
- [34 geeksforgeeks, “geeksforgeeks,” 2023. [Online]. Available: <https://www.geeksforgeeks.org/image-processing-with-keras-in-python/>. [Accessed 10 1 2023].

- [35 tensorflow, “tensorflow,” 2023. [Online]. Available:
] https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/MobileNetV2. [Accessed 10 1 2023].
- [36 tensorflow, “tensorflow,” 2023. [Online]. Available:
] https://www.tensorflow.org/api_docs/python/tf/keras/Model. [Accessed 10 1 2023].
- [37 M. Akay, Y. Du, C. L. Ser Shen, M. Wu, T. Y. Chen, S. Assassi, C. Mohan and Y. M. Akay,
] “Deep Learning Classification of Systemic Sclerosis Skin Using the MobileNetV2 Model,”
IEEE Open Journal of Engineering in Medicine and Biology, 2021.
- [38 R. Shinde, “medium,” 2019. [Online]. Available:
] <https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926>. [Accessed 14 1 2023].
- [39 margotwagner, “margotwagner,” 2023. [Online]. Available:
] <https://www.margotwagner.com/project/image-captioning-using-an-lstm-network/>.
[Accessed 17 1 2023].
- [40 iq.opengenus, “iq.opengenus,” 2023. [Online]. Available:
] <https://iq.opengenus.org/mobilenetv2-architecture/#:~:text=MobileNet%20V2%20model%20has%2053,Bottleneck%20Residual%20Block>. [Accessed 15 1 2023].
- [41 Z. YING, *SIGNIFICANT FACTORS AFFECTING CONSTRUCTION PRODUCTIVITY*,
] SINGAPORE: NATIONAL UNIVERSITY OF SINGAPORE, 2004.