Project Report

On

Image Captioning: Transforming Objects into Text.

Submitted
In partial fulfilment

For the award of the Degree of

PG-Diploma in Artificial Intelligence

(C-DAC, ACTS (Patna))

Guided By:

Swami Aditya Nath

Submitted By:

Patil Sumit Suresh (PRN No.:220980728007)
Bhise Aditya Krishna (PRN No.: 220980728004)
Patil Rahul Chandrakant (PRN No.: 220980728003)
Kumari Megha (PRN No.: 220980728002)

Centre for Development of Advanced Computing (C-DAC),

ACTS (Patna-800001)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Mr. Swami Aditya Nath** C-DAC ACTS, Patna for her constant guidance and helpful suggestion for preparing this project **Image Captioning: Transforming Objects into Text.** We express our deep gratitude towards him for his inspiration, personal involvement, and constructive criticism that he provided us along with technical guidance during the course of this project.

We take this opportunity to thank the Head of the department **Mr. Saket Jha** for providing us with such a great infrastructure and environment for our overall development.

We express sincere thanks to **Shri Aditya Kumar Sinha**, Director of C-DAC Patna, for their kind cooperation and support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude toward **Mr. Ranjan Sinha** (Course Coordinator, PG-DAI) for their valuable guidance and constant support throughout this work and helps to pursue additional studies.

Also our warm thanks to **C-DAC ACTS, Patna** which provides us with this opportunity to carry out this prestigious Project and to enhance our learning in various technical fields.

Patil Sumit Suresh (PRN No.:220980728007)
Bhise Aditya Krishna (PRN No.: 220980728004)
Patil Rahul Chandrakant (PRN No.: 220980728003)
Kumari Megha (PRN No.: 220980728002)

ABSTRACT

Scene interpretation has long been a key component of computer vision, and image captioning is one of the main topics of AI research since it seeks to imitate how humans can condense a huge quantity of visual data into a small number of sentences. The purpose of image caption generation is to produce a statement that describes an image. The challenge calls for the employment of methods from computer vision and natural language processing in order to generate a brief but comprehensive caption of the image. Significant study in the field has been made possible by recent advancements in deep learning and the accessibility of image caption datasets from sources like Flickr_8k. A multilayer Convolutional Neural Network (CNN), Transformer, and other approaches are proposed in this research.

Table of Contents

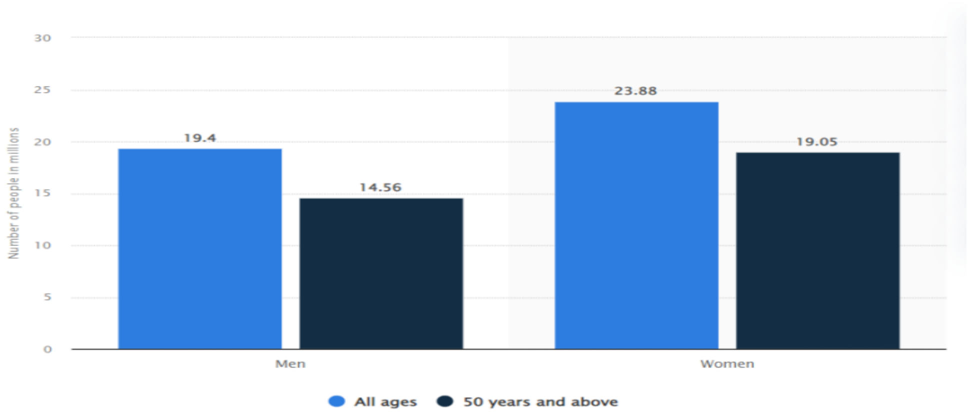
List of Acronyms

Sr. No.	Acronyms	Abbreviations
1	ML	Machine Learning
2	DL	Deep Learning
3	DNN	Deep Neural Network
4	RNN	Recurrent Neural Network
5	LSTM	Long-Short Term Memory
6	NLP	Natural Language Processing
7	AI	Artificial Intelligence

Chapter 1

Introduction

1.1 Relevant Contemporary Issues Image captioning is a multimodal task which can be improved by the aid of Deep Learning. However, even after so much progress in the field, captions generated by humans are more effective which makes image caption generator an interesting topic to work on with Deep learning. Fetching the story of an image and automatically generating captions is a challenging task. The whole objective of generating captions solely lay in the derivation of the relationship between the captured image and the object, generated natural language and judging the quality of the generated captions. To determine the context of image requires the detection, spotting a recognition of the attributes, characteristics and the flow of relationships between these entities in an image. Applications involving computer vision or image/video processing can benefit greatly from deep learning. It is generally used to categorize photos, cluster them based on similarities, and recognize objects in scenarios. ViSENZE, for instance, has created commercial products that use deep learning networks to enhance image recognition and tagging. This enables buyers to search for a company's products or related things using images rather than keywords. Object identification is a more difficult form of this task that requires precisely detecting one or more items within the scene of the shot and boxing them in. This is accomplished using algorithms that can recognize faces, people, street signs, flowers, and many other visual data elements.



The whole model of generating accurate captions has a potential to have large impact. According to World Health Organization, globally, at least 2.2 billion people have a near or distance vision impairment as shown in Figure 1.1. This survey implies that these people cannot view any image on the web, that is where an Image caption generator comes into play. It can help visually impaired people understand the context of an image by deciphering the objects of the image. Image caption generator can also play a major role in the growth of social media where captions play a major role. Till now the only modification with respect to generating captions is given by Instagram which generates a script with respect to the audio fetched, however there is no progress of generating captions by just looking at the image. With internet and technologies continuously growing and people getting hooked to their mobile phones and also social media being a competitive market, inclusion of such a feature that can provide that human touch to an image without the help of human knowledge or emotions to express the essence of the image and also can make the inclusion of visually impaired people feel more included in the society.

1.2 Identification of Problem

Though numerous researches and deep learning models have been already put into play to generate captions, the major problem that lies in hand still is the fact that humans describe image using natural languages which is compact, crisp and easy to understand. Machines on the other hand have not been found efficient to beat the effectiveness the human generated descriptions of a certain image. The task to identify a well-structured object or image is quite an easy task and has been in study by the computer vision community for long but to derive a relationship between objects of an image is quite a task. Indeed, a description must capture the essence and put up the description of the objects of the image but it must also express how the objects correlate with each other as well as their attributes and activities they are performing in the image. The above knowledge has to be expressed in a natural language like English to actually communicate the context of the image effectively. In deep machine learning based techniques, features are learned by the machine on its own from training data and these techniques are able to handle a large and diverse set of images and videos. Hence, they are preferred over traditional methods. In the last few years, numerous articles have been published on image captioning with deep learning. To provide an abridged version of the literature, this paper presents a survey majorly focusing on the deep learning-based papers on image captioning.

1.3Identification of Tasks

A lot of tasks need to be accomplished and relevant skills required to successfully build this project:

1. System design or architecture
2. Dataset fetching from an online website
3. Data Pre-processing or cleaning
4. Building a model – implement using a specific programming language.
5. Testing

6. Report Generation

7. Deployment of the model through an interface

Skills required to complete this project are stated:

1. System design – knowledge of UML diagram, system design concepts
2. Data collection – through online websites like kaggle
3. Data Pre-processing or cleaning – knowledge of python, or any contemporary modern day programming language
4. Building a model – knowledge about a specific programming language preferably Python, expertise in various machine learning techniques.
5. Testing – Knowledge about Unit testing, Black box testing or any special test package in Python.
6. Report Generation – Good command over English language.

1.4 Organization of the Report

The following chapters will give an elaborate description of the project.

- Chapter 2, of this report describes the various literature surveyed, hence give an idea of the recent trends and development in the field. This chapter also propose solutions to solve the problems identified in chapter 1, goals and objectives this project wish to achieve.
- Chapter 3, describes the design constraints we would have. In the same chapter data flow diagram, flowchart will give a visual description of the project details.
- Chapter 4, gives the results obtained, testing methods adopted.
- Chapter 5, gives conclusion and future work that can be pursued to improve the image caption generator.

CHAPTER 2
LITERATURE SURVEY

2.1 Timeline of the Reported Problem

The two main categories of image captioning algorithms are those based on template methods and those based on encoder decoder structures. The two approaches are mostly examined in the following. Fig. depicts the progress of picture captioning. 1. Using templates is mostly how the initial picture captioning method is implemented. to extract a number of important feature data, such as key objects and special attributes, and then transforms the data into description text using a lexical model or other targeted templates. The object in the image, the behavior of the object, and the scene in which it is present are the three basic pieces of information that Farhadi et al. mainly includes three information: the object in the image, the action of the object and the scene in which the object is located. The noise item is removed by some common smoothing methods. Some typical smoothing techniques eliminate the noise component. Finally, Li et al. investigate the relationship between the extracted data and the final picture description result. Early methods for creating image descriptions combined image data utilizing static object class libraries and statistical language models. Aker and Gaizauskas suggest a method for automatically geotagging photographs and utilize a dependency model to summarize several web articles that contain information about image locations. Li and co. Present an n-gram approach based on network scale that gathers potential phrases and combines them to create sentences that describe images starting from scratch. To estimate motion in an image, Yang et al. suggest a language model trained using the English Gigaword corpus. And the probability of collocated nouns, scenes, and prepositions and the likelihood of collocated nouns, situations, and prepositions, and use these projections as hidden Markov model parameters. The most likely nouns, verbs, situations, and prepositions that make up the sentence are used to generate the image description. According to Kulkarni et al., a detector should be used to identify objects in an image, each candidate region should then be classified and subjected to prepositional

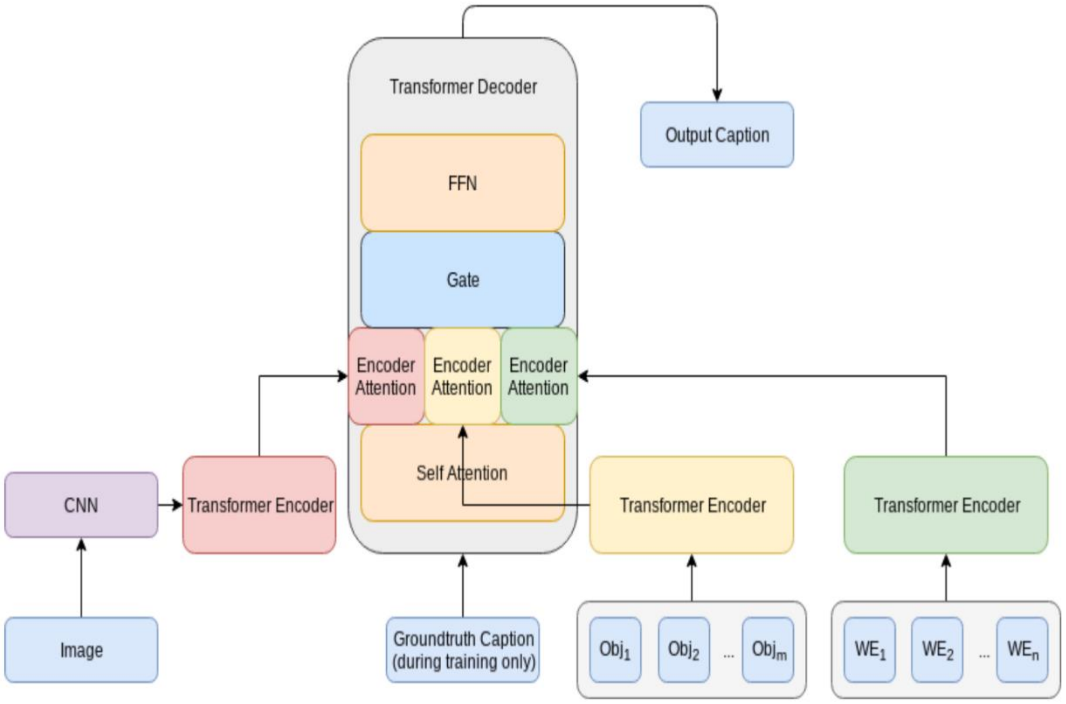
relationship processing, and finally a conditional random field (CRF) prediction image tag should be used to produce a natural language description. On photos, object detection is also done. In order to infer objects, properties, and relationships in an image and transform them into a sequence of semantic trees, Lin et al. used a 3D visual analysis system. After learning the grammar, the trees were used to generate text descriptions.

2.1.1 Handcraft Features with Statistical Language Model

The visual detector and language model are directly learned from the image description dataset using this technique, which is a Midge system based on maximum likelihood estimation, as shown in Figure 1. In order to identify the object in the image and create a caption, Fang et al. first analyze the image. By applying a convolutional neural network (CNN) on the image area and merging the data with MIL , words are recognized. To reduce a priori assumptions about the sentence structure, the sentence structure is then taught straight from the caption. Last but not least, it optimizes the image caption generation problem by looking for the most likely statement

2.1.2 Deep Learning Features with Neural Network

In the field of deep learning, the recurrent neural network (RNN) has received a lot of attention. As can be seen in Figure 2.1, it was initially widely used in the field of natural language processing and produced positive language modelling outcomes . Speech-related RNN functions include text-to-speech conversion , machine translation , question-and-answer sessions , and more. Of course, at the level of characters and words, they are also employed as effective language models. Right now, word-level models appear to be superior to character-level models



but this is undoubtedly a passing improvement. In computer vision, RNN is also quickly rising in prominence. Sequence modelling, frame-level video classification, and current visual questionanswer tasks. 2.1.3 Datasets and Evaluation The foundation of artificial intelligence is data. People are becoming more aware of the fact that a lot of hard-to-find legislation can be found from a lot of data. There are currently several diverse datasets in the picture description generation task, including Flickr8k, and STAIR Captions, which are progressively becoming a point of controversy. Each image in the dataset has five reference descriptions, which add up how many images are in each dataset. The dataset employs alternative syntax to describe the same image in order to have numerous independent descriptions of each image. As seen in the example in Figure 10, various descriptions of the same image concentrate on various features of the scene.

The entire image caption generation task has been covered in this overview, along with the model framework that has been recently suggested to address the description task. We have also concentrated on the algorithmic core of various attention mechanisms and provided a brief overview of how the attention mechanism is used. We provide an overview of the sizable datasets and standard practice rating criteria.

2.2 Proposed Solution

There are many different image captioning techniques, some of which are seldom ever utilized nowadays, but before moving on, it is important to have a general understanding of those technologies. The three primary types of picture captioning techniques now in use are template-based caption generation, retrieval-based captioning, and novel caption production. Innovative ways for creating captions for images typically rely on deep machine learning and visual space. Multimodal space can also be used to create captions. Supervised learning, Reinforcement learning, and Unsupervised learning are three learning strategies that can be used to classify deep learning-based picture captioning solutions. We classify other deep learning into two categories: reinforcement learning and

unsupervised learning. Typically, captions are created for the entire scene depicted in the image However, distinct areas of an image can also create captions (Dense captioning). Either a compositional design or a straightforward encoder-decoder architecture can be used for image captioning. There are techniques that make advantage of the attention mechanism, semantic notion, and various visual description styles. Some techniques can even produce descriptions for invisible objects. However, there are also methods that utilize other language models, such as CNN and RNN, thus we include a language modelbased category as "LSTM vs. Others." The majority of the picture captioning algorithms employ LSTM as their language model.

2.2.1 Template-Based Approaches

Fixed templates with numerous vacant slots are used in template-based systems to produce captions. These methods start by identifying various objects, attributes, and actions before filling up the template's empty spots. For instance, Farhadi et al. fill the template spaces for creating image captions with a triplet of scene elements. For this objective, Li et al. extract the sentences relevant to detected objects, properties, and relationships. Kulkarni et al. use a Conditional Random Field (CRF) to infer the objects, characteristics, and prepositions before filling in the blanks. Grammarly perfect captions can be produced using template-based techniques. Templates, however, are predetermined and unable to produce captions of variable length. Additionally, later on, parsing-based language models for image captioning were established which are more powerful than fixed template-based methods. Therefore, in this paper, we do not focus on these template based methods.

2.2.2 Retrieval-Based Approaches

Both visual and multimodal spaces can yield captions. Captions are retrieved from a group of already-existing captions in retrieval-based systems. The initial step in retrieval-based approaches is to identify the captions and visually comparable images from the training data set. Candidate captions are what they are. These captions are pulled from a pool to create the captions for the search image. These techniques result in comprehensive and punctuation-perfect captions. They are unable to produce captions that are both image-specific and semantically accurate.

2.2.3 Novel Caption Generation

Novel picture captions are ones that the model creates on its own using a language model and a mix of image attributes rather than by comparing them to preexisting captions. The topic of creating original image captions is far more intriguing and practical because it addresses both drawbacks of using existing captions. Visual space and multimodal space can both be used to develop original captions. An strategy used generally in this category is to first analyze the image's visual information before creating image captions from it using a language model, as shown in Figure 2.2. These techniques can produce new captions for each image that are more semantically precise than earlier techniques. Deep machine learning is the most common technique for creating new captions. Therefore, deep learning based novel image caption generating methods are our main focus in this literature.

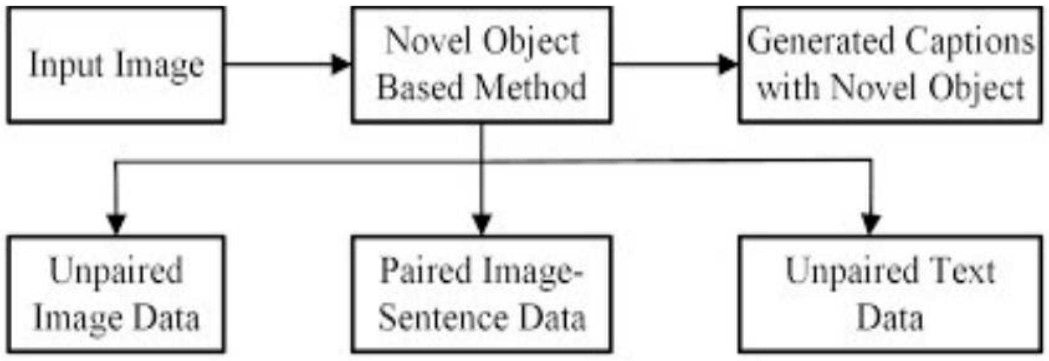

Input Image
Novel Object Based Method
Generated Captions with Novel Object
Unpaired Image Data
Paired Image-Sentence Data
Unpaired Text Data

Figure 2.2 Novel Caption Generation

2.3 Bibliometric Analysis In Figure 2.3, we depict a broad taxonomy for deep learning-based picture captioning techniques. We categorize them into visual space versus multimodal space, dense captioning versus scene-wide captioning, supervised learning versus other deep learning, encoder-decoder architecture versus compositional architecture, and one "Others" group that includes attention-based, semantic concept-based, stylized captions, and novel objectbased captioning to discuss their similarities and differences. Additionally, we develop the category LSTM vs. Others. The table provides a brief summary of the deep learning-based picture captioning techniques. It lists the names of the picture captioning techniques, the class of deep neural networks that were used to encrypt the image data, and the language models that were employed to describe the images. In the final column, we give a category label to each captioning technique based on the taxonomy in Figure 2.3.

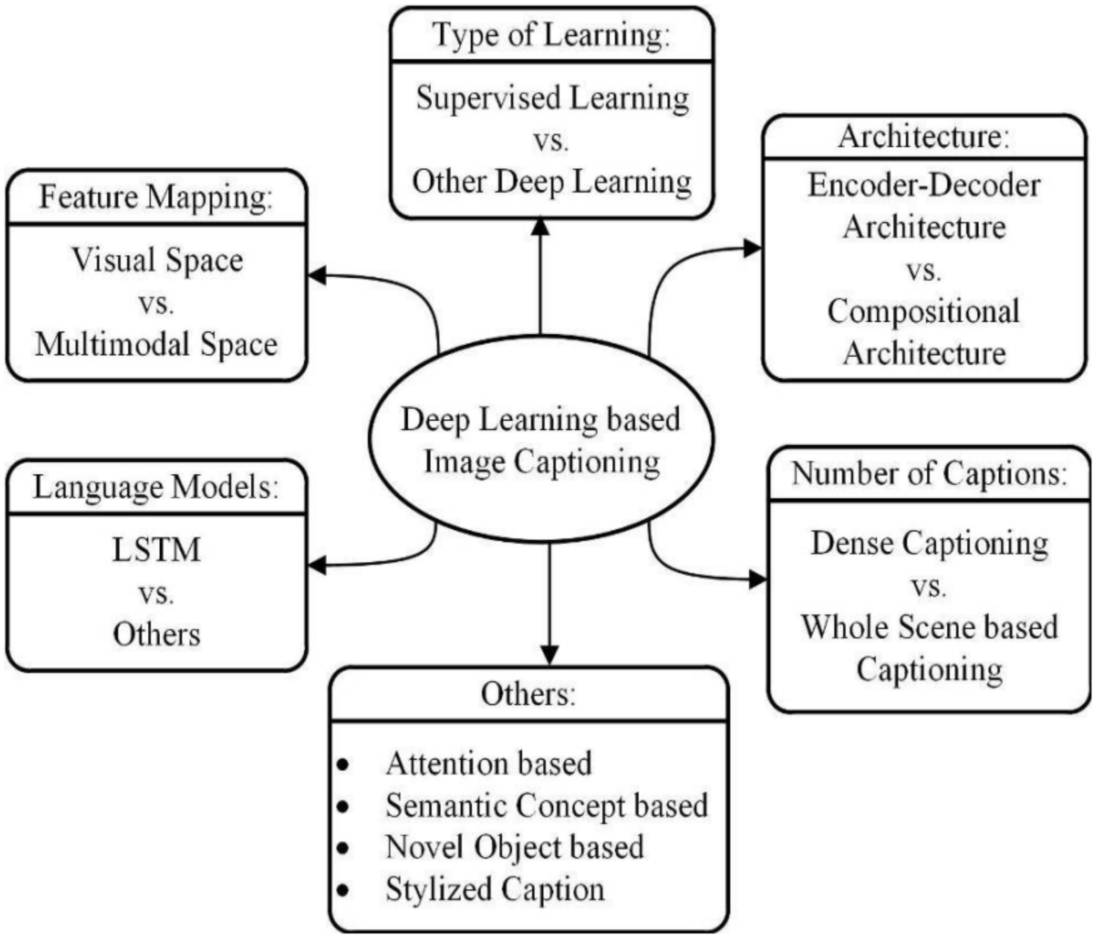

Type of Learning:
Supervised Learning vs. Other Deep Learning
Architecture:
Encoder-Decoder Architecture vs. Compositional Architecture
Feature Mapping:
Visual Space vs. Multimodal Space
Deep Learning based Image Captioning
Language Models:
LSTM vs. Others
Number of Captions:
Dense Captioning vs. Whole Scene based Captioning
Others:
Attention based
Semantic Concept based
Novel Object based
Stylized Caption

Figure 2.3 An overall taxonomy of deep learning-based image captioning

Visual Space Vs Multimodal Space

Deep learning-based algorithms for captioning images can produce captions from both multimodal and visual spaces. Naturally, databases for image captioning contain text versions of the associated captions. The image features and the related captions are independently provided to the language decoder in the visual space-based approaches. In contrast, a shared multimodal space is learned from the images and the related captions in a multimodal space example. The language decoder is then given access to this multimodal representation. Visual Space Bulk of the image captioning methods use visual space for generating captions. In the visual space-based methods, the image features and the corresponding captions are independently passed to the language decoder. Multimodal Space Language encoder, vision part, multimodal space part, and language decoder are all components of a typical multimodal space-based method's design. In order to extract the image features, the vision component uses a deep convolutional neural network as a feature extractor. The language encoder portion learns a dense feature embedding for each word after extracting the word features. The semantic temporal context is subsequently transmitted to the recurrent levels. The multimodal space component places the word features and image features in the same spatial location. 2.3.2 Limitations A helpful framework for learning to map from photos to human-level image captions is provided by the neural image caption generator. The algorithm learns to extract pertinent semantic information from visual attributes by training on a large number of image-caption pairs. However, when used with a static image, our caption generator will concentrate on aspects of the

image that are helpful for image classification rather than necessarily aspects that are helpful for caption production. We may train the image embedding model (the CNN used to encode features) as a component of the caption generation model, enabling us to precisely adjust the image encoder to more effectively perform the duty of creating captions.

2.4 Review Summary

The approaches for brainstorming that have been mentioned each have their own advantages, but they all share the drawback of not providing an all-encompassing mature general model to address this issue or making intuitive feature observations on items or actions in the image. Prior to the onset of the era of big data and the spread of deep learning techniques, the efficiency and popularization of neural networks had achieved significant advances in the field of picture description and offered new hope. To tackle the description problem, a model framework has recently been developed. We have combined all components of this task, reviewed it, concentrated on the algorithmic core of various attention mechanisms, and explained how the attention mechanism is used. We provide an overview of the sizable datasets and standard practice rating criteria. Although there are many different image caption systems available today and image caption can be used for image retrieval, video caption, and video movement, testing results demonstrate that this task still requires higher performance systems and improvement. It struggles with the following three issues: first, how to produce entire phrases in natural language that are grammatically correct; second, how to construct the generated sentences; and third, how to provide captions semantics as clear as possible and consistent with the given image content. For future work, we propose the following four possible improvements:

1. An image is often rich in content. 0e model should be able to generate description sentences corresponding to multiple main objects for images with multiple target objects, instead of just describing a single target object.

2. For corpus description languages of different languages, a general image description system capable of handling multiple languages should be developed.

3. Evaluating the result of natural language generation systems is a difficult problem. 0e best way to evaluate the quality of automatically generated texts is subjective assessment by linguists, which is hard to achieve. In order to improve system performance, the evaluation indicators should be optimized to make them more in line with human experts' assessments.

4. A very real problem is the speed of training, testing, and generating sentences for the model should be

optimized to improve performance.

2.5 Problem Definition

A crucial task that pertains to both the field of computer vision and the field of natural language processing is the creation of captions for photographs. The ability of a machine to mimic human abilities to describe images with text is already a great advancement in artificial intelligence. In the past, computer systems have used predefined templates to generate text descriptions for images. The fundamental problem of this endeavor is to capture how objects connect to each other in the image and to communicate them in a natural language (like English). This method, however, falls short of offering the variation needed to produce lexically rich text descriptions. With neural networks' enhanced efficiency, this flaw has been eliminated. Numerous cutting-edge models use neural networks to produce subtitles by taking image as input and predicting next lexical unit in the output sentence. Despite the successes of many systems based on the Recurrent Neural Networks (RNN) many issues remain to be addressed. Among those issues the following two are prominent for most systems.

1. The Vanishing Gradient Problem.
2. Training an RNN is a very difficult task.

A deep learning system called a recurrent neural network is created to handle various challenging computer tasks including speech recognition and object classification. The understanding of each event is based on knowledge from earlier occurrences, and RNNs are built to manage a succession of events. The deeper RNNs would be ideal because they would have a greater memory capacity and superior capabilities. These could be used in numerous real-world use cases, including improved speech recognition and stock prediction. RNNs aren't frequently used in real-world applications, despite their seeming promise, because of the vanishing gradient issue.

2.6 Goals and Objectives

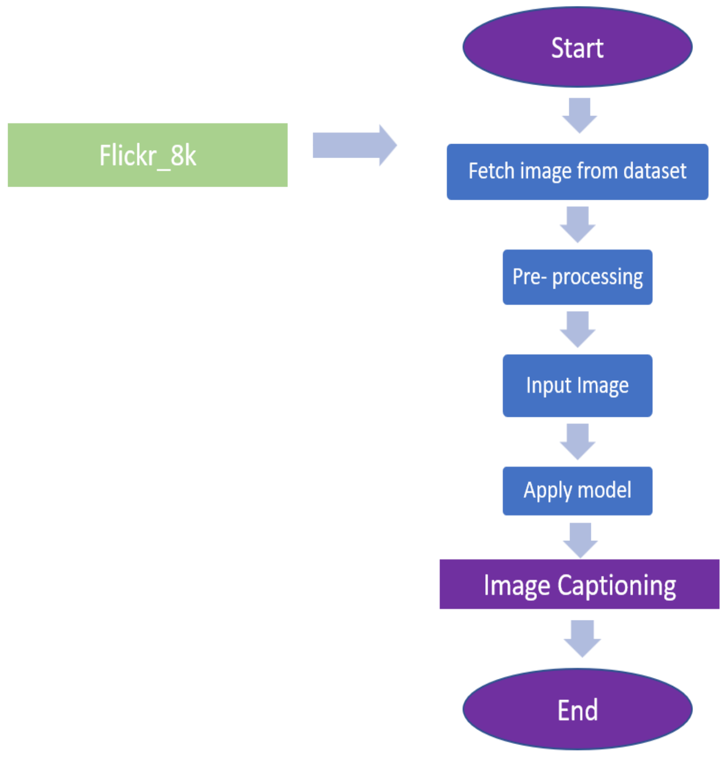
Image captioning has recently gathered a lot of attention specifically in the natural language domain. There is a pressing need for context based natural language description of images, however, this may seem a bit farfetched but recent developments in fields like neural networks, computer vision and natural language processing has paved a way for accurately describing images i.e. representing their visually grounded meaning. We are leveraging state-of-the-art techniques like Convolutional Neural Network (CNN), and appropriate datasets of images and their human perceived description to achieve the same. We demonstrate that our alignment model produces results in retrieval experiments on datasets such as Flicker_8k.

CHAPTER 3
DESIGN FLOW

3.1 Evaluation & Selection of Specifications/Features

Shuang Ma et al. introduced a multimodal image captioning technique in which they tackled the problem of translating instances from one modality to another without the help of paired data by leveraging an intermediate modality that was common to the two other modalities. In the paper, Shuang Ma et al. chose to translate images to speech and leveraged disjoint datasets with one shared modality, i.e., image-text pairs and text-speech pairs, with text as the shared modality. Since the shared modality is skipped during the generation process, they called this problem. Shuang et al. proposed a multimodal information bottleneck approach to tackle the problem. The model showed qualitative results on image-to-speech synthesis and also improves performance on traditional cross-modal generation.

Our approach is to infer these alignments and use them to learn a generative model of descriptions. We develop a deep neural network model that infers the alignment between segments of sentences and the region of the image that they describe. We introduce a Recurrent Neural Network architecture that takes an input image and generates its description in text. Our experiments show that the generated sentences produce sensible qualitative predictions



3.3 Analysis and Feature Finalization

Based on above constraints, we have selected following steps for our model:

1. Data Collection
2. Understanding the data
3. Data Cleaning
4. Loading the training set
5. Data Pre-processing – Images
6. Data Pre-processing – Captions
7. Data preparation using Generator Function
8. Model Architecture
9. Inference The overall workflow can be divided into these main steps:

1. Read Captions File: Reading the text and token flickr8k file , finding the length of the file and splitting it.

2. Data Cleaning: Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct.

3. Loading Training Testing Data: The process includes training Images File, testing it and creating a train description dictionary that adds starting and ending sequence

4. Data Preprocessing - Images: Loading the image, preprocessing and encoding it and testing it.

5. Data Preprocessing - Captions : Loading the captions ,appending the start and the end sequence , finding the maximum length of the caption

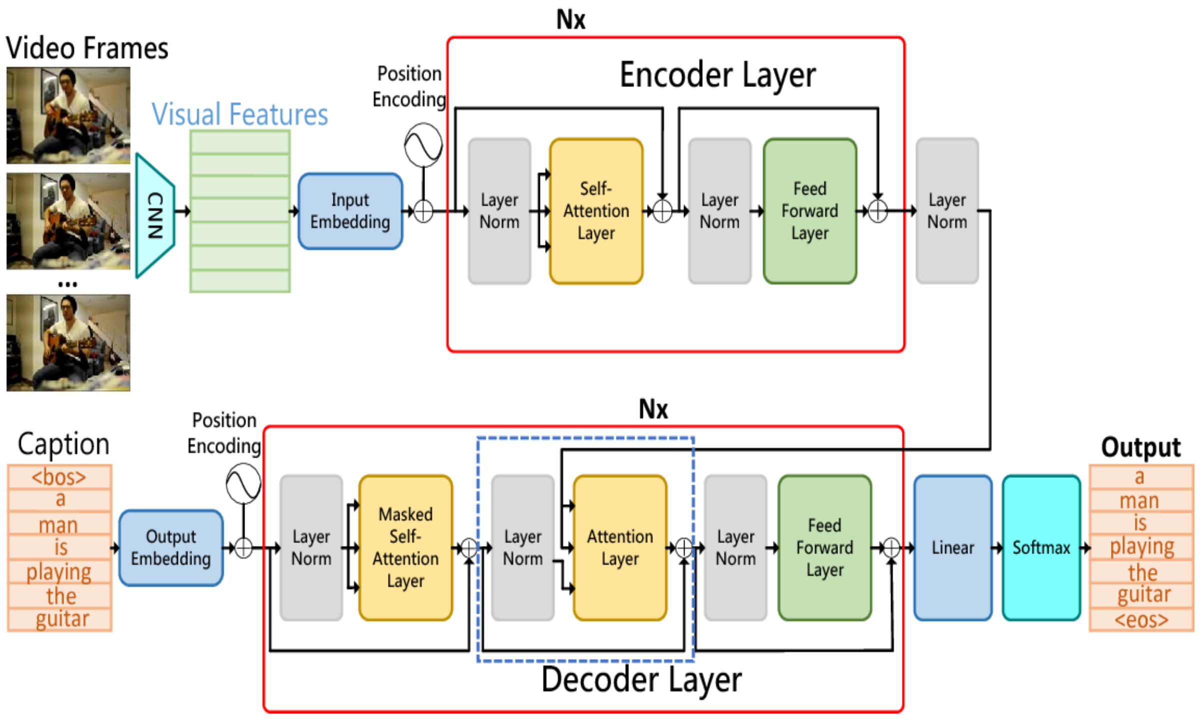
6. Data Preparation using Generator: Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. It is an important step prior to processing and often involves reformatting data, making corrections to data and the combining of data sets to enrich data.

7. Word Embedding: Converting words into Vectors (Embedding Layer Output) \

8. Model Architecture: Making an image feature extractor model, partial caption sequence model and merging the two networks

9. Train Our Model: A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output.

10. Predictions: Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome here predicting Caption for a photo



3.2 Design Flow

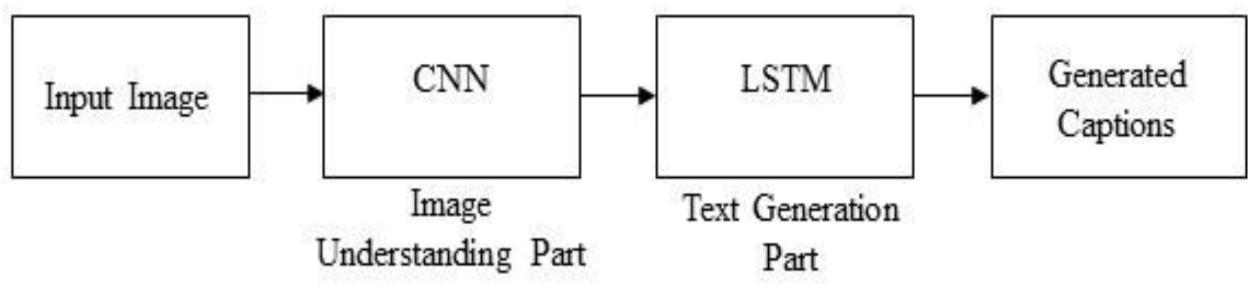
Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a natural language like English. So, to make an image caption generator model, we have merged these architectures. It is also called a CNN-RNN model.

3.4.1 CNN and LSTM Model

• CNN is used for extracting features from the image.

We will use the pre-trained model RESNET.

• LSTM will use the information from CNN to help generate a description of the image. Convolutional Neural Networks are customized deep neural networks that are capable of processing data with input shapes similar to a 2D matrix. CNN is particularly helpful when working with photos and can readily express images as a 2D matrix. CNN is primarily used to classify images and determine whether they are a bird, a jet, Superman, etc. Images are scanned from top to bottom and left to right to extract key details, which are then combined to identify the images. It can handle images that have been resized, rotated, translated, and perspectiveshifted. Long short-term memory, or LSTM, is a form of RNN (recurrent neural network) that is useful for solving issues involving sequence prediction. We can anticipate the following word based on the prior text. By getting over RNN's short-term memory restrictions, it has distinguished itself from regular RNN as an effective technology. Through the use of a forget gate, the LSTM may carry out relevant information while processing inputs and reject irrelevant information.



3.4.2 Other Deep Learning-Based Image Captioning

In our day-to-day life, data are increasing with unlabeled data because it is often impractical to accurately annotate data. Therefore, recently, researchers are focusing more on reinforcement learning and unsupervised learning-based techniques for image captioning.

Dense Captioning Vs Captions

for the Whole Scene In dense captioning, captions are generated for each region of the scene. Other methods generate captions for the whole scene.

Dense Captioning

The previous techniques for creating picture captions can only produce one caption for the entire image. To gather data on various things, they use various parts of the image. These techniques do not, however, produce regionally specific captions. DenseCap is a method for captioning images that Johnson et al. proposed. This technique localizes every important area of a picture before producing descriptions for each area. A typical method of this category has the following steps:

(1) Region proposals are generated for the different regions of the given image.

(2) CNN is used to obtain the region-based image features.

(3) The outputs of Step 2 are used by a language model to generate captions for every region.

Captions for the Whole Scene

Encoder-Decoder architecture, Compositional architecture, attention-based, semantic concept-based, stylized captions, Novel object-based image captioning, and other deep learning networks-based image captioning methods generate single or multiple captions for the whole scene.

3.5 Design Selection

We use two techniques mainly CNN and LSTM for image classification. So, to make our image caption generator model, we will be merging these architectures. It is also called a CNN-RNN model. • CNN is used for extracting features from the image. We will use the pre-trained model Xception. • LSTM will use the information from CNN to help generate a description of the image

3.4.1 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning method that can take in an input image, give various elements and objects in the image importance (learnable weights and biases), and be able to distinguish between them. Comparatively speaking, a ConvNet requires substantially less pre-processing than other classification techniques. Convolutional Neural Networks are customized deep neural networks that are capable of processing data with input shapes similar to a 2D matrix. CNN is particularly helpful when working with photos and can readily express images as a 2D matrix. Images are scanned from top to bottom and left to right to extract key details, which are then combined to identify the images. It is capable of handling translated images, rotated, scaled and changes in perspective

3.4.2 Transformer

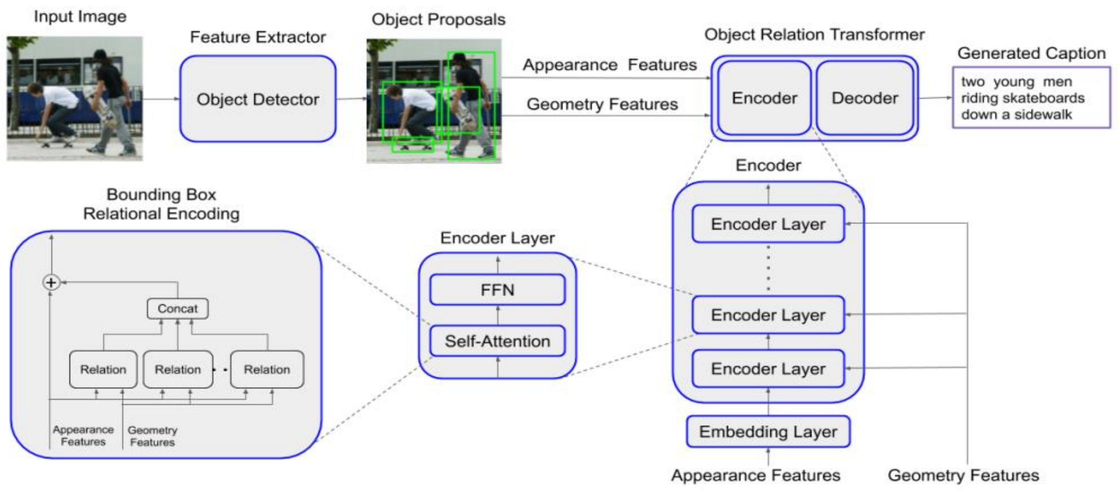


Figure 2: Overview of Object Relation Transformer architecture. The Bounding Box Relational Encoding diagram describes the changes made to the Transformer architecture

1. A TransformerEncoder: The extracted image features are then passed to a Transformer based encoder that generates a new representation of the inputs

2. A TransformerDecoder: This model takes the encoder output and the text data (sequences) as inputs and tries to learn to generate the caption.

The fully-attentive paradigm proposed by Vaswani et al. [62] in the seminal paper “Attention is all you need” has completely changed the perspective of language generation. Shortly after, the Transformer model became the building block of other breakthroughs in NLP, and the standard de-facto architecture for many language understanding tasks. As image captioning can be cast as a set-to-sequence problem when using image regions, the Transformer architecture has been employed also for this task. The standard Transformer decoder performs a masked self-attention operation, which is applied to words, followed.

3.5. Flickr8k Dataset

An openly accessible benchmark for image-to-set instructions is the Flickr8k dataset. This collection contains 8000 images, each with five captions. These images came from several Flickr groups. Each caption provides a thorough overview of the subjects and activities seen on camera. The dataset is larger since it shows a variety of scenes and events rather than only famous people or locations. There are 6000 photographs in the training dataset, 1000 in the development dataset, and 1000 in the test dataset. The dataset's qualities that make it suitable for this project are as follows: • The model becomes increasingly prevalent and is kept from being overfit by having numerous labels for a single image. • The image annotation model can work in numerous picture categories thanks to different training image categories, which makes the model more resilient Convolutional Neural Network (CNN) layers for feature extraction on input data are paired with LSTMs to facilitate sequence prediction in the CNN LSTM architecture. Although we will refer to LSTMs that employ a CNN as a front end in this lesson by the more general label "CNN LSTM," this architecture was initially known as a Long-term Recurrent Convolutional Network or LRCN model. To provide textual descriptions of photographs, this architecture is employed. Use of a CNN that has already been trained on a difficult picture classification task and has been repurposed as a feature extractor for the caption producing challenge is crucial. 3.5.4 Model Advantages The text file also contains the captions for the almost 8000 photographs that make up the Flickr 8k dataset that we used. Although deep learning-based image captioning techniques have made significant strides in recent years, a reliable technique that can provide captions of a high caliber for almost all photos has not yet been developed. Automatic picture captioning will continue to be a hot research topic for some time to come with the introduction of novel deep learning network architectures. With more people using social media every day and the majority of them

posting images, the potential for image captioning is very broad in the future. Therefore, they will benefit more from this project.

1. Assistance for visually impaired

• The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals.

• With AI-powered image caption generators, image descriptions can be read out to the visually impaired, enabling them to get a better sense of their surroundings.

3. Recommendations in editing

   • The image captioning model automates and accelerates the closed captioning process for digital content production, editing, delivery, and archival.

   • Well-trained models replace manual efforts for generating quality captions for images as well as videos. 3. Media and Publishing Houses

   • The media and public relations industry circulate tens of thousands of visual data across borders in the form of newsletters, emails, etc.

   • The image captioning model accelerates subtitle creation and enables executives to focus on more important tasks.

4. Self-driving cars

   • By using the captions generated by image caption generator self driving cars become aware of the surroundings and make decisions to control the car.

5. Reduce vehicle accidents

   • By installing an image caption generator

3.6 Methodology

3.6.1 Model Overview

The model is trained to maximize the probability of p(S|I), where S is the sequence of words produced by the model and each word St. is produced using a lexicon that was created using the training dataset. A deep vision convolutional neural network (CNN) is fed the input image I, which aids in object detection in the image. As shown in Figure 3.4, the Language Generating Recurrent Neural Network (RNN) receives the picture encodings and uses them to create a relevant phrase for the image. The model can be compared to a language translation RNN model, where the goal is to maximize the p(T|S), where T is the translation to the sentence S. However, in our model the encoder RNN which helps in transforming an input sentence to a fixed length vector is replaced by a CNN encoder. Recent research has shown that the CNN can easily transform an input image to a vector. We employ the pre-trained model CNN for the picture classification task. The next section discusses the models' specifics. The pre-trained CNN is then followed by an LSTM network. Language creation is accomplished using the LSTM network. Since a current token depends on the preceding tokens for a sentence to have sense, LSTM networks are different from typical Neural Networks in that they take this into consideration

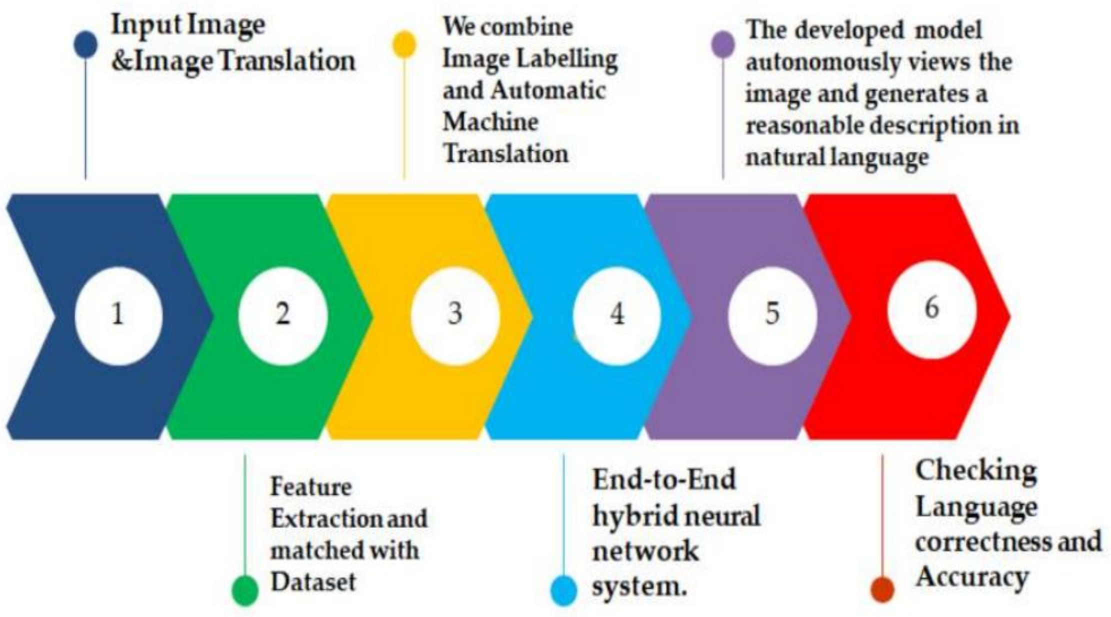

Input Image &Image Translation
We combine Image Labelling and Automatic Machine Translation
The developed model autonomously views the image and generates a reasonable description in natural language
1
2
3
4
5
6
Feature Extraction and matched with Dataset
End-to-End hybrid neural network system.
Checking Language correctness and Accuracy

Figure 3.4 Methodology Flow

We use a CNN pretrained model to the task of classifying images. The next section discusses the models' specifics. The CNN is followed by an LSTM network, which stands for long short-term memory. Language creation is accomplished using the LSTM network. Traditional Neural Networks are different from LSTM since a current token depends on the previous tokens for a sentence to be meaningful and LSTM networks take this factor into account

CHAPTER 4 RESULTS ANALYSIS AND VALIDATION

4.1 Implementation

Here we are showing full applied implementation of the given code and the all properly given prediction image and its given accurately resultant output.

4.1.1 Pre-Requisites

This project requires good knowledge of

Deep learning,

Python,

working on

Jupyter notebooks,

Keras library,

Numpy, and

Natural language processing .

Make sure you have installed all the following necessary libraries:

• pip install tensorflow

• keras

• pillow

• numpy

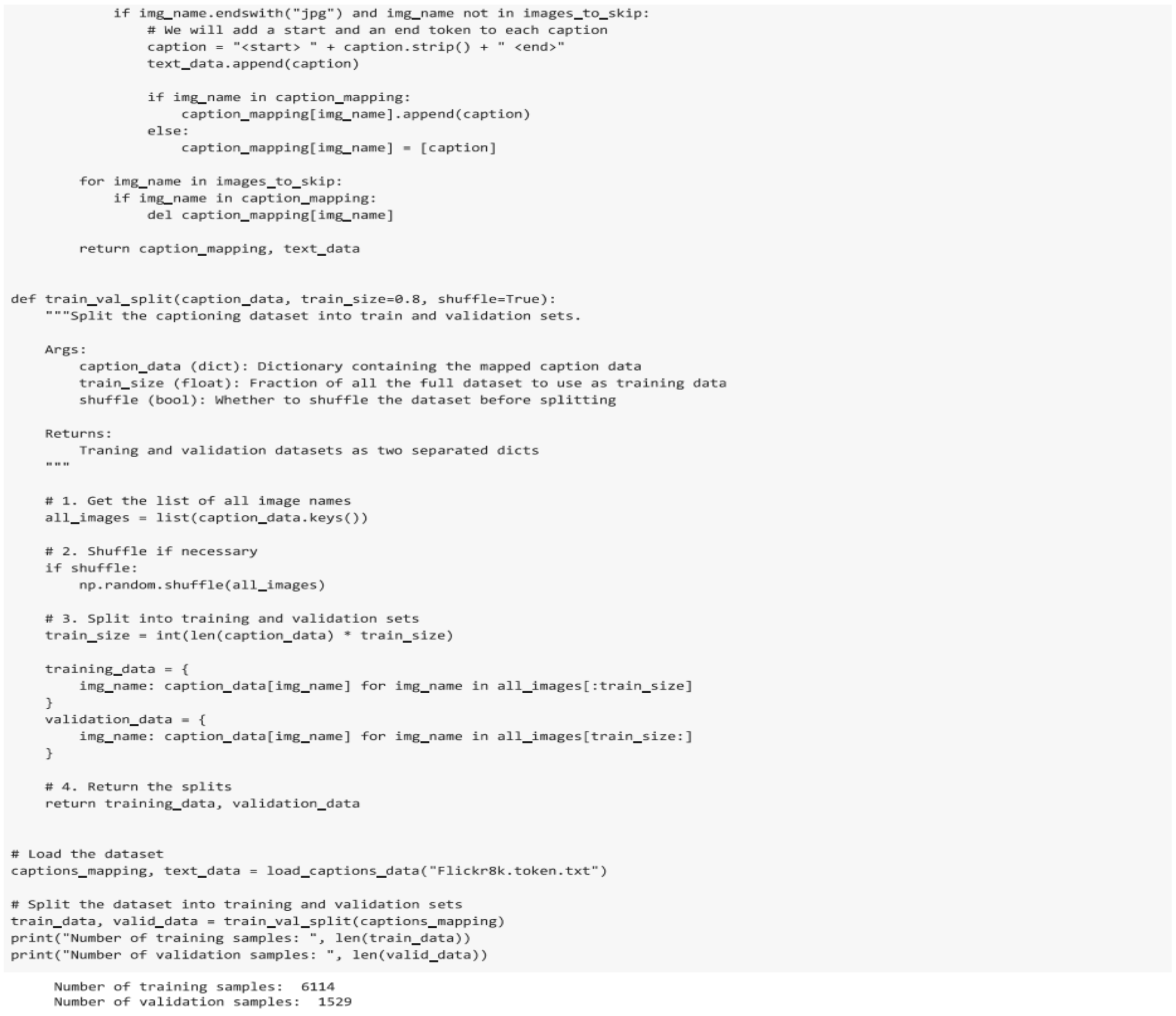
• tqdm

```
import os
import re
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import efficientnet
from tensorflow.keras.layers import TextVectorization


seed = 111
np.random.seed(seed)
tf.random.set_seed(seed)
```

```
!wget -q https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_Dataset.zip
!wget -q https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_text.zip
!unzip -qq Flickr8k_Dataset.zip
!unzip -qq Flickr8k_text.zip
!rm Flickr8k_Dataset.zip Flickr8k_text.zip
```

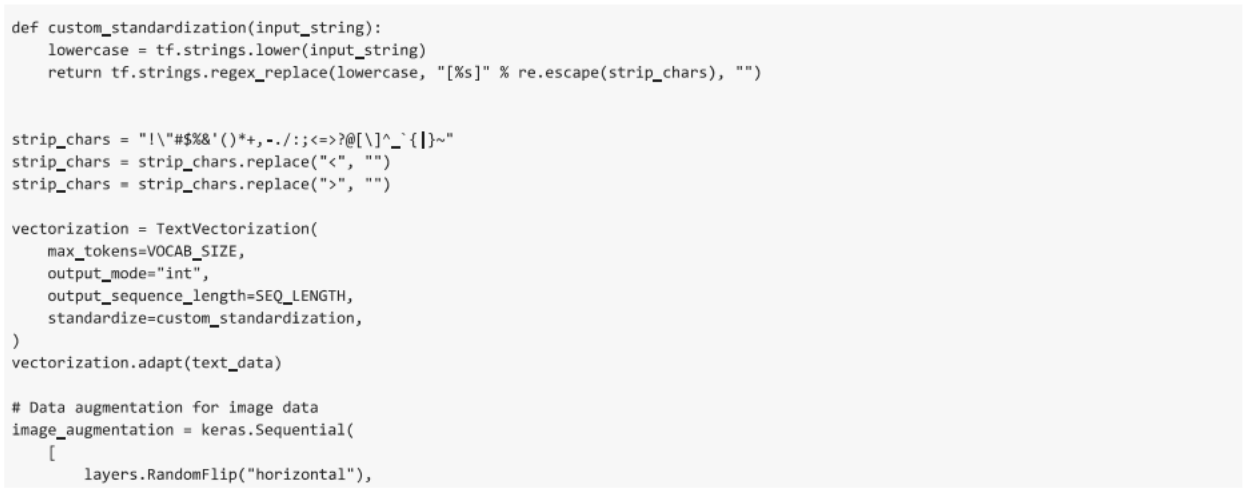
```
# Path to the images
IMAGES_PATH = "Flicker8k_Dataset"

# Desired image dimensions
IMAGE_SIZE = (299, 299)

# Vocabulary size
VOCAB_SIZE = 10000

# Fixed length allowed for any sequence
SEQ_LENGTH = 25

# Dimension for the image embeddings and token embeddings
EMBED_DIM = 512

# Per-layer units in the feed-forward network
FF_DIM = 512

# Other training parameters
BATCH_SIZE = 64
EPOCHS = 30
AUTOTUNE = tf.data.AUTOTUNE
```

```
def load_captions_data(filename):
    """Loads captions (text) data and maps them to corresponding images.

    Args:
        filename: Path to the text file containing caption data.

    Returns:
        caption_mapping: Dictionary mapping image names and the corresponding captions
        text_data: List containing all the available captions
    """

    with open(filename) as caption_file:
        caption_data = caption_file.readlines()
        caption_mapping = {}
        text_data = []
        images_to_skip = set()

        for line in caption_data:
            line = line.rstrip("\n")
            # Image name and captions are separated using a tab
            img_name, caption = line.split("\t")

            # Each image is repeated five times for the five different captions.
            # Each image name has a suffix `#(caption_number)`
            img_name = img_name.split("#")[0]
            img_name = os.path.join(IMAGES_PATH, img_name.strip())

            # We will remove caption that are either too short to too long
            tokens = caption.strip().split()

            if len(tokens) < 5 or len(tokens) > SEQ_LENGTH:
                images_to_skip.add(img_name)
                continue
```



```
            if img_name.endswith("jpg") and img_name not in images_to_skip:
                # We will add a start and an end token to each caption
                caption = "<start> " + caption.strip() + " <end>"
                text_data.append(caption)

                if img_name in caption_mapping:
                    caption_mapping[img_name].append(caption)
                else:
                    caption_mapping[img_name] = [caption]

        for img_name in images_to_skip:
            if img_name in caption_mapping:
                del caption_mapping[img_name]

        return caption_mapping, text_data


def train_val_split(caption_data, train_size=0.8, shuffle=True):
    """Split the captioning dataset into train and validation sets.

    Args:
        caption_data (dict): Dictionary containing the mapped caption data
        train_size (float): Fraction of all the full dataset to use as training data
        shuffle (bool): Whether to shuffle the dataset before splitting

    Returns:
        Traning and validation datasets as two separated dicts
    """

    # 1. Get the list of all image names
    all_images = list(caption_data.keys())

    # 2. Shuffle if necessary
    if shuffle:
        np.random.shuffle(all_images)

    # 3. Split into training and validation sets
    train_size = int(len(caption_data) * train_size)

    training_data = {
        img_name: caption_data[img_name] for img_name in all_images[:train_size]
    }
    validation_data = {
        img_name: caption_data[img_name] for img_name in all_images[train_size:]
    }

    # 4. Return the splits
    return training_data, validation_data


# Load the dataset
captions_mapping, text_data = load_captions_data("Flickr8k.token.txt")

# Split the dataset into training and validation sets
train_data, valid_data = train_val_split(captions_mapping)
print("Number of training samples: ", len(train_data))
print("Number of validation samples: ", len(valid_data))
```

```
      Number of training samples:  6114
      Number of validation samples:  1529
```



```
def custom_standardization(input_string):
    lowercase = tf.strings.lower(input_string)
    return tf.strings.regex_replace(lowercase, "[%s]" % re.escape(strip_chars), "")


strip_chars = "!\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"
strip_chars = strip_chars.replace("<", "")
strip_chars = strip_chars.replace(">", "")

vectorization = TextVectorization(
    max_tokens=VOCAB_SIZE,
    output_mode="int",
    output_sequence_length=SEQ_LENGTH,
    standardize=custom_standardization,
)
vectorization.adapt(text_data)

# Data augmentation for image data
image_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.2),
        layers.RandomContrast(0.3),
    ]
)
```

```
def decode_and_resize(img_path):
    img = tf.io.read_file(img_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, IMAGE_SIZE)
    img = tf.image.convert_image_dtype(img, tf.float32)
    return img


def process_input(img_path, captions):
    return decode_and_resize(img_path), vectorization(captions)


def make_dataset(images, captions):
    dataset = tf.data.Dataset.from_tensor_slices((images, captions))
    dataset = dataset.shuffle(BATCH_SIZE * 8)
    dataset = dataset.map(process_input, num_parallel_calls=AUTOTUNE)
    dataset = dataset.batch(BATCH_SIZE).prefetch(AUTOTUNE)

    return dataset


# Pass the list of images and the list of corresponding captions
train_dataset = make_dataset(list(train_data.keys()), list(train_data.values()))

valid_dataset = make_dataset(list(valid_data.keys()), list(valid_data.values()))
```

```
def get_cnn_model():
    base_model = efficientnet.EfficientNetB0(
        input_shape=(*IMAGE_SIZE, 3), include_top=False, weights="imagenet",
    )
    # We freeze our feature extractor
    base_model.trainable = False
    base_model_out = base_model.output
    base_model_out = layers.Reshape((-1, base_model_out.shape[-1]))(base_model_out)
    cnn_model = keras.models.Model(base_model.input, base_model_out)
    return cnn_model


class TransformerEncoderBlock(layers.Layer):
    def __init__(self, embed_dim, dense_dim, num_heads, **kwargs):
        super().__init__(**kwargs)
        self.embed_dim = embed_dim
        self.dense_dim = dense_dim
        self.num_heads = num_heads
        self.attention_1 = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.0
        )
        self.layernorm_1 = layers.LayerNormalization()
        self.layernorm_2 = layers.LayerNormalization()
        self.dense_1 = layers.Dense(embed_dim, activation="relu")

    def call(self, inputs, training, mask=None):
        inputs = self.layernorm_1(inputs)
        inputs = self.dense_1(inputs)

        attention_output_1 = self.attention_1(
            query=inputs,
            value=inputs,
            key=inputs,
            attention_mask=None,
            training=training,
        )
        out_1 = self.layernorm_2(inputs + attention_output_1)
        return out_1


class PositionalEmbedding(layers.Layer):
    def __init__(self, sequence_length, vocab_size, embed_dim, **kwargs):
        super().__init__(**kwargs)
        self.token_embeddings = layers.Embedding(
            input_dim=vocab_size, output_dim=embed_dim
        )
        self.position_embeddings = layers.Embedding(
            input_dim=sequence_length, output_dim=embed_dim
        )
        self.sequence_length = sequence_length
        self.vocab_size = vocab_size
        self.embed_dim = embed_dim
        self.embed_scale = tf.math.sqrt(tf.cast(embed_dim, tf.float32))

    def call(self, inputs):
        length = tf.shape(inputs)[-1]
        positions = tf.range(start=0, limit=length, delta=1)
        embedded_tokens = self.token_embeddings(inputs)
        embedded_tokens = embedded_tokens * self.embed_scale
        embedded_positions = self.position_embeddings(positions)
        return embedded_tokens + embedded_positions

    def compute_mask(self, inputs, mask=None):
        return tf.math.not_equal(inputs, 0)


class TransformerDecoderBlock(layers.Layer):
    def __init__(self, embed_dim, ff_dim, num_heads, **kwargs):
        super().__init__(**kwargs)
        self.embed_dim = embed_dim
        self.ff_dim = ff_dim
        self.num_heads = num_heads
        self.attention_1 = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.1
        )
        self.attention_2 = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.1
        )
        self.ffn_layer_1 = layers.Dense(ff_dim, activation="relu")
        self.ffn_layer_2 = layers.Dense(embed_dim)

        self.layernorm_1 = layers.LayerNormalization()
        self.layernorm_2 = layers.LayerNormalization()
        self.layernorm_3 = layers.LayerNormalization()

        self.embedding = PositionalEmbedding(
            embed_dim=EMBED_DIM, sequence_length=SEQ_LENGTH, vocab_size=VOCAB_SIZE
        )
        self.out = layers.Dense(VOCAB_SIZE, activation="softmax")

        self.dropout_1 = layers.Dropout(0.3)
        self.dropout_2 = layers.Dropout(0.5)
        self.supports_masking = True

    def call(self, inputs, encoder_outputs, training, mask=None):
        inputs = self.embedding(inputs)
        causal_mask = self.get_causal_attention_mask(inputs)

        if mask is not None:
            padding_mask = tf.cast(mask[:, :, tf.newaxis], dtype=tf.int32)
            combined_mask = tf.cast(mask[:, tf.newaxis, :], dtype=tf.int32)
            combined_mask = tf.minimum(combined_mask, causal_mask)
        input_shape = tf.shape(inputs)
        batch_size, sequence_length = input_shape[0], input_shape[1]
        i = tf.range(sequence_length)[:, tf.newaxis]
        j = tf.range(sequence_length)
        mask = tf.cast(i >= j, dtype="int32")
        mask = tf.reshape(mask, (1, input_shape[1], input_shape[1]))
        mult = tf.concat(
            [tf.expand_dims(batch_size, -1), tf.constant([1, 1], dtype=tf.int32)],
            axis=0,
        )
        return tf.tile(mask, mult)


class ImageCaptioningModel(keras.Model):
    def __init__(
        self, cnn_model, encoder, decoder, num_captions_per_image=5, image_aug=None,
    ):
        super().__init__()
        self.cnn_model = cnn_model
        self.encoder = encoder
        self.decoder = decoder
        self.loss_tracker = keras.metrics.Mean(name="loss")
        self.acc_tracker = keras.metrics.Mean(name="accuracy")
        self.num_captions_per_image = num_captions_per_image
        self.image_aug = image_aug

    def calculate_loss(self, y_true, y_pred, mask):
        loss = self.loss(y_true, y_pred)
        mask = tf.cast(mask, dtype=loss.dtype)
        loss *= mask
        return tf.reduce_sum(loss) / tf.reduce_sum(mask)

    def calculate_accuracy(self, y_true, y_pred, mask):
        accuracy = tf.equal(y_true, tf.argmax(y_pred, axis=2))
        accuracy = tf.math.logical_and(mask, accuracy)
        accuracy = tf.cast(accuracy, dtype=tf.float32)
        mask = tf.cast(mask, dtype=tf.float32)
        return tf.reduce_sum(accuracy) / tf.reduce_sum(mask)
```

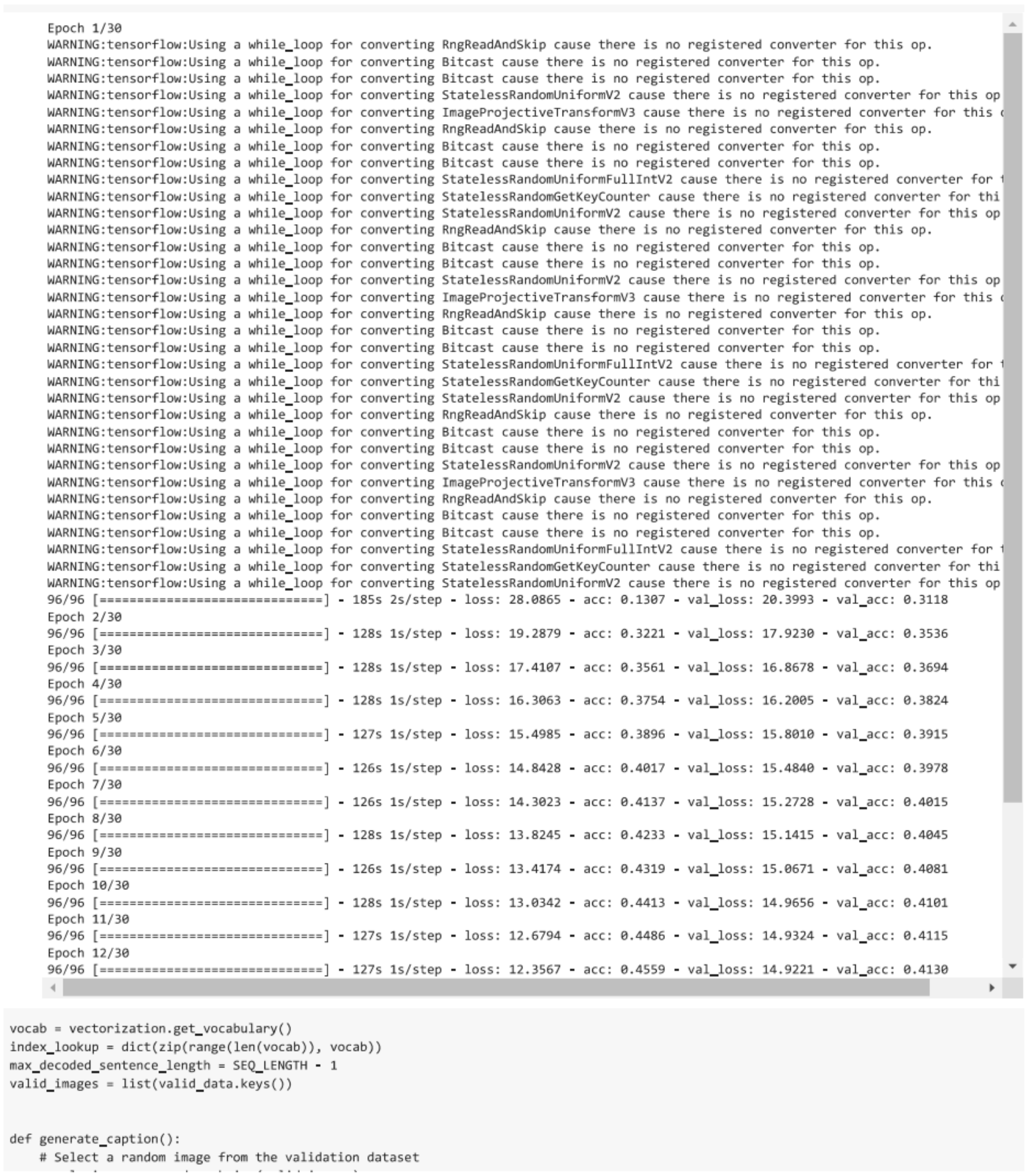
```
    def _compute_caption_loss_and_acc(self, img_embed, batch_seq, training=True):
        encoder_out = self.encoder(img_embed, training=training)
        batch_seq_inp = batch_seq[:, :-1]
        batch_seq_true = batch_seq[:, 1:]
        mask = tf.math.not_equal(batch_seq_true, 0)
        batch_seq_pred = self.decoder(
            batch_seq_inp, encoder_out, training=training, mask=mask
        )
        loss = self.calculate_loss(batch_seq_true, batch_seq_pred, mask)
        acc = self.calculate_accuracy(batch_seq_true, batch_seq_pred, mask)
        return loss, acc

    def train_step(self, batch_data):
        batch_img, batch_seq = batch_data
        batch_loss = 0
        batch_acc = 0

        if self.image_aug:
            batch_img = self.image_aug(batch_img)

        # 1. Get image embeddings
        img_embed = self.cnn_model(batch_img)

        # 2. Pass each of the five captions one by one to the decoder
        # along with the encoder outputs and compute the loss as well as accuracy
        # for each caption.
        for i in range(self.num_captions_per_image):
            with tf.GradientTape() as tape:
                loss, acc = self._compute_caption_loss_and_acc(
                    img_embed, batch_seq[:, i, :], training=True
                )

                # 3. Update loss and accuracy
                batch_loss += loss
                batch_acc += acc

            # 4. Get the list of all the trainable weights
            train_vars = (
                self.encoder.trainable_variables + self.decoder.trainable_variables
            )

            # 5. Get the gradients
            grads = tape.gradient(loss, train_vars)
            # 6. Update the trainable weights
            self.optimizer.apply_gradients(zip(grads, train_vars))

        # 7. Update the trackers
        batch_acc /= float(self.num_captions_per_image)
        self.loss_tracker.update_state(batch_loss)
        self.acc_tracker.update_state(batch_acc)

        # 8. Return the loss and accuracy values
        return {"loss": self.loss_tracker.result(), "acc": self.acc_tracker.result()}

    def test_step(self, batch_data):
        batch_img, batch_seq = batch_data
        batch_loss = 0
        batch_acc = 0

        # 1. Get image embeddings
        img_embed = self.cnn_model(batch_img)

        # 2. Pass each of the five captions one by one to the decoder
        # along with the encoder outputs and compute the loss as well as accuracy
        # for each caption.
        for i in range(self.num_captions_per_image):
            loss, acc = self._compute_caption_loss_and_acc(
                img_embed, batch_seq[:, i, :], training=False
            )

            # 3. Update batch loss and batch accuracy
            batch_loss += loss
            batch_acc += acc

        batch_acc /= float(self.num_captions_per_image)

        # 4. Update the trackers
        self.loss_tracker.update_state(batch_loss)
        self.acc_tracker.update_state(batch_acc)

        # 5. Return the loss and accuracy values
        return {"loss": self.loss_tracker.result(), "acc": self.acc_tracker.result()}

    @property
    def metrics(self):
        # We need to list our metrics here so the `reset_states()` can be
        # called automatically.
        return [self.loss_tracker, self.acc_tracker]


cnn_model = get_cnn_model()
encoder = TransformerEncoderBlock(embed_dim=EMBED_DIM, dense_dim=FF_DIM, num_heads=1)
decoder = TransformerDecoderBlock(embed_dim=EMBED_DIM, ff_dim=FF_DIM, num_heads=2)
caption_model = ImageCaptioningModel(
    cnn_model=cnn_model, encoder=encoder, decoder=decoder, image_aug=image_augmentation,
)
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 [==============================] - 2s 0us/step
```

```
    16705208/16705208 [==============================] - 2s 0us/step
```

```
# Define the loss function
cross_entropy = keras.losses.SparseCategoricalCrossentropy(
    from_logits=False, reduction="none"
)

# EarlyStopping criteria
early_stopping = keras.callbacks.EarlyStopping(patience=3, restore_best_weights=True)


# Learning Rate Scheduler for the optimizer
class LRSchedule(keras.optimizers.schedules.LearningRateSchedule):
    def __init__(self, post_warmup_learning_rate, warmup_steps):
        super().__init__()
        self.post_warmup_learning_rate = post_warmup_learning_rate
        self.warmup_steps = warmup_steps

    def __call__(self, step):
        global_step = tf.cast(step, tf.float32)
        warmup_steps = tf.cast(self.warmup_steps, tf.float32)
        warmup_progress = global_step / warmup_steps
        warmup_learning_rate = self.post_warmup_learning_rate * warmup_progress
        return tf.cond(
            global_step < warmup_steps,
            lambda: warmup_learning_rate,
```

```
            lambda: self.post_warmup_learning_rate,
        )


# Create a learning rate schedule
num_train_steps = len(train_dataset) * EPOCHS
num_warmup_steps = num_train_steps // 15
lr_schedule = LRSchedule(post_warmup_learning_rate=1e-4, warmup_steps=num_warmup_steps)

# Compile the model
caption_model.compile(optimizer=keras.optimizers.Adam(lr_schedule), loss=cross_entropy)

# Fit the model
caption_model.fit(
    train_dataset,
    epochs=EPOCHS,
    validation_data=valid_dataset,
    callbacks=[early_stopping],
)
```



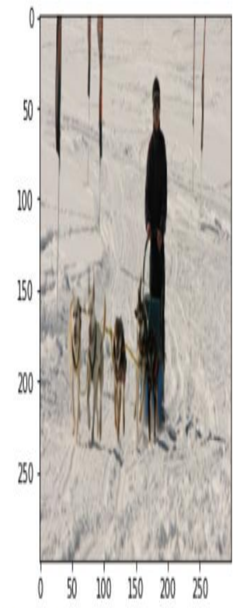
```
Epoch 1/30
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullIntV2 cause there is no registered converter for
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter cause there is no registered converter for thi
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullIntV2 cause there is no registered converter for
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter cause there is no registered converter for thi
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullIntV2 cause there is no registered converter for
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter cause there is no registered converter for thi
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op
96/96 [==============================] - 185s 2s/step - loss: 28.0865 - acc: 0.1307 - val_loss: 20.3993 - val_acc: 0.3118
Epoch 2/30
96/96 [==============================] - 128s 1s/step - loss: 19.2879 - acc: 0.3221 - val_loss: 17.9230 - val_acc: 0.3536
Epoch 3/30
96/96 [==============================] - 128s 1s/step - loss: 17.4107 - acc: 0.3561 - val_loss: 16.8678 - val_acc: 0.3694
Epoch 4/30
96/96 [==============================] - 128s 1s/step - loss: 16.3063 - acc: 0.3754 - val_loss: 16.2005 - val_acc: 0.3824
Epoch 5/30
96/96 [==============================] - 127s 1s/step - loss: 15.4985 - acc: 0.3896 - val_loss: 15.8010 - val_acc: 0.3915
Epoch 6/30
96/96 [==============================] - 126s 1s/step - loss: 14.8428 - acc: 0.4017 - val_loss: 15.4840 - val_acc: 0.3978
Epoch 7/30
96/96 [==============================] - 126s 1s/step - loss: 14.3023 - acc: 0.4137 - val_loss: 15.2728 - val_acc: 0.4015
Epoch 8/30
96/96 [==============================] - 128s 1s/step - loss: 13.8245 - acc: 0.4233 - val_loss: 15.1415 - val_acc: 0.4045
Epoch 9/30
96/96 [==============================] - 126s 1s/step - loss: 13.4174 - acc: 0.4319 - val_loss: 15.0671 - val_acc: 0.4081
Epoch 10/30
96/96 [==============================] - 128s 1s/step - loss: 13.0342 - acc: 0.4413 - val_loss: 14.9656 - val_acc: 0.4101
Epoch 11/30
96/96 [==============================] - 127s 1s/step - loss: 12.6794 - acc: 0.4486 - val_loss: 14.9324 - val_acc: 0.4115
Epoch 12/30
96/96 [==============================] - 127s 1s/step - loss: 12.3567 - acc: 0.4559 - val_loss: 14.9221 - val_acc: 0.4130
```

```python
vocab = vectorization.get_vocabulary()
index_lookup = dict(zip(range(len(vocab)), vocab))
max_decoded_sentence_length = SEQ_LENGTH - 1
valid_images = list(valid_data.keys())


def generate_caption():
    # Select a random image from the validation dataset
    sample_img = np.random.choice(valid_images)

    # Read the image from the disk
    sample_img = decode_and_resize(sample_img)
    img = sample_img.numpy().clip(0, 255).astype(np.uint8)
    plt.imshow(img)
    plt.show()

    # Pass the image to the CNN
    img = tf.expand_dims(sample_img, 0)
    img = caption_model.cnn_model(img)

    # Pass the image features to the Transformer encoder
    encoded_img = caption_model.encoder(img, training=False)

    # Generate the caption using the Transformer decoder
    decoded_caption = "<start> "
    for i in range(max_decoded_sentence_length):
        tokenized_caption = vectorization([decoded_caption])[:, :-1]
        mask = tf.math.not_equal(tokenized_caption, 0)
        predictions = caption_model.decoder(
            tokenized_caption, encoded_img, training=False, mask=mask
        )
        sampled_token_index = np.argmax(predictions[0, i, :])
        sampled_token = index_lookup[sampled_token_index]
        if sampled_token == " <end>":
            break
        decoded_caption += " " + sampled_token

    decoded_caption = decoded_caption.replace("<start> ", "")
    decoded_caption = decoded_caption.replace(" <end>", "").strip()
    print("Predicted Caption: ", decoded_caption)


# Check predictions for a few samples
generate_caption()
generate_caption()

generate_caption()
generate_caption()
```

Properly Generated Results:

```
Predicted Caption:  a basketball player in white is trying to block the ball
```

```
Predicted Caption:  two dogs are playing in a field
```



Predicted Caption: a man in a black jacket and a dog is walking through the snow

Predicted Caption: a black dog is running on a rocky beach

5.1 Conclusion

We reviewed deep learning-based picture captioning techniques in this study. We've provided a taxonomy of picture captioning methods, illustrated general block diagrams of the main groups, and highlighted the advantages and disadvantages of each. We talked about several evaluation criteria and datasets, as well as their advantages and disadvantages. The results of the experiment are also briefly summarized. We briefly discussed possible possibilities for future research in this field. Although deep learningbased image captioning techniques have made significant strides in recent years, a reliable technique that can provide captions of a high caliber for almost all photos has not yet been developed. Automatic picture captioning will continue to be a hot research topic for some time to come with the introduction of novel deep learning network architectures utilized here is Flickr 8k which includes nearly 8000 images, and the corresponding captions are also stored in the text file. Although deep learning-based image captioning techniques have made significant strides in recent years, a reliable technique that can provide captions of a high calibre for almost all photos has not yet been developed. Automatic picture captioning will continue to be a popular study topic for some time to come with the introduction of novel deep learning network architectures. With more people using social media every day and the majority of them posting images, the potential for image captioning is very broad in the future. Therefore, they will benefit more from this project.

5.2 Future Scope

Due to the internet's and social media's exponential rise in image content, image captioning has recently become a significant issue. This article reviews the many image retrieval studies conducted in the past while highlighting the various methods and strategies employed. There is a huge potential for future research in this area because feature extraction and similarity computation in photos are difficult tasks. Picture RETRIEVAL USING IMAGE CAPTIONING 54 Using features like color, tags, the histogram, and other features, current image retrieval algorithms calculate similarity. Since these approaches are independent of the image's context, findings cannot be entirely correct. Therefore, a thorough study of picture retrieval using the image context, such as image captioning, will help to resolve this issue in the future. By including new picture captioning datasets into the project's training, it will be possible to improve the identification of classes with lesser precision in the future.

Chapter 6

Reference

- **https://data.world/crowdflower/sentiment-analysis-in-text**
- **https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/**
- **https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/**
- **https://huggingface.co/microsoft/resnet-50**
- **https://www.kaggle.com/code/ngyptr/multi-class-classification-with-lstm/notebook/**
- **https://www.tensorflow.org/text/tutorials/classify_text_with_bert**
- **https://cs.stanford.edu/people/karpathy/cvpr2015.pdf**
- **https://arxiv.org/abs/1411.4555**
- **https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8**
- **https://towardsdatascience.com/how-to-easily-deploy-machine-learning-models-using-flask-b95af8fe34d4**

-