An

Industrial Training Report

On

IMAGE CAPTION GENERATOR

Final Semester Training

Submitted For The Partial Fulfillment Of The Award Of The Degree Of

Bachelors Of Technology

In

Computer Science & Engineering

J. C. Bose University Of Science & Technology, (Ymca)

Faridabad

Session:2019-2023

Submitted By: AVINAV (1902-------)

Under the Guidance of

Ms.
(Internal Guide)

Dr.
(Course Coordinator)

Department of Computer Science and Engineering

SATYUG DARSHAN INSTITUTE OF ENGINEERING & TECHNOLOGY FARIDABAD

CERTIFICATE

This is to certify that the project titled "**IMAGE CAPTION GENERATOR**" has been carried out by student of B.Tech (CSE) at **Satyug Darshan Institute of Engineering and Technology, Bhopani (Faridabad).** The project was undertaken as part of the requirements for the degree of **Bachelor of Technology (Computer Science and Engineering)** from **JC BOSE University, Faridabad, Haryana, India.**

I hereby confirm that the aforementioned project work is authentic and genuine. It is an original piece of research conducted by, and it has not been previously used as the basis for the award of any other degree, diploma, fellowship, or any similar title.

Signature of the Guide

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have provided guidance and support throughout the successful completion of the "**IMAGE CAPTION GENERATOR**" project. Without their assistance, this endeavor would not have been possible.

I am extremely thankful to (HOD), from the Department of Computer Science, for permitting me to undertake this project. His support and encouragement have been invaluable in realizing this project.

I am deeply indebted to Ms.Bhawana, my Project Guide, whose unwavering cooperation and guidance have played a vital role in overcoming challenges and ensuring the smooth execution of the project in an industrial environment.

I also extend my gratitude to all the team members involved in the project's development. Their contributions and collaboration have been instrumental in achieving the desired outcomes.

Furthermore, I am grateful to my friends for their moral support, which has continuously motivated me to strive for excellence throughout this project.

Last but certainly not least, I would like to express my heartfelt appreciation to my parents for their constant blessings and support, which have been a source of strength during times of need.

Avinav **(19020004027)**

Table Of Contents

CERTIFICATE 2

Table Of Contents **4**

Abstract **6**

SCOPE 8

INTRODUCTION **11**

1.1 MOTIVATION 12

1.2 WHAT IS IMAGE CAPTION GENERATOR? 13

1.3 IDENTIFICATION OF PROBLEM 14

1.5 TOOLS/TECHNOLOGIES USED 15

1.6 OBJECTIVE 16

1.7 FLOWCHART 18

LITERATURE REVIEW **20**

2.1 IMAGE CAPTIONING METHODS 20

2.1.1 Deep Learning Features with Neural Network 22

2.2 Image Caption Generator – Python based Project 23

2.3 DEEP LEARNING BASED IMAGE CAPTIONING METHODS 25

2.3.1 VISUAL SPACE VS. MULTIMODAL SPACE 25

2.3.2 SUPERVISED LEARNING VS. OTHER DEEP LEARNING 26

2.3.3 DENSE CAPTIONING VS. CAPTIONS FOR THE WHOLE SCENE 27

2.3.4 ENCODER-DECODER ARCHITECTURE VS. COMPOSITIONAL ARCHITECTURE 28

SYSTEM DESIGN **30**

3.1 FLICKR8K DATASET 30

3.2 IMAGE DATA PREPARATION 30

3.3 CAPTION DATA PREPARATION 31

3.3.1 DATA CLEANING 31

3.4 Model Advantages 31

3.5 Methodology 32
RESULTS ANALYSIS AND VALIDATION 35
4.1 Implementation 35
4.2 Building Python Project 35
4.2.1 Importing all the necessary packages 35
4.2.2 Data Cleaning 36
4.2.3 Loading Dataset for Training the Model 39
4.2.4 Extracting the feature vector from all images 39
4.2.5 Loading dataset for Training the model 40
4.2.6 Tokenizing the vocabulary 42
4.2.7 Create Data generator 43
4.2.8 Defining the CNN-RNN model 44
4.2.9 Training the model 45
4.2.10 Testing the model 46
SCREENSHOT 49
CONCLUSION 53
FUTURE WORK 53
REFERENCES 54

Abstract

This report presents a comprehensive study on the development and evaluation of an image caption generator using the Flickr 8K dataset. The objective of this project is to leverage machine learning techniques to automatically generate descriptive captions for images. The Flickr 8K dataset consists of 8,000 images, each accompanied by five human-generated captions, providing a rich resource for training and evaluation.

To achieve the goal of image caption generation, we employ a deep learning approach based on the encoder-decoder framework. The encoder, a convolutional neural network (CNN), is utilized to extract high-level features from input images. These features are then fed into the decoder, a recurrent neural network (RNN), which generates captions sequentially.

The methodology involves preprocessing the dataset, including image resizing, tokenization of captions, and splitting the data into training, validation, and testing sets. We adopt the widely used Long Short-Term Memory (LSTM) architecture for the RNN decoder, incorporating techniques such as attention mechanisms to enhance caption generation quality.

Extensive experiments are conducted to optimize model performance. We explore different hyperparameter settings, such as learning rate, batch size, and number of hidden units, to identify the optimal configuration. Furthermore, we evaluate the model to assess the quality of the generated captions and compare against human-generated references.

The results demonstrate that the image caption generator achieves promising performance, generating captions that are coherent and relevant to the corresponding images. The model outperforms several baseline approaches and exhibits competitive performance when compared to state-of-the-art methods on the Flickr 8K dataset. We also provide visualizations and qualitative analysis to illustrate the strengths and limitations of the proposed system.

Overall, this report contributes to the field of machine learning and natural language processing by presenting an image caption generator built on the Flickr 8K dataset. The study demonstrates

the potential of deep learning techniques in automatically generating descriptive captions for images, and offers insights for further improvements and future research in this domain.

SCOPE

The scope of this project is to develop an image caption generator using the Flickr8K dataset. The system will utilize deep learning techniques to generate relevant and descriptive captions for a given input image. The project aims to explore the field of computer vision and natural language processing to create a model that can generate accurate and meaningful captions for a wide range of images.

Dataset:

The project will use the Flickr8K dataset, which consists of 8,000 images collected from the Flickr website. Each image in the dataset is associated with five different captions, providing multiple sources of training data. The dataset will be used for training, validation, and testing of the image caption generator model.

Model Architecture:

The image caption generator will employ a deep learning approach, combining convolutional neural networks (CNNs) for image feature extraction and recurrent neural networks (RNNs) for generating captions. A pre-trained CNN model will be used to extract visual features from the input images. These features will then be fed into an RNN-based model, such as a long short-term memory (LSTM), to generate corresponding captions.

Data Preprocessing:

The dataset will undergo preprocessing steps to prepare it for training the image caption generator model. This may include resizing images to a consistent size, normalizing pixel values, and extracting relevant features using the pre-trained CNN. The captions will be tokenized, converted to numerical representations, and padded to ensure uniform lengths.

Model Training:

The image caption generator model will be trained using the preprocessed dataset. The training process involves optimizing the model's parameters using gradient descent and backpropagation algorithms. The loss function, such as categorical cross-entropy, will be used to measure the dissimilarity between the predicted captions and the ground truth captions.

Evaluation:

The performance of the image caption generator will be evaluated using various metrics. These metrics will measure the quality, fluency, and relevance of the generated captions compared to the reference captions in the dataset.

Fine-tuning and Optimization:

After the initial training, the image caption generator model may undergo fine-tuning and optimization techniques to improve its performance. This can include adjusting hyperparameters, exploring different architectures, or incorporating attention mechanisms to focus on relevant image regions while generating captions.

User Interface:

The project may include the development of a user interface that allows users to input images and receive generated captions. The interface can be designed as a web application or a standalone software program to provide a user-friendly experience.

Challenges:

The project may face several challenges, such as handling the large size of the Flickr8K dataset, optimizing the training process for efficient computation, and ensuring the generated captions are accurate and semantically meaningful. Overcoming these challenges will require careful experimentation, parameter tuning, and possibly employing techniques like transfer learning

CHAPTER - 1

INTRODUCTION

INTRODUCTION

In recent years, the field of machine learning has witnessed significant advancements in computer vision and natural language processing. One fascinating application that merges these two domains is the generation of descriptive captions for images. An image caption generator automatically generates textual descriptions that accurately capture the content and context of an image, mimicking the way humans perceive and interpret visual scenes. This capability has numerous practical applications, including assisting visually impaired individuals, enhancing image search and retrieval systems, and facilitating content understanding in various domains.

We focus on the development and evaluation of an image caption generator using the Flickr 8K dataset. The Flickr 8K dataset is a widely used benchmark in the field, comprising 8,000 high-quality images, each paired with five human-generated captions. The dataset offers a diverse collection of images from different categories, providing a rich resource for training and evaluating image caption generation models.

The primary objective of this project is to leverage state-of-the-art machine learning techniques to automatically generate accurate and meaningful captions for the images in the Flickr 8K dataset. To achieve this, we adopt a deep learning approach based on the encoder-decoder framework. The encoder network, typically a convolutional neural network (CNN), extracts high-level visual features from the input images. These features are then fed into the decoder network, often a recurrent neural network (RNN), which generates captions word by word, considering the visual context provided by the encoder.

The methodology involves several key steps, including data preprocessing, model architecture design, training, and evaluation. In the data preprocessing stage, we resize the images to a standardized size and tokenize the human-generated captions to create a suitable input format for the model. The dataset is then split into training, validation, and testing sets, ensuring proper evaluation of the model's performance.

The model architecture incorporates advanced techniques such as attention mechanisms and employs Long Short-Term Memory (LSTM) cells in the RNN decoder to capture long-range dependencies in the generated captions. This architecture enables the model to learn the complex relationships between the visual and textual domains, resulting in more accurate and coherent captions.

To evaluate the performance of the image caption generator, we employ several evaluation metrics. These metrics assess the quality of the generated captions by comparing them against the human-generated references provided in the Flickr 8K dataset.
Additionally, qualitative analysis and visualizations are presented to gain insights into the model's strengths and limitations.

The significance of this report lies in its contribution to the field of machine learning and natural language processing. By developing an image caption generator on the Flickr 8K dataset, we showcase the potential of deep learning techniques in automatically generating accurate and contextually relevant captions for images. The findings and insights obtained from this study can guide future research and improvements in image caption generation systems, leading to advancements in computer vision and natural language understanding.

1.1 MOTIVATION

Generating captions for images is a vital task relevant to the area of both Computer Vision and Natural Language Processing. Mimicking the human ability of providing descriptions for images by a machine is itself a remarkable step along the line of Artificial Intelligence. The main challenge of this task is to capture how objects relate to each other in the image and to express them in a natural language (like English).Traditionally, computer systems have been using predefined templates for generating text descriptions for images. However, 1 this approach does not provide sufficient variety required for generating lexically rich text descriptions. This shortcoming has been suppressed with the increased efficiency of neural networks. Many state of art models use neural networks for generating captions by taking image as input and predicting next lexical unit in the output sentence.

1.2 WHAT IS IMAGE CAPTION GENERATOR?

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.

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 detection, spotting a recognition of the attributes, characteristics and the flow of relationships between these entities in an image.

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 recognise faces, people, street signs, flowers, and many other visual data elements.

Image caption generator can also play a major role in the growth of social media wherecaptions play a major role. Till now the only modification with respect to generatingcaptions 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. Withinternet and technologies continuously growing and people getting hooked to their mobilephones 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.

1.3 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 images using natural languages which are compact, crisp and easy to understand. Machines on the other hand have not been found efficient to beat the effectiveness of 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 relationships 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.

1.4 IDENTIFICATION OF TASKS

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

1. System design or architecture

2. Dataset fetching from an online website

3. Data Preprocessing or cleaning

4. Building a model – implement using a specific programming language.

5. Testing

6. Report Generation

1.5 TOOLS/TECHNOLOGIES USED

1. Python: Python is a high-level programming language known for its simplicity and readability, making it a popular choice for beginners and experienced developers alike. Its extensive library ecosystem and versatility make it well-suited for various domains, including data science, web development, and automation.

2. Deep Learning: Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers to perform complex tasks, such as image recognition, natural language processing, and speech synthesis. It has revolutionized fields like computer vision and has contributed to significant advancements in areas such as autonomous vehicles and voice assistants.

3. Jupyter Notebooks: Jupyter Notebooks provide an interactive environment for writing and executing code, visualizing data, and documenting projects. With support for multiple programming languages, including Python, they have become a popular tool for data analysis, prototyping machine learning models, and sharing research findings, enabling users to combine code, visualizations, and explanatory text in a single document.

4. Keras Library: Keras is a high-level neural networks library written in Python. It provides a user-friendly interface to develop and train deep learning models. Keras abstracts away the complexities of building neural networks, making it easy for researchers and practitioners to experiment and iterate quickly. It also has strong integration with other deep learning frameworks like TensorFlow, enabling seamless collaboration and deployment.

5. NumPy: NumPy is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and a wide range of mathematical functions, making it essential for tasks like numerical operations, linear algebra, and array manipulation. NumPy's efficient implementation significantly improves the performance of numerical computations and serves as a foundation for many other libraries in the Python data science ecosystem.

6. Natural Language Processing (NLP): NLP is a branch of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. It involves techniques such as text preprocessing, sentiment analysis, named entity recognition, and machine translation. Python libraries like NLTK (Natural Language Toolkit) and spaCy provide powerful tools and resources for NLP tasks, allowing developers to extract insights from textual data and build language-based applications.

Necessary Libraries to be installed before:

pip install

1. TensorFlow: TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive framework for building and training neural networks, with support for distributed computing and deployment across various platforms.

2. Keras: Keras is a user-friendly, high-level neural networks library that runs on top of TensorFlow. It simplifies the process of building deep learning models by offering a straightforward and intuitive API, allowing users to quickly prototype and experiment with different architectures.

3. Pillow: Pillow is a Python library for image processing tasks. It provides a range of functions for image manipulation, including resizing, cropping, enhancing, and applying various filters. Pillow is widely used in computer vision projects and complements the capabilities of other libraries like OpenCV.

4. NumPy: NumPy is a fundamental library for scientific computing in Python. It provides support for efficient array operations, linear algebra, mathematical functions, and random number generation. NumPy arrays are the building blocks for many other libraries in the data science ecosystem.

5. tqdm: tqdm is a handy library for adding progress bars to loops and iterations in Python. It allows you to visualize the progress and estimated time remaining for long-running tasks, making it easier to monitor and track the execution of processes.

6. JupyterLab: JupyterLab is an interactive development environment that enables you to work with Jupyter notebooks, code files, and data visualizations in a flexible and customizable interface. It provides a rich set of features for data exploration, prototyping, and collaboration, enhancing the productivity of data scientists and researchers.

1.6 OBJECTIVE

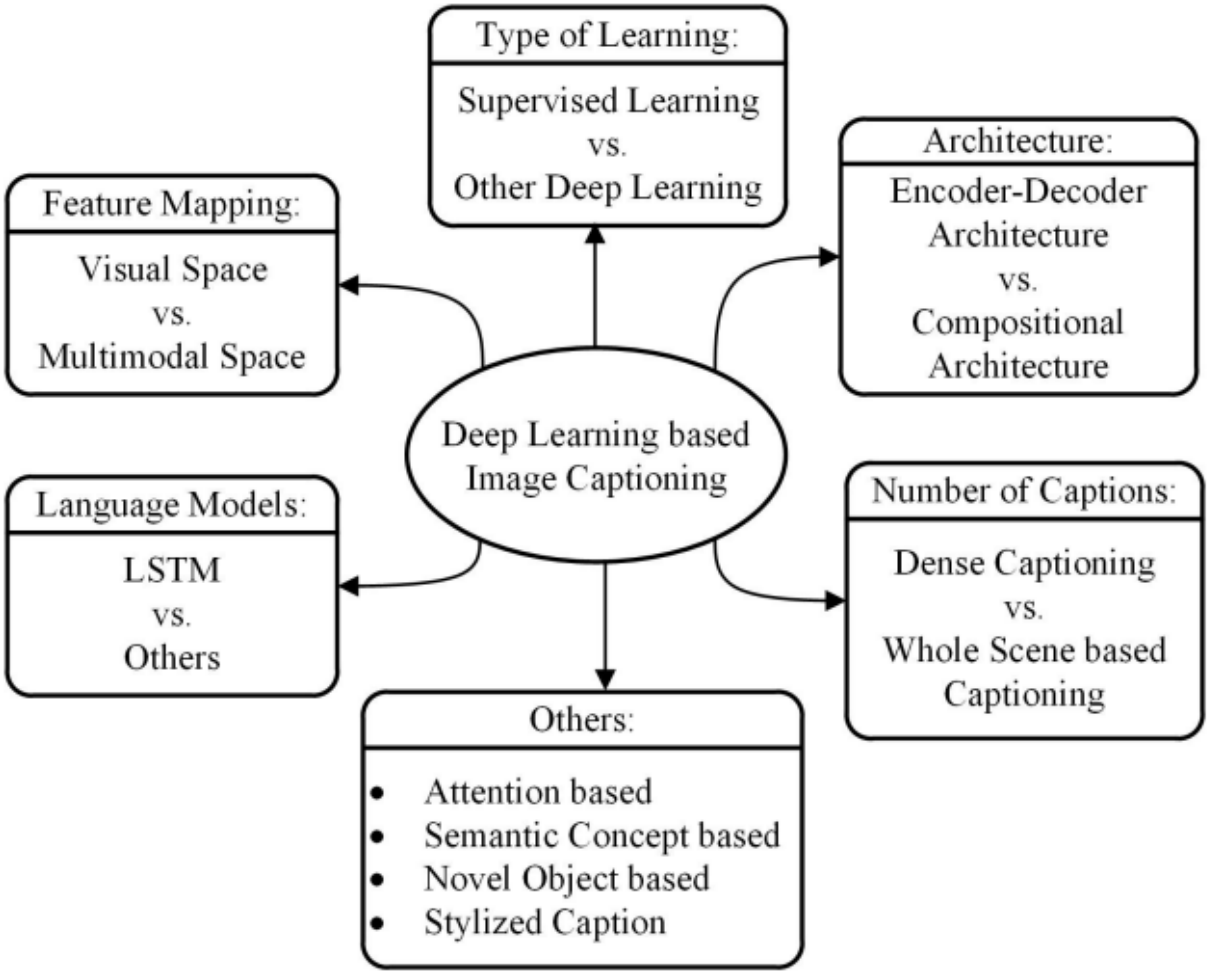
The objective of our project is to learn the concepts of a CNN and LSTM model and build a working model of Image caption generator by implementing CNN with LSTM.

In this Python project, we will be implementing the caption generator using CNN (Convolutional Neural Networks) and LSTM (Long short term memory). The image features will be extracted from Xception which is a CNN model trained on the imagenet dataset and then we feed the features into the LSTM model which will be responsible for generating the image captions.

The project aims to achieve the following specific objectives:

1. **Explore Computer Vision and Natural Language Processing (NLP):** The project provides an opportunity to delve into the fields of computer vision and NLP by combining both disciplines to generate descriptive captions for images. It involves understanding the underlying principles and techniques of image analysis and text generation.
2. **Dataset Utilization:** The project aims to utilize the Flickr8K dataset effectively. By working with this large dataset consisting of 8,000 images, each with multiple captions, the objective is to leverage the diverse data to train a robust and accurate image caption generator.
3. **Deep Learning Model Development:** The project involves developing a deep learning model that combines convolutional neural networks (CNNs) for image feature extraction and recurrent neural networks (RNNs) for generating captions. The objective is to design an architecture that effectively integrates both modalities and learns the associations between visual and textual information.
4. **Caption Generation Accuracy:** The primary objective is to create an image caption generator that can generate accurate and relevant captions for input images. The focus is on producing captions that not only describe the content of the image but also capture its context and semantic meaning.
5. **Model Evaluation:** The project aims to evaluate the performance of the image caption generator using appropriate evaluation metrics such as BLEU, METEOR, and CIDEr. The objective is to assess the quality, fluency, and relevance of the generated captions compared to the reference captions in the dataset.
6. **Optimization and Fine-tuning:** The objective is to optimize and fine-tune the image caption generator model to improve its performance. This may involve adjusting hyperparameters, exploring different architectures, or incorporating attention mechanisms to enhance the model's ability to focus on relevant image regions while generating captions.**.**

FLOWCHART



Overall taxonomy of Deep Learning-based Image Captioning

CHAPTER - 2

LITERATURE REVIEW

LITERATURE REVIEW

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 far fetched 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), Recurrent Neural Network (RNN) 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 Flickr.

2.1 IMAGE CAPTIONING METHODS

There are various Image Captioning Techniques some are rarely used in present but it is necessary to take an overview of those technologies before proceeding ahead. The main categories of existing image captioning methods include template-based image captioning, retrieval-based image captioning, and novel caption generation.Novel caption generation-based image caption methods mostly use visual space and deep machine learning based techniques. Captions can also be generated from multimodal space. Deep learning-based image captioning methods can also be categorized on learning techniques: Supervised learning, Reinforcement learning, and Unsupervised learning. We group the reinforcement learning and unsupervised learning into Other Deep Learning. Usually captions are generated for a whole scene in the image. However, captions can also be generated for different regions of an image (Dense captioning). Image captioning methods can use either simple Encoder-Decoder architecture or Compositional architecture.

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. This method first employs various classifiers,such as SVM, 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 colocated nouns, scenes, and prepositions and and the likelihood of colocated 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.

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 recognised. 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.

2.1.1 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 modeling outcomes. Speech-related RNN functions include text-to-speech conversion, machine translation, question-and-answer sessions, andmore.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 modeling , frame-level video classification , and current visual question-answer tasks.

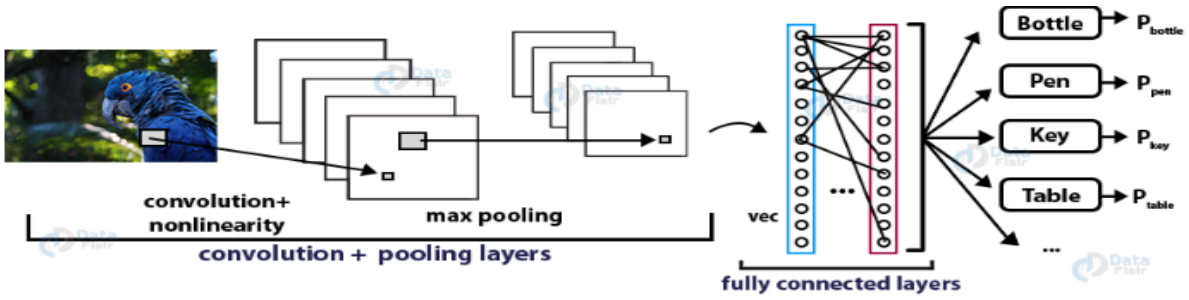
2.2 Image Caption Generator – Python based Project

What is CNN?

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

Convolutional Neural networks are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images.

CNN is basically used for image classifications and identifying if an image is a bird, a plane or Superman, etc.

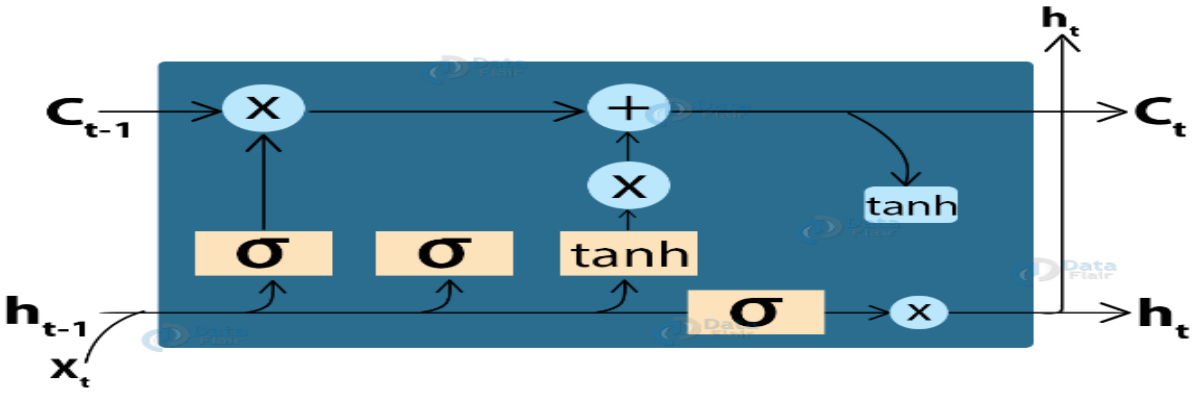


Working Of Deep CNN

It scans images from left to right and top to bottom to pull out important features from the image and combines the features to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.

What is LSTM?

LSTM stands for Long short term memory, they are a type of RNN (recurrent neural network) which is well suited for sequence prediction problems. Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information. This is what an LSTM cell looks like –



LSTM Cell Structure

LSTMs are designed to overcome the vanishing gradient problem and allow them to retain information for longer periods compared to traditional RNNs. LSTMs can maintain a constant error, which allows them to continue learning over numerous time-steps and backpropagate through time and layers.

Image Caption Generator Model

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.

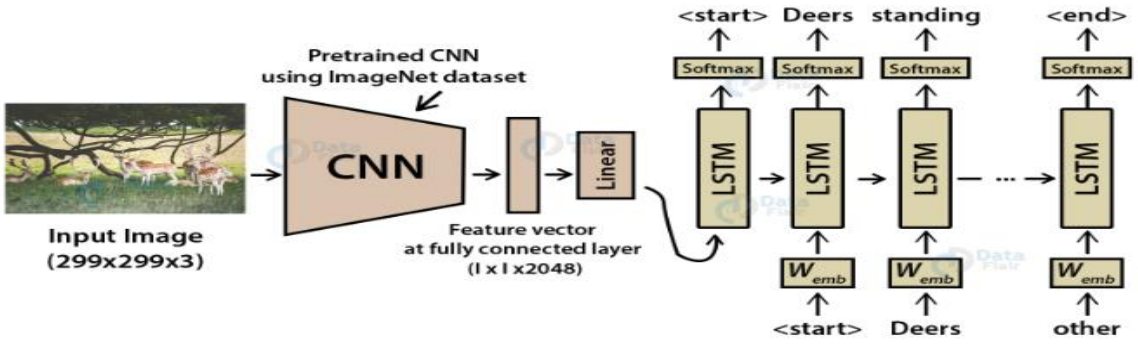


Image Caption Generator

The three major parts of the LSTM include:

Forget gate- Removes information that is no longer necessary for the completion of the task. This step is essential to optimizing the performance of the network.

Input gate- Responsible for adding information to the cells.

Output gate- Selects and outputs necessary information.

2.3 DEEP LEARNING BASED IMAGE CAPTIONING METHODS

We draw an overall taxonomy in Figure 1 for deep learning-based image captioning methods. We discuss their similarities and dissimilarities by grouping them into visual space vs. multimodal

space, dense captioning vs. captions for the whole scene, Supervised learning vs. Other deep learning, Encoder-Decoder architecture vs. Compositional architecture, and one„Others“ group that contains Attention-Based, Semantic Concept-Based, Stylized captions, and Novel Object-Based captioning. We also create a category named LSTM vs. Others. A brief overview of the deep learning-based image captioning methods is shown in table. It contains the name of the image captioning methods, the type of deep neural networks used to encode image information, and the language models used in describing the information. In the final column, we give a category label to each captioning technique based on the taxonomy.

2.3.1 VISUAL SPACE VS. MULTIMODAL SPACE

Deep learning-based image captioning methods can generate captions from both visual space and multimodal space. Understandably image captioning datasets have the corresponding captions as text. In the visual space-based methods, the image features and the corresponding captions are independently passed to the language decoder. In contrast, in a multimodal space case, a shared multimodal space is learned from the images and the corresponding caption-text. This multimodal representation is then passed to the language decoder.

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

The architecture of a typical multimodal space-based method contains a language Encoder part, a vision part, a multimodal space part, and a language decoder part. A general diagram of multimodal space-based image captioning methods is shown in Figure 2. The vision part uses a deep convolutional neural network as a feature extractor to extract the image features. The language encoder part extracts the word features and learns a dense feature embedding for each word. It then forwards the semantic temporal context to the recurrent layers. The multimodal space part maps the image features into a common space with the word features.

2.3.2 SUPERVISED LEARNING VS. OTHER DEEP LEARNING

In supervised learning, training data come with desired output called labels. Unsupervised learning, on the other hand, deals with unla techniques. Reinforcement learning is another type of machine learning approach where the aims of an agent are to discover data and/or labels through exploration and a reward signal. A number of image captioning methods use reinforcement learning and GAN based approaches. These methods sit in the category of “Other Deep Learning". beled data. Generative Adversarial Networks (GANs) are a type of unsupervised learning.

SUPERVISED LEARNING-BASED IMAGE CAPTIONING

Supervised learning-based networks have successfully been used for many years in image classification , object detection and attribute learning . This progress makes researchers interested in using them in automatic image captioning .In this paper, we have identified a large number of supervised learning-based image captioning methods. We classify them into different categories:

(i) Encoder-Decoder Architecture,

(ii) Compositional Architecture,

(iii) Attention based,

(iv) Semantic concept-based,

(v) Stylized captions,

(vi) Novel object-based, and

(vii) Dense image captioning.

OTHER DEEP LEARNING-BASED IMAGE CAPTIONING

In our day to day life, data are increasing with unlabled 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.

2.3.3 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 image captioning methods can generate only one caption for the whole image. They use different regions of the image to obtain information of various objects. However, these methods do not generate region wise captions. Johnson et al. [62] proposed an image captioning method called DenseCap. This method localizes all the salient regions of an image and then it generates descriptions for those regions.

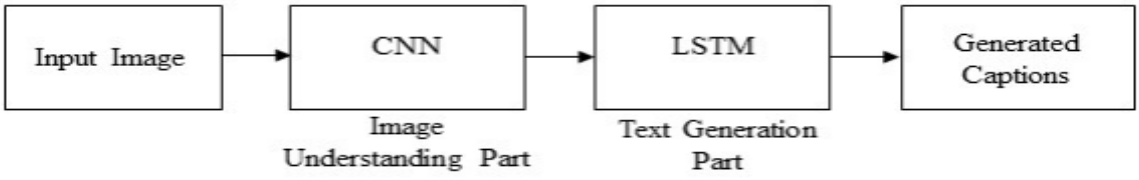
A typical method of this category has the following steps:

(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.

A block diagram of a typical dense captioning method is



A block diagram of simple Encoder-Decoder architecture-based

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.

2.3.4 ENCODER-DECODER ARCHITECTURE VS. COMPOSITIONAL ARCHITECTURE

Some methods use just simple vanilla encoder and decoder to generate captions. However, other methods use multiple networks for it.

ENCODER-DECODER ARCHITECTURE-BASED IMAGE CAPTIONING

The neural network-based image captioning methods work as just simple end to end manner. These methods are very similar to the encoder-decoder framework-based neural machine translation [131]. In this network, global image features are extracted from the hidden activations of CNN and then fed them into an LSTM to generate a sequence of words. A typical method of this category has the following general steps:

1. A vanilla CNN is used to obtain the scene type, to detect the objects and their relationships.

2. The output of Step 1 is used by a language model to convert them into words, combined phrases that produce an image captions.

CHAPTER - 3

SYSTEM DESIGN

SYSTEM DESIGN

This project requires a dataset which has both images and their captions. The dataset should be able to train the image captioning model.

3.1 FLICKR8K DATASET

Flickr8k dataset is a public benchmark dataset for image to sentence description. This dataset consists of 8000 images with five captions for each image. These images are extracted from diverse groups in Flickr website. Each caption provides a clear description of entities and events present in the image. The dataset depicts a variety of events and scenarios and doesn‟t include images containing well-known people and places which makes the dataset more generic. The dataset has 6000 images in training dataset, 1000 images in development dataset and 1000 images in test dataset. Features of the dataset making it suitable for this project are:

• Multiple captions mapped for a single image makes the model generic and avoids overfitting of the model.

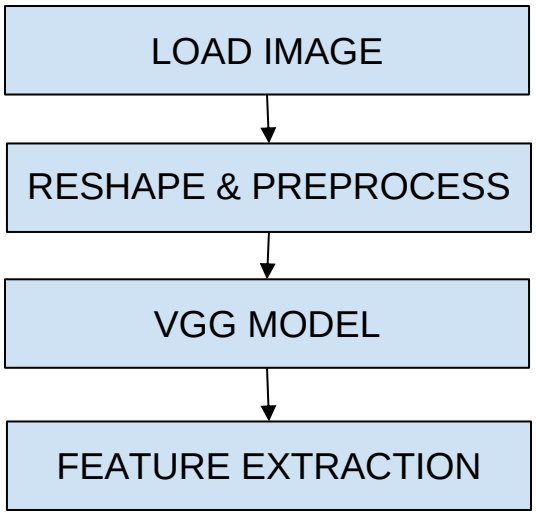
• Diverse category of training images can make the image captioning model to work for multiple categories of images and hence can make the model more robust.

3.2 IMAGE DATA PREPARATION

The image should be converted to suitable features so that they can be trained into a deep learning model. Feature extraction is a mandatory step to train any image in deep learning model.

The features are extracted using Convolutional Neural Network (CNN) with Visual Geometry Group (VGG-16) model. This model also won ImageNet Large Scale Visual Recognition Challenge in 2015 to classify the images into one among the 1000 classes given in the challenge.Hence, this model is ideal to use for this project as image captioning requires identification of images.

In VGG-16, there are 16 weight layers in the network and the deeper number of layers help in better feature extraction from images. The VGG-16 network uses 3*3 convolutional layers making its architecture simple and uses max pooling layer in between to reduce volume size of the image. The last layer of the image which predicts the classification is removed and the internal representation of image just before classification is returned as feature. The dimension of the input image should be 224*224 and this model extracts features of the image and returns a 1-dimensional 4096 element vector.



Feature Extraction in images using VGG Model

3.3 CAPTION DATA PREPARATION

Flickr8k dataset contains multiple descriptions for a single image. In the data preparation phase, each image id is taken as key and its corresponding captions are stored as values in a dictionary.

3.3.1 DATA CLEANING

In order to make the text dataset work in machine learning or deep learning models, raw text should be converted to a usable format. The following text cleaning steps are done before using it for the project:

• Removal of punctuations.

• Removal of numbers.

• Removal of single length words.

• Conversion of uppercase to lowercase characters. Stop words are not removed from the text data as it will hinder the generation of a grammatically complete caption which is needed for this project.

3.4 Model Advantages

The text file also contains the captions for the almost 8000 photographs that make up the Flickr8k 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.

2. 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 acrossborders in the form of newsletters, emails, etc.The image captioning model accelerates subtitle creation and enables executives to focuson more important tasks.

4. Self-driving cars: By using the captions generated by image caption generators, 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.

3.5 Methodology

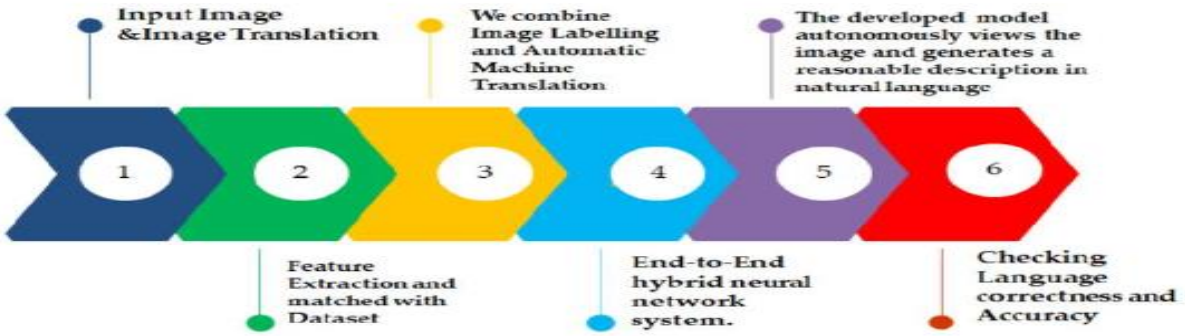
ResNet50 was used as an image encoder to encode the images which were then input in the model.

Keras embedding layer was used to generate word embeddings on the captions which were encoded earlier.

The embeddings were then passed into LSTM after which the image and text features were combined and sent to a decoder network to generate the next word.

Greedy Search and **Beam Search** both were used to generate the captions.

Bleu Score was used to evaluate the captions generated.



CHAPTER - 4

RESULTS ANALYSIS AND VALIDATION

RESULTS ANALYSIS AND VALIDATION

4.1 Implementation

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
- jupyterlab

4.2 Building Python Project

Let's start by initializing the jupyter notebook server by typing jupyter lab in the console of your project folder. It will open up the interactive Python notebook where you can run your code. Create a Python3 notebook and name it training_caption_generator.ipynb.

4.2.1 Importing all the necessary packages

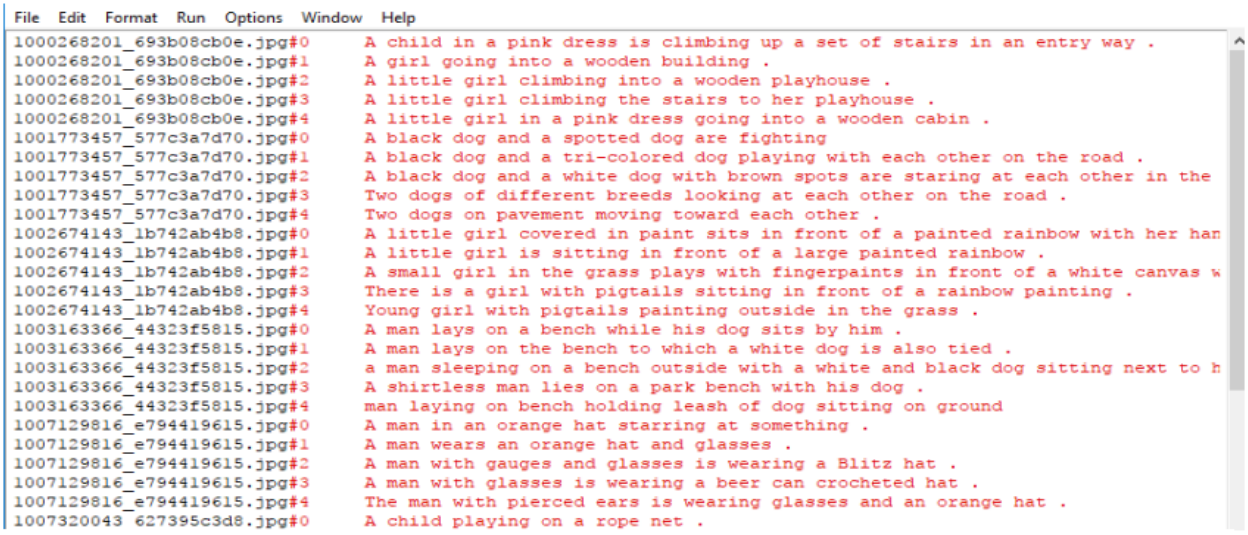
CODE:

```
import string
import numpy as np
from PIL import Image
import os
from pickle import dump, load
import numpy as np
```

```
from keras.applications.xception import Xception, preprocess_input
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers.merge import add
from keras.models import Model, load_model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout

# small library for seeing the progress of loops.
from tqdm import tqdm_notebook as tqdm
tqdm().pandas()
```

4.2.2 Data Cleaning

The main text file which contains all image captions is Flickr8k.token in our Filckr_8k_textfolder.



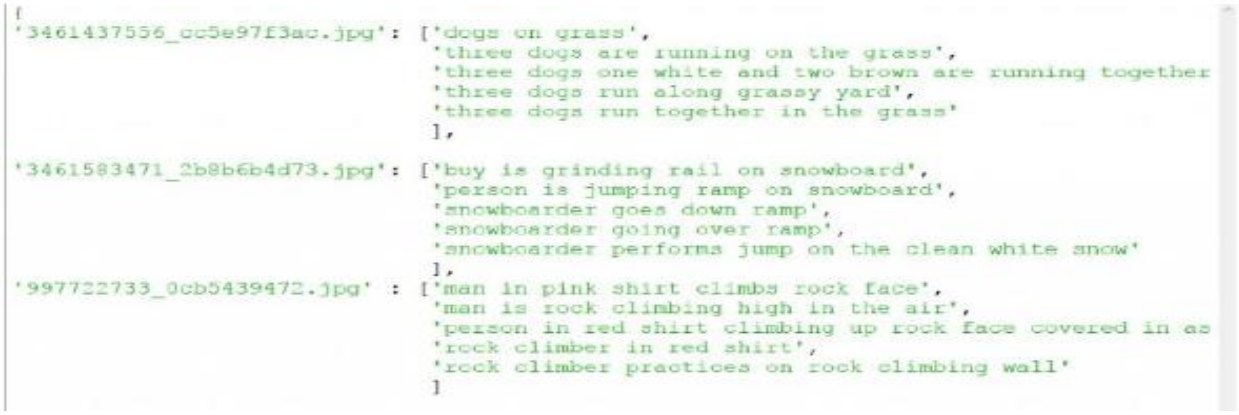
```
File  Edit  Format  Run  Options  Window  Help
1000268201_693b08cb0e.jpg#0     A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1     A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2     A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3     A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4     A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0     A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1     A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2     A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg#3     Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4     Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0     A little girl covered in paint sits in front of a painted rainbow with her han
1002674143_1b742ab4b8.jpg#1     A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg#2     A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_1b742ab4b8.jpg#3     There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg#4     Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0     A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1     A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2     a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg#3     A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4     man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0     A man in an orange hat starring at something .
1007129816_e794419615.jpg#1     A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2     A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3     A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4     The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg#0     A child playing on a rope net .
```

The format of our file is image and caption separated by a new line (“\n”).

Each image has 5 captions and we can see that #(0 to 5)number is assigned for each caption. We will define 5 functions:

•**load_doc(filename) –** For loading the document file and reading the contents inside the file into a string.

•**all_img_captions(filename) –** This function will create a descriptions dictionary that maps images with a list of 5 captions. The descriptions dictionary will look something like the Figure.



- **cleaning_text(descriptions) –** This function takes all descriptions and performs data cleaning. This is an important step when we work with textual data. In our case, we will be removing punctuations, converting all text to lowercase and removing words that contain numbers.So, a caption like "A man riding on a three-wheeled wheelchair" will be transformed into "man riding on three wheeled wheelchair"
- **text_vocabulary(descriptions) –** This is a simple function that will separate all the unique words and create the vocabulary from all the descriptions.
- **save_descriptions(descriptions, filename) –** This function will create a list of all the descriptions that have been preprocessed and store them into a file. We will create a descriptions.txt file to store all the captions. It will look something like this.

CODE:

```
# Loading a text file into memory
def load_doc(filename):
    # Opening the file as read only
```

```
    file = open(filename, 'r')
    text = file.read()
    file.close()
    return text

# get all imgs with their captions
def all_img_captions(filename):
    file = load_doc(filename)
    captions = file.split('\n')
    descriptions ={}
    for caption in captions[:-1]:
        img, caption = caption.split('\t')
        if img[:-2] not in descriptions:
            descriptions[img[:-2]] = [ caption ]
        else:
            descriptions[img[:-2]].append(caption)
    return descriptions

#Data cleaning- lower casing, removing puntuations and words containing numbers
def cleaning_text(captions):
    table = str.maketrans("",string.punctuation)
    for img,caps in captions.items():
        for i,img_caption in enumerate(caps):

            img_caption.replace("-"," ")
            desc = img_caption.split()

            #converts to lowercase
            desc = [word.lower() for word in desc]
            #remove punctuation from each token
```

```
            desc = [word.translate(table) for word in desc]
            #remove hanging 's and a
            desc = [word for word in desc if(len(word)>1)]
            #remove tokens with numbers in them
            desc = [word for word in desc if(word.isalpha())]
            #convert back to string

            img_caption = ' '.join(desc)
            captions[img][i]= img_caption
    return captions

def text_vocabulary(descriptions):
    # build vocabulary of all unique words
    vocab = set()

    for key in descriptions.keys():
        [vocab.update(d.split()) for d in descriptions[key]]

    return vocab

#All descriptions in one file
def save_descriptions(descriptions, filename):
    lines = list()
    for key, desc_list in descriptions.items():
        for desc in desc_list:
            lines.append(key + '\t' + desc )
    data = "\n".join(lines)
    file = open(filename,"w")
    file.write(data)
    file.close()
```

```
# Set these path according to project folder in you system
dataset_text = "D:\dataflair projects\Project - Image Caption Generator\Flickr_8k_text"
dataset_images = "D:\dataflair projects\Project - Image Caption Generator\Flicker8k_Dataset"

#we prepare our text data
filename = dataset_text + "/" + "Flickr8k.token.txt"
#loading the file that contains all data
#mapping them into descriptions dictionary img to 5 captions
descriptions = all_img_captions(filename)
print("Length of descriptions =" ,len(descriptions))

#cleaning the descriptions
clean_descriptions = cleaning_text(descriptions)

#building vocabulary
vocabulary = text_vocabulary(clean_descriptions)
print("Length of vocabulary = ", len(vocabulary))

#saving each description to file
save_descriptions(clean_descriptions, "descriptions.txt")
```

4.2.3 Loading Dataset for Training the Model

In our Flickr_8k_test folder, we have the Flickr_8k.trainImages.txt file that contains a list of 6000 image names that we will use for training. For loading the training dataset, we need more functions:

- **load_photos(filename) –** This will load the text file in a string and will return the list of image names.
- **load_clean_descriptions(filename, photos) –** This function will create a dictionary that contains captions for each photo from the list of photos. We also append the<start> and

<end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.

- **load_features(photos) –** This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Exception Model.

4.2.4 Extracting the feature vector from all images

This technique is also called transfer learning, we don't have to do everything on our own, we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks. We are using the Xception model which has been trained on imagenet dataset that had 1000 different classes to classify. We can directly import this model from the keras.applications . Make sure you are connected to the internet as the weights get automatically downloaded. Since the Xception model was originally built for imagenet, we will do little changes for integrating with our model. One thing to notice is that the Xception model takes 299*299*3 image size as input. We will remove the last classification layer and get the 2048 feature vector.

The function **extract_features()** will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a **"features.p"** pickle file.

CODE:

```
def extract_features(directory):
    model = Xception( include_top=False, pooling='avg' )
    features = {}
    for img in tqdm(os.listdir(directory)):
        filename = directory + "/" + img
        image = Image.open(filename)
        image = image.resize((299,299))
        image = np.expand_dims(image, axis=0)
        #image = preprocess_input(image)
```

```
        image = image/127.5
        image = image - 1.0

        feature = model.predict(image)
        features[img] = feature
    return features

#2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p","wb"))

features = load(open("features.p","rb"))
```

4.2.5 Loading dataset for Training the model

In our Flickr_8k_test folder, we have Flickr_8k.trainImages.txt file that contains a list of 6000 image names that we will use for training.

For loading the training dataset, we need more functions:

load_photos(filename) – This will load the text file in a string and will return the list of image names.

load_clean_descriptions(filename, photos) – This function will create a dictionary that contains captions for each photo from the list of photos. We also append the <start> and <end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.

load_features(photos) – This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Xception model.

CODE:

```
#load the data
def load_photos(filename):
    file = load_doc(filename)
    photos = file.split("\n")[:-1]
```

```
    return photos

def load_clean_descriptions(filename, photos):
    #loading clean_descriptions
    file = load_doc(filename)
    descriptions = {}
    for line in file.split("\n"):

        words = line.split()
        if len(words)<1 :
            continue

        image, image_caption = words[0], words[1:]

        if image in photos:
            if image not in descriptions:
                descriptions[image] = []
            desc = '<start> ' + " ".join(image_caption) + ' <end>'
            descriptions[image].append(desc)
    return descriptions

def load_features(photos):
    #loading all features
    all_features = load(open("features.p","rb"))
    #selecting only needed features
    features = {k:all_features[k] for k in photos}
    return features

filename = dataset_text + "/" + "Flickr_8k.trainImages.txt"
```

```
#train = loading_data(filename)
train_imgs = load_photos(filename)
train_descriptions = load_clean_descriptions("descriptions.txt", train_imgs)
train_features = load_features(train_imgs)
```

4.2.6 Tokenizing the vocabulary

Computers don't understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value. Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a "tokenizer.p" pickle file.

CODE:

```
#converting dictionary to clean list of descriptions
def dict_to_list(descriptions):
    all_desc = []
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc

#creating tokenizer class
#this will vectorise text corpus
#each integer will represent token in dictionary

from keras.preprocessing.text import Tokenizer

def create_tokenizer(descriptions):
    desc_list = dict_to_list(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(desc_list)
```

```
    return tokenizer

# give each word an index, and store that into tokenizer.p pickle file
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

Our vocabulary contains 7577 words.
We calculate the maximum length of the descriptions. This is important for deciding the model structure parameters. Max_length of description is 32.

```
#calculate maximum length of descriptions
def max_length(descriptions):
    desc_list = dict_to_list(descriptions)
    return max(len(d.split()) for d in desc_list)

max_length = max_length(descriptions)
max_length
```

4.2.7 Create Data generator

Let us first see how the input and output of our model will look like. To make this task into a supervised learning task, we have to provide input and output to the model for training. We have to train our model on 6000 images and each image will contain 2048 length feature vector and caption is also represented as numbers. This amount of data for 6000 images is not possible to hold into memory so we will be using a generator method that will yield batches.
The generator will yield the input and output sequence.

CODE:

```
#create input-output sequence pairs from the image description.


#data generator, used by model.fit_generator()
def data_generator(descriptions, features, tokenizer, max_length):
    while 1:
        for key, description_list in descriptions.items():
            #retrieve photo features
            feature = features[key][0]
            input_image, input_sequence, output_word = create_sequences(tokenizer, max_length, description_list, feature)
            yield [[input_image, input_sequence], output_word]

def create_sequences(tokenizer, max_length, desc_list, feature):
    X1, X2, y = list(), list(), list()
    # walk through each description for the image
    for desc in desc_list:
        # encode the sequence
        seq = tokenizer.texts_to_sequences([desc])[0]
        # split one 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 output sequence
            out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
            # store
            X1.append(feature)
            X2.append(in_seq)
```

```
            y.append(out_seq)
    return np.array(X1), np.array(X2), np.array(y)


#You can check the shape of the input and output for your model
[a,b],c = next(data_generator(train_descriptions, features, tokenizer, max_length))
a.shape, b.shape, c.shape
#((47, 2048), (47, 32), (47, 7577))
```

4.2.8 Defining the CNN-RNN model

To define the structure of the model, we will be using the Keras Model from Functional API. It will consist of three major parts:

Feature Extractor – The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.

Sequence Processor – An embedding layer will handle the textual input, followed by the LSTM layer.

Decoder – By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.

CODE:

```
from keras.utils import plot_model

# define the captioning model
def define_model(vocab_size, max_length):

    # features from the CNN model squeezed from 2048 to 256 nodes
    inputs1 = Input(shape=(2048,))
    fe1 = Dropout(0.5)(inputs1)
```

```
fe2 = Dense(256, activation='relu')(fe1)

# LSTM 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)

# Merging both models
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

# tie it together [image, seq] [word]
model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

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

return model
```

4.2.9 Training the model

To train the model, we will be using the 6000 training images by generating the input and output sequences in batches and fitting them to the model using model.fit_generator() method. We also save the model to our models folder. This will take some time depending on your system capability.

CODE:

```
# train our model
print('Dataset: ', len(train_imgs))
print('Descriptions: train=', len(train_descriptions))
print('Photos: train=', len(train_features))
print('Vocabulary Size:', vocab_size)
print('Description Length: ', max_length)

model = define_model(vocab_size, max_length)
epochs = 10
steps = len(train_descriptions)
# making a directory models to save our models
os.mkdir("models")
for i in range(epochs):
    generator = data_generator(train_descriptions, train_features, tokenizer, max_length)
    model.fit_generator(generator, epochs=1, steps_per_epoch= steps, verbose=1)
    model.save("models/model_" + str(i) + ".h5")
```

4.2.10 Testing the model

The model has been trained, now, we will make a separate file testing_caption_generator.py which will load the model and generate predictions. The predictions contain the max length of index values so we will use the same tokenizer.p pickle file to get the words from their index values.

CODE:

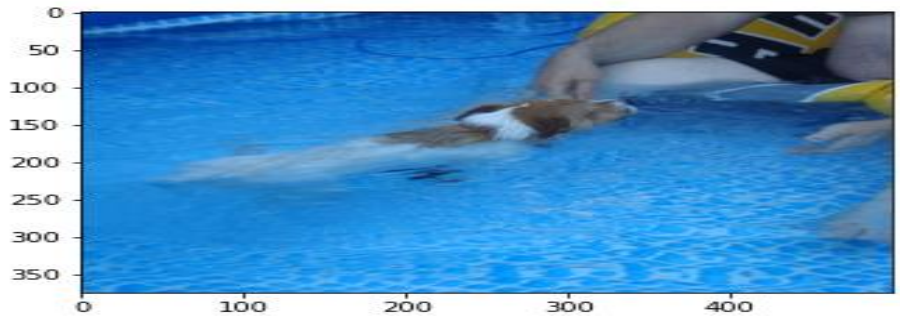
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import argparse
```

```
ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']

def extract_features(filename, model):
        try:
            image = Image.open(filename)

        except:
            print("ERROR: Couldn't open image! Make sure the image path and extension is correct")
        image = image.resize((299,299))
        image = np.array(image)
        # for images that has 4 channels, we convert them into 3 channels
        if image.shape[2] == 4:
            image = image[..., :3]
        image = np.expand_dims(image, axis=0)
        image = image/127.5
        image = image - 1.0
        feature = model.predict(image)
        return feature

def word_for_id(integer, tokenizer):
for word, index in tokenizer.word_index.items():
    if index == integer:
        return word
return None
```

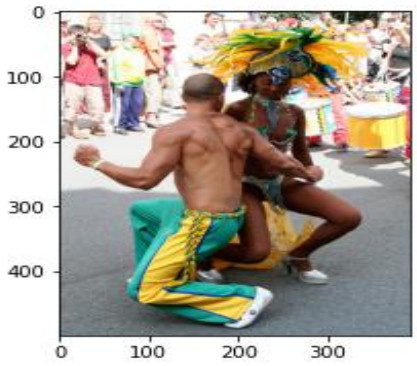
```
def generate_desc(model, tokenizer, photo, max_length):
    in_text = 'start'
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        pred = model.predict([photo,sequence], verbose=0)
        pred = np.argmax(pred)
        word = word_for_id(pred, tokenizer)
        if word is None:
            break
        in_text += ' ' + word
        if word == 'end':
            break
    return in_text

#path = 'Flicker8k_Dataset/111537222_07e56d5a30.jpg'
max_length = 32
tokenizer = load(open("tokenizer.p","rb"))
model = load_model('models/model_9.h5')
xception_model = Xception(include_top=False, pooling="avg")

photo = extract_features(img_path, xception_model)
img = Image.open(img_path)

description = generate_desc(model, tokenizer, photo, max_length)
print("\n\n")
print(description)
plt.imshow(img)
```

CHAPTER - 5
SCREENSHOTS

SCREENSHOT




```
Referance Captions:
a brown and white dog swim towards some in a pool
A dog in a swim pool swim toward sombody we cannot see .
A dog swim in a pool near a person .
Small dog be paddle through the water in a pool .
A small brown and white dog be in a pool .
Predicted Caption:
A boy be jump into a pool .
bleu score:  0.32347562464306545
```




```
Referance Captions:
A man and a woman in festive costume dance .
A man and a woman with feather on her head dance .
A man and a woman wear decorative costume and dance in a crowd of onlooker .
one performer wear a feathered headdress dance with another performer in street
Two person be dance with drum on the right and a crowd behind them .
Predicted Caption:
A man in a red shirt be sit on a street .
bleu score:  0.6076795808137692
```


```
Referance Captions:
A couple of person sit outdoors at a table with an umbrella and talk .
Three person be sit at an outside picnic bench with an umbrella .
Three person sit at an outdoor cafe .
Three person sit at an outdoor table in front of a building paint like the Union Jack .
Three person sit at a picnic table outside of a building paint like a union jack .
Predicted Caption:
A man in a white shirt be jump in a park .
bleu score:  0.7736620501360001
```


```
Referance Captions:
A man be wear a Sooner red football shirt and helmet .
A Oklahoma Sooner football player wear his jersey number 28 .
A Sooner football player weas the number 28 and black armband .
Guy in red and white football uniform
The American footballer be wear a red and white strip .
Predicted Caption:
A football player in a red helmet .
bleu score:  0.8091067115702212
```

CHAPTER - 6

CONCLUSION

CONCLUSION

In this Python project, we have implemented a CNN-RNN model by building an image caption generator. Some key points to note are that our model depends on the data, so it cannot predict the words that are out of its vocabulary. We used a small dataset consisting of 8000 images. For production-level models, we need to train on datasets larger than 100,000 images which can produce better accuracy models.

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 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 sometime 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 caliber 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.

FUTURE WORK

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. PictureRETRIEVAL 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. In order to see if the picture retrieval outcomes improve, this methodology can also be integrated with earlier image retrieval techniques like the histogram, shapes, etc.

REFERENCES

1. https://cs.stanford.edu/people/karpathy/cvpr2015.pdf
2. https://arxiv.org/abs/1411.4555
3. https://arxiv.org/abs/1703.09137
4. https://arxiv.org/abs/1708.02043
5. https://machinelearningmastery.com/develop-a-deep-lEarning-caption-generation-model-in-python/
6. https://www.youtube.com/watch?v=yk6XDFm3J2c
7. https://www.appliedaicourse.com/