A REPORT ON

IMAGE CAPTIONING WITH VISUAL ATTENTION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

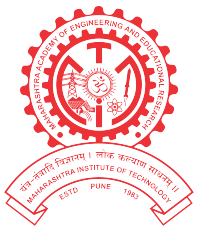
SUBMITTED BY

SAMBHAV AGARWAL	**B150024201**
HIMANI MALI	**B150024250**
PRITHVIRAJ KHELKAR	**B150024271**
SOHAM MAHABALESHWARKAR	**B150024282**

DEPARTMENT OF COMPUTER ENGINEERING

MAEER'S
MAHARASHTRA INSTITUTE OF TECHNOLOGY
PAUD RD-KOTHRUD,PUNE 411038

SAVITRIBAI PHULE PUNE UNIVERSITY
2018-2019



C.E.R.T.I.F.I.C.A.T.E.

This is to certify that the project report entitled

"IMAGE CAPTIONING WITH VISUAL ATTENTION"

Submitted by

SAMBHAV AGARWAL	**B150024201**
HIMANI MALI	**B150024250**
PRITHVIRAJ KHELKAR	**B150024271**
SOHAM MAHABALESHWARKAR	**B150024282**

are bonafide students of this institute and the work has been carried out by them under the supervision of **Prof. D.P. Baviskar** and it is approved for the partial fulfilment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

Prof.Mrs. D.P.Baviskar
(Guide)
(Dept of Computer Engineering)

Prof. Mrs.V.Y.Kulkarni
(Head)
(Dept of Computer Engineering)

(Dr.L.K.Kshirsagar)
(Principal)
(MAAER'S Maharashtra Institute of Technology,Pune-38)

Place: Pune
Date:

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank Dr.L.K.Kshirsagar, for providing us with the opportunity to do the project work in MIT, Pune and giving us all support and guidance which made us complete the project duly. We are extremely thankful to Dr.V.Y.Kulkarni for providing such a nice support and guidance.

We owe our deep gratitude to our project guide Prof.D.P.Baviskar, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all teaching staff of Computer Engineering Department which helped us in successfully completing our project work. Also, we would like to extend our sincere esteems to all staff in laboratory for their timely support.

SAMBHAV AGARWAL
HIMANI MALI
PRITHVIRAJ KHELKAR
S.MAHABALESHWARKAR

ABSTRACT

Automatically generating a natural language explanation of an image, a problem known as image captioning, has lately received a lot of attention in Computer Vision. The problem is interesting not only because it has important practical applications, such as helping visually impaired people see, but also because it is regarded as a impressive challenge for image understanding which is a core problem in Computer Vision. Generating a meaningful and significant natural language description of an image requires a level of image understanding that goes well beyond image classification and object detection. The problem is also appealing because it connects Computer Vision with Natural Language Processing which are two major fields in Artificial Intelligence.

Inspired by recent work in machine translation and object detection, we introduce an attention based model that automatically learns to describe the content of images. We describe how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by minimizing a variational loss function.We also show through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence using RNN Decoder.

In this report, we elaborate on image captioning concerned with, especially dense image captioning.We present the technical fundamentals of a model striving to solve such a task. Concretely, a structure of Neural Image Caption is highlighted.The concept of visual attention is used to generate the captions for images. We validate the use of attention with state-of-the art performance on benchmark dataset: MS COCO.

Keywords: image captioning, dense captioning, convolutional neural networks, recurrent neural networks ,visual attention.

Contents

List of Figures

List of Tables

List of Abbreviations

RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
KNN	K Nearest Neighbors
BRNN	Bidirectional Recurrent Neural Network
CUDA	Compute Unitified Device Architecture
GPU	Graphics Processing Unit
SVM	Support Vector Machines
UML	Unified Modelling Language

Chapter 1

INTRODUCTION

1.1 Overview

In the past few years, neural networks have fueled dramatic advances in image classification. Emboldened, researchers are looking for more challenging applications for computer vision and artificial intelligence systems. They seek not only to assign numerical labels to input data, but to describe the world in human terms. Image and video captioning is among the most popular applications in this trend toward more intelligent computing systems.

For our course project, we designing an image captioning system using recurrent neural networks (RNNs) and attention models. Image captioning is the task of generating text descriptions of images. This is a quickly-growing research area in computer vision, suggesting more intelligence of the machine than mere classification or detection.

RNNs are variants of the neural network paradigm that make predictions over sequences of inputs. For each sequence element, outputs from previous elements are used as inputs, in combination with new sequence data. This gives the networks a sort of memory which might make captions more informative and context-aware.

RNNs tend to be computationally expensive to train and evaluate, so in practice memory is limited to just a few elements. Attention models help address this problem

by selecting the most relevant elements from a larger bank of input data. Such schemes are called attention models by analogy to the biological phenomenon of focusing attention on a small fraction of the visual scene.

In this work, we develop a system which extracts features from images using a convolutional neural network (CNN), combines the features with an attention model, and generates captions with an RNN.

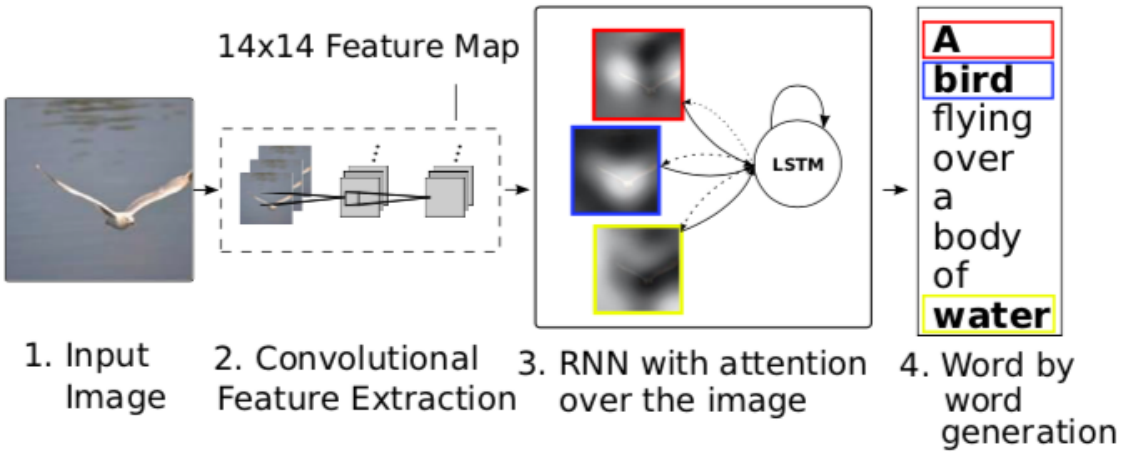


Figure 1.1: Basic Overview of Image Captioning with Visual Attention

1.2 Motivation

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. Research in this area spans and connects numerous domains, such as computer vision, natural language processing, and machine learning. A quick glance at an image is sufficient for a human to point out and describe an immense amount of details about the visual scene. Visual description is challenging because it requires recognising not only objects, but other visual elements, such as actions and attributes, and constructing a fluent sentence describing how objects, actions, and attributes are related in an image. One could say that the problem could be defined as that of a machine translation with

the source language as the pixels of the image and the target language as English.

Captioning is a potentially appealing task because in theory it requires :

(1) a detailed understanding of an image and

(2) ability to communicate that information via natural language.

However, this remarkable ability has proven to be an elusive task for our visual recognition models. The current use of solving this problem could be in fields like image search, tell stories from album uploads, help visually impaired people understand the web, etc. And many more use case are bound to present themselves as research and development continues in the field.

1.3 Problem Definition and Objectives

1.3.1 Problem Definition

The task of visual description aims to develop visual systems that generate contextual descriptions about objects in images. Given an image, break it down to extract the different objects, actions, and attributes, and finally generate a meaningful sentence(caption/description) for the image. A description must capture not only the objects contained in an image, but it also must express how these objects relate to each other as well as their attributes and the activities they are involved in. The above information must also be presented in a semantically correct format in a natural language. Hence we also need a language model in addition to the visual understanding. Thus the problem boils down to two things - image analysis to get features, and then a language model to generate meaningful captions.

1.3.2 Objectives

- To achieve a model for generating a caption for a given image
- The generated caption should accurately describe the given image surroundings

along with its subject and their interdependence.

1.4 Project Scope and Limitations

1.4.1 Project Scope

Recent breakthroughs in Image Captioning have shown that an approach using Neural networks with attention model can be used to solve the problem. The task is to generate an image caption based on an image provided by the user. The project scope for the current iteration is strictly restricted to image captioning with no attempt of video captioning.

1.4.2 Limitations

- Currently is not functional or practical for audio or video based inputs.
- Is only capable of image captioning of limited words and not longer works.
- Is not capable of generating or guessing any literary formats (Poetry, Ballad) based on input.
- Requires a high level of computational power for training
- As of now, our project is limited to Google Colaboratory.
- Only those objects will be identified which are already labelled.
- As of now, it is unable to generate captions in real-time.

Chapter 2

LITERATURE SURVEY

2.1 YOLO 9000: Better, Faster, Stronger

The YOLO model is a simple to construct method that throws away complex pipelines used by the art object classifiers and simultaneously calculates the class confidences simultaneously in all the grids in which the image is divided. It outperforms object detectors and classifiers when tested on videos in terms of speed by a great degree. It is a ot faster in processing objects in a real time video. It predicts a bounding box for each of the object present in that image and all the bounding boxes and their corresponding class probabilities are predicted by a single CNN model simultaneously. It has even proved better than R-CNN and DPM in cases where it is trained on natural images and tested on classification task for any artwork. The object detection problem is proposed as a regression problem so the need of the complex pipelines is removed. It makes less number of incorrect background classification errors compared to Fast R-CNN. Batch normalization proved as a very good substitute for dropout increasing the mean Average Precision by 2%. As compared to state of the art detection systems YOLO makes localization errors mainly because CNNs have a tendency to fail in localization. The cross-entropy function used for training and detection in small as well as large bounding boxed have a great difference of errors depending upon the size of boxes.

2.2 Translating Videos to Natural Language using Deep Recurrent Neural Networks

In this paper a model is proposed for video description which uses neural networks for the entire pipeline from pixels to sentences and can potentially allow for the training and tuning of the entire network. In an extensive experimental evaluation, we showed that the approach generates better sentences than related approaches. The mean pooling which captures the features of the FC layer of the CNN has helped increasing the performance metric significantly as compared to the state-of-the-art method that gives features from FC layer directly to the LSTM. The model trained on Flickr30k when tested on random frames from the video scored on subjects and verbs with accuracy of 75.16% and 11.65% respectively and 9.01% on objects.

2.3 Long-Term Recurrent Convolutional Network for Visual Recognition and Description

This approach uses frames of the video as inputs to a corresponding CNN with LSTM units corresponding to each which produces a sentence summary based on a strong visual time series model. It outperforms a single CNN-RNN cascade model for an image and even the multi-model neural networks model. It outperforms KNN algorithm in mapping visual features against probable sentences. The results consistently demonstrates that by learning sequential dynamics with a deep sequence model, we can improve upon previous methods which learn a deep hierarchy of parameters only in the visual domain, and on methods which take a fixed visual representation of the input and only learn the dynamics of the output sequence.

2.4 Show and Tell: A Neural Image Captioning Generator

In the paper a generative model is presented based on a deep recurrent architecture that combines Vision Deep CNN and Language Generating RNN to generate natural sentences describing an image. Their model is trained to maximize the likelihood of the target description sentence given the training image. This model is popularly known as Google NIC(Neural Image Caption) Generator. They propose a neural and probabilistic framework to generate a descriptions from images. This model make use of a recurrent neural network which encodes the variable length input into a xed dimensional vector, and uses this representation to decode it to the desired output sentence. It is a single joint model that takes an image I as input, and is trained to maximize the likelihood p(S—I) of producing a target sequence of words S =S1, S2, ... where each word St comes from a given dictionary, that describes the image adequately. The model is not tested and ready to handle the unsupervised data, both from images alone and text alone. Hence, it is not know about how to use unsupervised data to improve image description approaches.

2.5 Show, Attend and Tell: Neural Image Captioning Generation with Visual Attention

In this paper they introduce an attention based model that automatically learns to describes the content of images. They describes the methods to train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. The model is able to automatically learn to x its gaze on salient objects while generating the corresponding words in the output sequence and this is represented through visualization. By visualizing the attention component learned by the model, they were able to add an extra layer of interpretability to the output of the model. They introduce two attention-based image caption

generators under a common framework. (i) a soft deterministic attention mechanism trainable by standard backpropagation methods and (ii) a hard stochastic attention mechanism trainable by maximizing an approximate variational lower bound.

2.6 Deep Visual-Semantic Alignments for Generating Image Descriptions

In this paper, a model is presented that generates natural language descriptions of images and their regions. Their alignment model is based on a novel combination of Convolutional Neural Network(CNNs) over image regions, bidirectional Recurrent Neural Networks(BRNNs) over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. They also describe a Multimodal Recurrent Neural Network architecture that uses the inferred alignments to learn to generate novel descriptions of image regions. They detect objects in every image with a Region Convolutional Neural Network (RCNN). The CNN is pre-trained on ImageNet and finetuned on the 200 classes. The BRNN takes a sequence of N words and transforms each one into an h-dimensional vector. The model can only generate a description of one input array of pixels at a xed resolution. The RNN receives the image information only through additive bias interactions, which are known to be less expressive than more complicated multiplicative interactions. Their approach consists of two separate models. Going directly from an image-sentence dataset to region-level annotations as part of a single model trained end-to-end remains an open problem.

2.7 Caffe: Convolutional Architecture for Fast Feature Embedding

Caffe provides a clean and modiable framework and have an orderly and extendible toolkit for state-of-the-art deep learning algorithms, with a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB

bindings for training and deploying general purpose convolutional neural networks and other deep models efficiently on commodity architectures. Fast CUDA code and GPU computation achieves processing speeds of more than 40 million images per day on a single K40 or Titan GPU. Caffe provides a complete toolkit for training, testing, netuning, and deploying models, with well-documented examples for all of these tasks. Blobs (4-dimensional arrays) conceal the computational and mental overhead of mixed CPU/GPU operation by synchronizing from the CPU host to the GPU device as needed.

2.8 Microsoft COCO: Common Objects in Context

This paper focuses on advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. A new large-scale dataset is introduced that addresses three core research problems in scene understanding they are detecting non-iconic views (or non-canonical perspective) of objects, contextual reasoning between objects, the precise 2D localization of objects. Annotation pipeline is split into 3 primary tasks they are category labelling: labelling the categories present in the image, instance spotting: locating and marking all instances of the labelled categories, instance segmentation: segmenting each object instance. The system currently only label things, but labelling stuff may also provide significant contextual information that may be useful for detection.

2.9 Deep Neural Network for Object Detection

This paper explains a binary mask is created around the object. That is the pixel containing the value 1 inside the mask contains the object else it is 0. The binary mask is applied in multi-scale fashion, ie, in both vertical and horizontal fashion to identify the borders of the image. After that refinement process is applied and

repeated until the classification is as precise as possible. So the basic approach is use the full image as an input and perform localization, DNN localizer is used on small set of large windows. The comparison done with other techniques was somewhat biased, as this technique was trained on the larger VOC2012 training set while other techniques were published before that training set existed. Some mis-detections did occured due to similar looking objects or imprecise localization which was due to the ambiguous definition of object extend by the training data - in some images only the head of the bird is visible while in others the full body.

2.10 Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation

In this paper a approach is introduced that uses Bounding boxes annotations than per pixel masks. To begin with an unsupervised region proposal method is used to generate candidate segmentation masks. The convolutional network is trained under the supervision of these approximate masks. Although the masks are coarse at the beginning, they are gradually improved and then provide useful information for network training. So basically a rough mask is considered as a rectangle mask instead of going pixel by pixel using unsupervised methods (eg, Selective Search). A segmentation mask is generated using Grabeut method and it is trained against candidate segments to learn semantic features to pick better candidates. This process is iterated. After few iterations these method has been found as pixel based mask method.

2.11 Explain Images with Multimodal Recurrent Neural Networks

The architecture contains a language model part, an image part and a multimodal part. The language model par learns the dense feature embedding for each word in the dictionary and stores the semantic temporal context in recurrent layers. The image part contains a deep Convolutional Neural Network which extracts image features. The multimodal part connects the language model and the deep CNN together by a one-layer representation. m-RNN model is learned using a perplexity based cost function. The errors are back-propagated to the three parts of the m-RNN model to update the model parameters simultaneously.

2.12 From Caption to Visual Concepts and Back

In this paper, the system trains on the images and corresponding captions, and learns to extract nouns, verbs, and adjectives from regions in the image. The detected words then guide a language model to generate text that reads well and includes the detected words. After this the deep multimodal similarity model is used to re-rank candidate captions. Due to direct use of captions the caption detector trained from images with captions containing that specific word will be biased towards detecting an object corresponding to that word that are salient in the image. Training a language model (LM) on image captions captures commonsense knowledge about a scene. A multimodal joint representation for learning a caption detector is used which increases the efficiency of mapping between image features and text.

2.13 Multimodal semi-supervised learning for image classification

The goal of this paper is to learn a classifier for images alone, but they use the keywords associated with labelled and unlabelled images to improve the classifier using semi-supervised learning. They first learn a strong Multiple Kernel Learning (MKL) classifier using both the image content and keywords, and use it to score unlabelled images. They then learn classifiers on visual features only , either support vector machines (SVM) or least square regression (LSR), from the MKL output values on both the labelled and unlabelled images. They also considered learning the textual-visual classifier and the visual only classifier jointly, rather than sequentially, but it is unclear how to make the combined classifier benefit from the visual classifier.

2.14 Learning CNN-LSTM Architectures for Image Caption Generation

This paper introduces a method which uses a Deep Convolutional neural network to create a semantic representation of an image, which is then decoded using a LSTM (Long Short Term Memory) network. All LSTMs share the same parameters. The vectorized image representation is fed into the network, followed by then special start of sentence token. The hidden state produced is then used by the LSTM to predict/generate the caption for the given image. Analogous to recent successful approaches in statistical machine translation. But using an encoder recurrent neural network, these model learn an expressive representation of the original sentence. Conducted extensive hyperparameter tuning on dropout to tackle overfitting. The CNN-LSTM model learns to identify pictures in increasing detail and correct its earlier mistakes. However it is not always correct for unseen randomized validation images ,ie, the output was categorized into three sets which are namely correct, partially correct, and wrong.

2.15 Support Vector Machine classification for Object-Based Image Analysis

The SVM classification methodology was found very promising for Object-Based Image Analysis. Its overall accuracy was better than Nearest Neighbor in every aspect when their confusion matrix was compared. The results was limited to test images up to 1000X1000 pixel size and very large remote sensing datasets is the barrier to overcome. The main goal was to compare computation efficiency of SVM with Nearest neighbor Object-based Classifier results. A SVM approach for multi-class classification was followed, based on primitive image objects provided by a multi-resolution segmentation algorithm. Then a feature selection step, in order to provide the features for classification which involved spectral, textura and shape information. After that a module that integrated a SVM classifier and the segmentation algorithm was developed in C++. The SVM module is capable to use 4 types of kernel for training and classification: linear, polynomial, radial basis function and sigmoid. It has great potential for remote sensing data. It surpassed Nearest Neighbor, Maximum Likelihood Decision Tree Classifiers in robustness and accura

2.16 Automatic Image Captioning

The proposed new methods (Corr, Cos, SvdCorr, SvdCos) consistently outperform the state of the art EM (45 % relative improvement) in captioning accuracy. The improved adaptive blob-tokens generation consistently leads to performance gains. The methods are less biased to the training set and more generalized in terms of retrieval precision and recall. This paper includes proposition of four methods: Corr, Cos, ScdCorr, SvdCos to estimate a translation table, whose element can be viewed as the probability used to caption the term which is used as blob-token. The blob-tokens are generated using the K-means algorithm on feature vectors of all image regions in the image collection, with the number of blob-tokens, B, set at 500 (not optimal).

The correlation-based translation table Tcorr is defined by normalizing each column of Tcorr,0 such that each column sum up to 1. Tcorr measures the association between a term and blob-token by the co-occurrence counts or how similar the overall pattern of a term and a blob-token is.

2.17 Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models

This paper describes a new approach to the problem of image caption generation, casted into the framework of encoder-decoder models. For the encoder, they use a joint image-sentence embedding where sentences are encoded using long short-term memory (LSTM) recurrent neural networks. Image features from a deep convolutional network are projected into the embedding space of the LSTM hidden states. A pairwise ranking loss is minimized in order to learn to rank images and their descriptions. For decoding, they introduce a new neural language model called the structure-content neural language model (SC-NLM). The SC-NLM differs from existing models in that it disentangles the structure of a sentence to its content, conditioned on distributed representations produced by the encoder. The SC-NLM model uses a sequence of word-specific structure variables, the structure variables help guide the model during the generation phrase and can be thought of as a soft template to help avoid the model from generating grammatical nonsense. The result of this technique when compared with nearest neighbor was arguably good. And comparing with best results of Treetalk it could closely describe with the original captions. The model cannot align parts of captions to images and use these alignments to determine where to attend next.

Chapter 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

- The image provided for testing should not be used for the training of the model.
- Version Conflicts must be adhered to while implementing the system.
- Accurate results for hazy or blurred images would not be produced properly.

3.2 Functional Requirements

This section contains all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. This section describes major functional requirements of the system.

3.2.1 System Feature 1

The objective is to build a system that can generate a descriptive caption for a user provided image.

3.2.2 System Feature 2

The system must use traditional CNN-Encoder and RNN- Decoder with attention based model architecture for caption generation. The use of Attention networks is widespread in deep learning, and with good reason. This is a way for a model to choose only those parts of the encoding that it thinks is relevant to the task in hand.

3.2.3 System Feature 3

The MS-COCO dataset shall be used to train the model. This dataset contains more than 82,000 images, each of which has been annotated with at least 5 different captions. The code is going to download and extract the dataset automatically.

3.2.4 System Feature 4

The system should apply different preprocessing techniques for cleaning the dataset.

3.2.5 System Feature 5

Image feature extraction - The system should be able to extract useful features from an image source. For this purpose, InceptionV3(pretrained on Imagenet) can be used. InceptionV3 has 21 million parameters and its top-5 error on Imagenet is 3.46 %. Features can be extracted from the last convolutional layer.

3.2.6 System Feature 6

Finally, the predictions made should be used to calculate the loss value and gradients that should further be applied to the optimizer.

3.3 External Interface requirements

3.3.1 User Interfaces

User interface generally refers to the visual layout of the elements that a user might interact with. We have used Google Colab's essentially minimalist interface for the development of our project, with proper image and textual outputs displayed at every cell block.

3.3.2 Hardware Interfaces

Google Colab runs with single core hyper threaded Xeon Processors @2.3Ghz (i.e 1 core, 2 threads). It uses a Tesla K80 GPU having 2496 CUDA cores,and 12GB GDDR5 VRAM. It is equipped with 33GB of free SSD space.

3.3.3 Software Interfaces

Most browsers are compatible with Google Colaboratory's Jupyter Notebook interface.

3.4 Non functional Requirements

This section describes major non-functional requirements of the system.

3.4.1 Performance Requirements

- System latency should be minimum.
- System should be able to correctly detect the objects.
- System should be able to generate accurate captions in human readable form.

3.4.2 Software Quality Attributes

- **Usability** The system must be easy to learn for users. Error messages must give the user specific instructions for recovery. The help manual will explain all functions and how to achieve common tasks, though the system must feel intuitive enough so that the manual is never required.

- **Testability** The system must be tested on various types of images in terms of colors, mood and content.

  It should also be tested on different genres of the captions generated and also that the caption is appropriate

- **Efficiency** The system must efficiently output the caption immediately after the image is uploaded with the lowest response time possible.

- **Effectiveness** The system must effectively capture the mood of the image, and output an appropriate caption for the image.

- **Extensibility** The system must extend to images of all types and moods. It must also extend to images of various sizes and extensions.

- **Maintainability** The previously uploaded images must be maintained in a proper sorted directory with folders for easy access and maintenance.

- **Portability** The system could also include a mobile interface or application.

- **Reliability** The system must have minimum downtime, and be available for immediate access if needed.

3.5 System Requirements

3.5.1 Database Requirements

MS COCO

MS COCO is a large scale object detection,segmentation and captioning dataset.

The full form of COCO is Common Objects in Context. Microsoft COCO focusses on advancing the state of object recognition by placing the question of object recognition in the context of the broader question of sense understanding. It is a new large scale dataset is introduced that addresses three core research problems in scene understanding they are detecting non-iconic views (or non-canonical perspectives) of objects, contextual reasoning between objects, the precise 2D localization of objects. Annotation pipeline is split into 3 primary tasks they are category labeling: labeling the categories present in the image, Instance spotting: locating and marking all instances of the labeled categories, Instance segmentation: segmenting each object instance. The system currently only labels "things" but labeling "stuff" may also provide significant contextual information that may be useful for detection.
Some common features of of COCO are:

- Object Segmentation
- Recognition in Context
- Superpixel stuff segmentation
- 330K images (¿200K labelled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

3.5.2 Software Requirements

1. **Platform**: Google Collab
2. **Operating System**: Windows/Linux/MacOSX

3. **Programming Language**: Python

4. **Dependency Packages**: Numpy, SciKitLearn, Pandas, Keras, Tensorflow, Matplotlib, os,re, time, json, glob, PIL, pickle

3.5.3 Hardware Requirements

1. Pentium III Processor

2. 4GB Graphic Card

3. 8GB Ram

3.6 Analysis Models: SDLC Model to be Applied

3.6.1 Iterative and Incremental Model

It is developed to overcome to weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles(iterative) and in smaller portions at a time(incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system. It can consist of mini waterfalls.

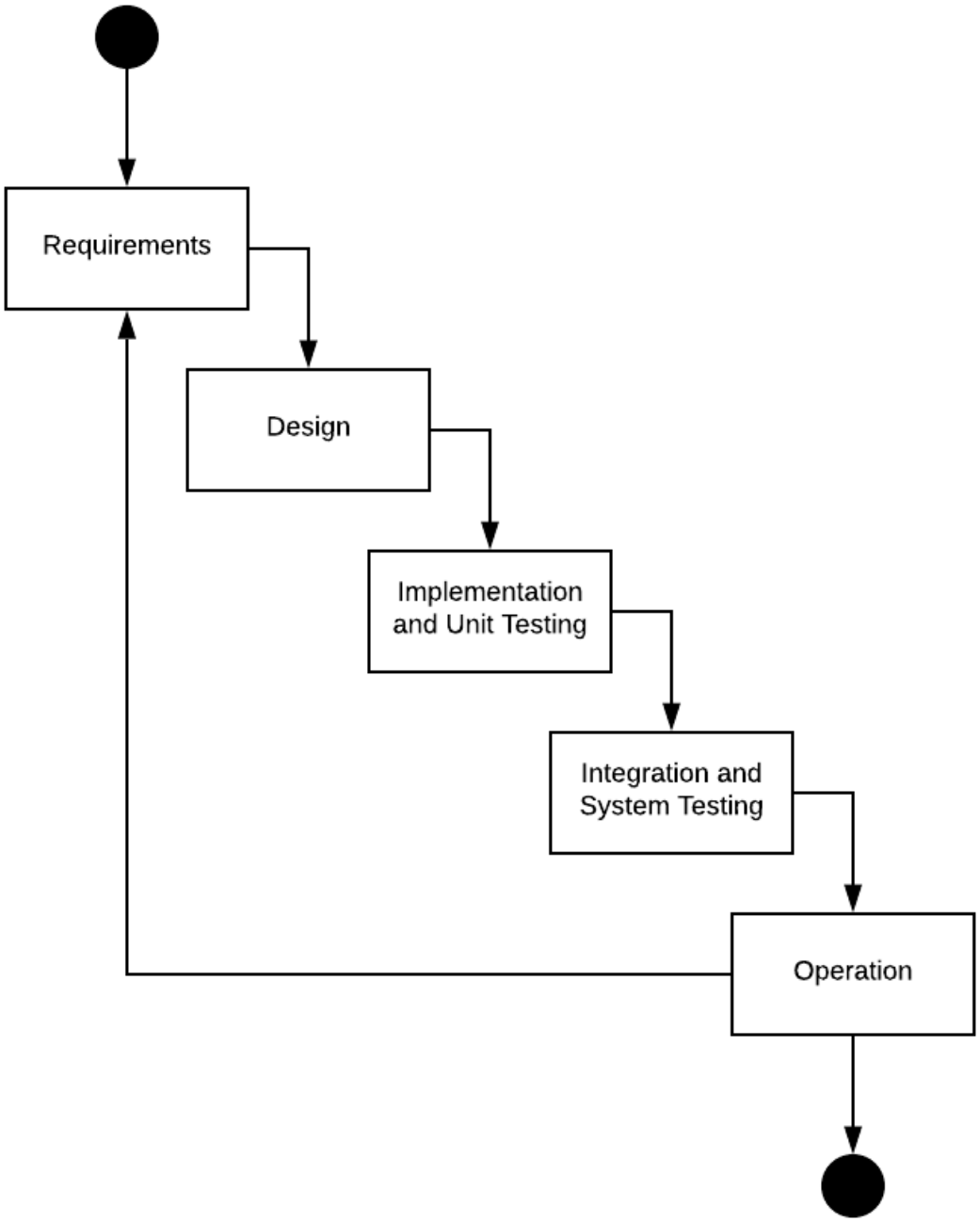


Figure 3.1: Iterative and Incremental Model.

In our project we have followed the Iterative and Incremental Model. The strategy behind using this model is that it calls for flexibility that is various parts of the project can be built in stages and iteratively.

We can repeat the process(waterfall) cycle as and when required. The basic struc-

ture of the Waterfall Model is as follows:

Figure 3.2: Waterfall Model

We first started off with the requirements analysis. At this stage we found out the requirements for the system such as the output required, the dataset to be used, the training model etc.

Then System Designing took wherein we devised the Technologies to be used during the implementation and a detailed layout for the model.

In the implementation phase, the actual coding part was done in Google Colaboratory.

And then the testing of the implemented part was done wherein the caption generated by the system were compared to those produced by the humans for the same test image.

Furthermore another requirement of adding a Graphical User Interface was observed and was taken into account.

Then the task of selecting the platform for UI design was chosen and the layout for the front end was designed.

In the implementation phase the front end was implemented and then incorporated with the Image Captioning code.

Then the entire system was tested for smooth performance.

Chapter 4

SYSTEM DESIGN

4.1 System Architecture

Our model will has three main parts: a convolutional neural network (CNN) that extracts features from images, an attention mechanism that weighs the image features, and an RNN that generates captions to describe the weighted image features. In the following sections, we will describe each in turn.

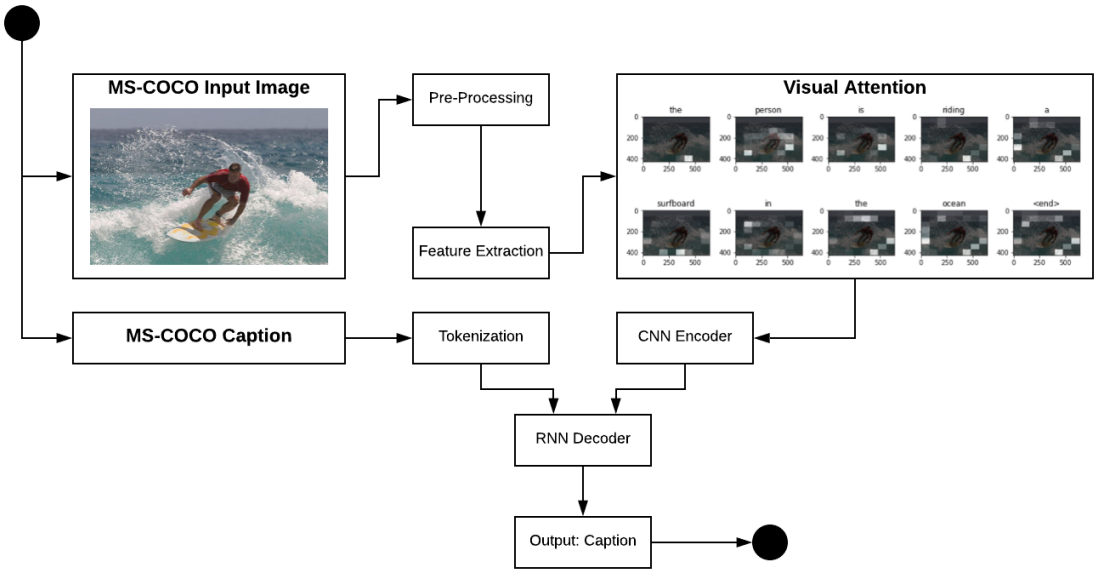


Figure 4.1: System Architecture

4.1.1 Convolutional Neural Network

A convolutional neural network is a standard feed-forward neural network, except that instead of computing the ane function f(x) = Ax + b, we restrict A to be a Toeplitz matrix. This is the same as saying that A is a convolution. This basic idea generalizes to images, where we have a 2D convolution over each of the K color channels. Each layer performs an affine operator consisting of convolutions and element-wise summations.

Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

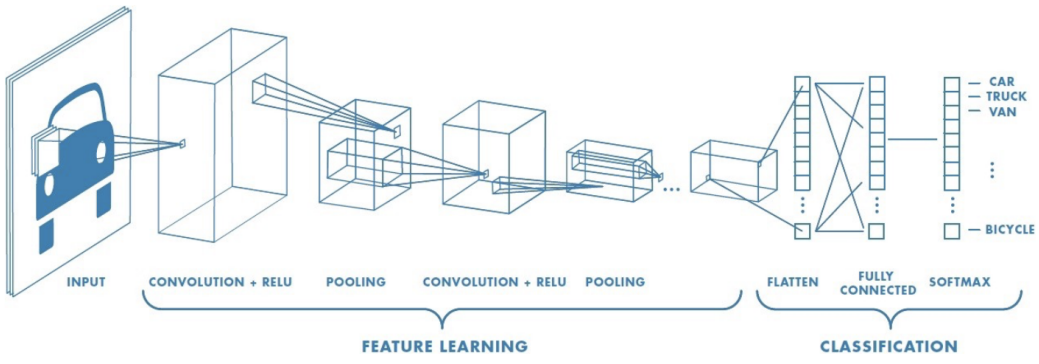


Figure 4.2: Working of a Convolutional Neural Network

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic

values between 0 and 1.

4.1.2 Recurrent Neural Network

While CNNs are adept at signal processing, they cannot easily express the idea of patterns in sequences.

Recurrent neural networks (RNNs) are similar to feed-forward neural networks, except their outputs are fed back into the input. The networks maintain a hidden state that allows them to adjust their behavior throughout iterations of this feedback loop. RNNs are considered state of the art in machine translation and other tasks involving generating text, as they easily learn the patterns of human grammar.

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and are special because they are the only ones with an internal memory.

Because of their internal memory, RNNs are able to remember important things about the input they received, which enables them to be very precise in predicting whats coming next.

This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.

RNNs and Feed-Forward Neural Networks are both named after the way they channel information. In a Feed-Forward neural network, the information only moves in one direction, from the input layer, through the hidden layers, to the output layer. The information moves straight through the network. Because of that, the information never touches a node twice. Feed-Forward Neural Networks, have no memory of the input they received previously and are therefore bad in predicting whats coming next. Because a feed forward network only considers the current input, it has no notion of order in time. They simply cant remember anything about what happened in the past, except their training.

In a RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously.

Therefore a Recurrent Neural Network has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about what is coming next, which is why a RNN can do things other algorithms cant.

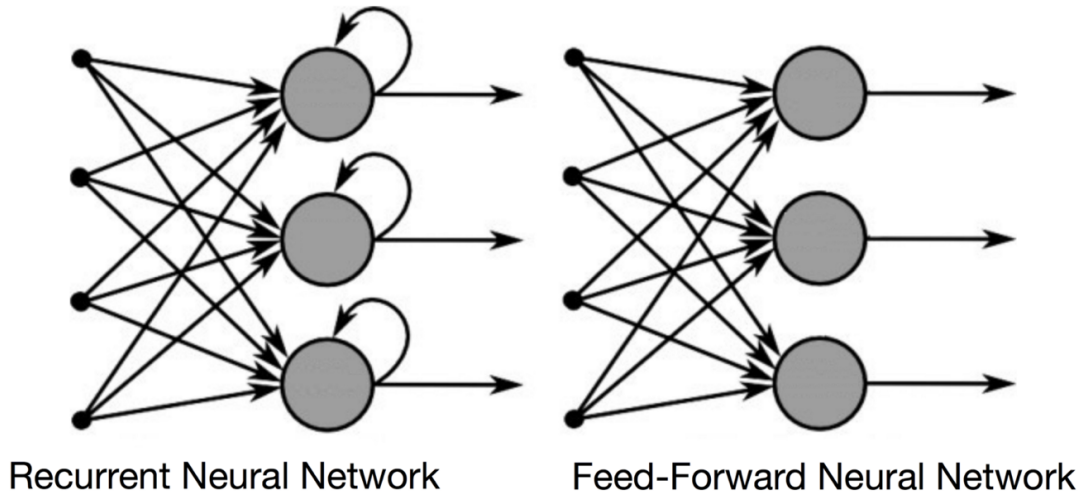


Figure 4.3: Comparing Feed Forward Neural Network with RNN

4.1.3 Inception V3 Model

Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher". The Inception deep convolutional architecture was introduced as GoogLeNet in (Szegedy et al. 2015a), and was named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Ioffe and Szegedy 2015) and was named Inception-v2. Later by additional factorization ideas in the third iteration (Szegedy et al. 2015b) called as Inception V3,the model was made more accurate and powerful

Factorizing Convolutions

The aim of factorizing Convolutions is to reduce the number of connections/parameters without decreasing the network efficiency.

- **Factorization Into Smaller Convolutions :** 3x3 convolutions replaces one 5x5 convolution as follows:

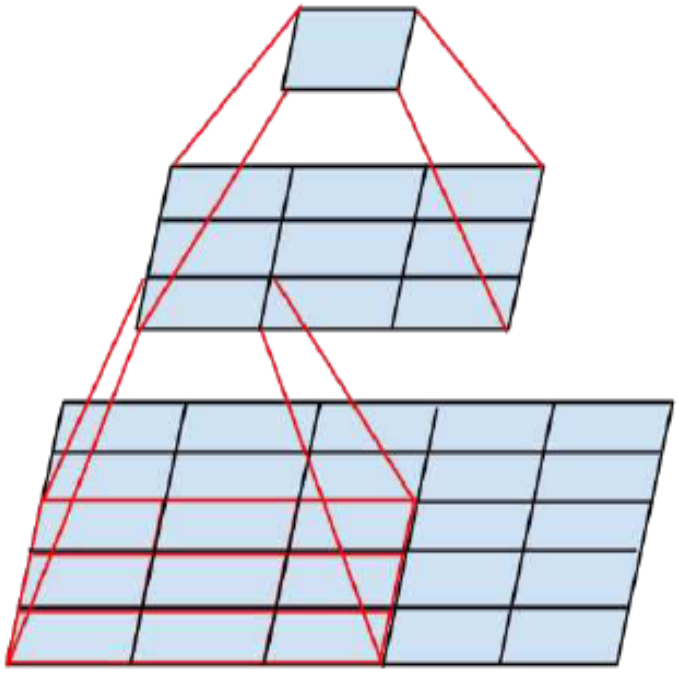


Figure 4.4: Two 3x3 convolutions replacing one 5x5 convolution

By using 1 layer of 5x5 filter, number of parameters = 5x5=25

By using 2 layers of 3x3 filters, number of parameters = 3x3+3x3=18

Number of parameters is reduced by 28%.

With this technique, one of the new Inception modules (We call it Inception Module A here) becomes:

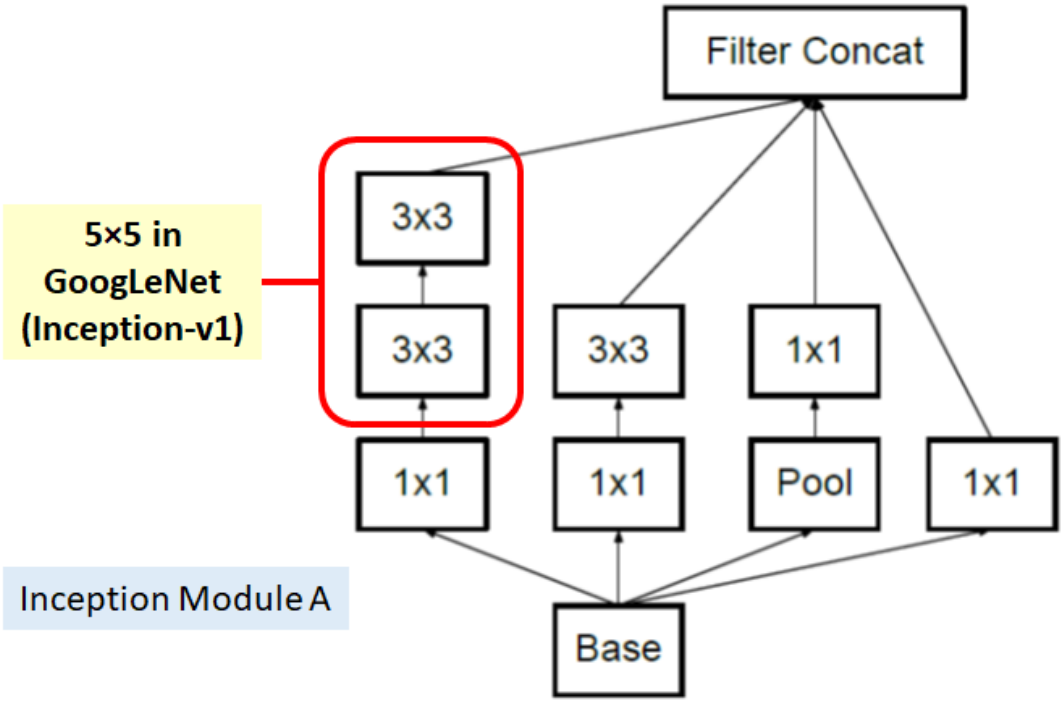


Figure 4.5: Inception model A using Factorization

- **Factorization Into Asymmetric Convolutions** One 3x1 convolution followed by one 1x3 convolution replaces one 3x3 convolution as follows:

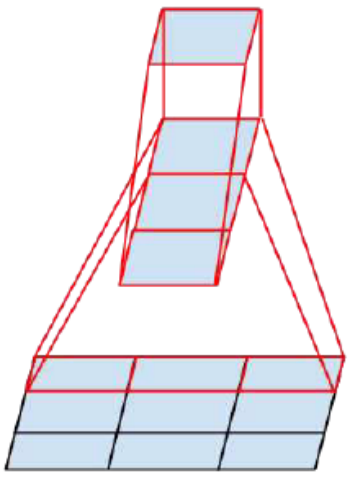


Figure 4.6: One 3x1 convolution followed by one 1x3 convolution replacing one 3x3 convolution

By using 3x3 filter, number of parameters = 3x3=9

By using 3x1 and 1x3 filters, number of parameters = 3x1+1x3=6

Number of parameters is reduced by 33%

You may ask why we dont use two 2x2 filters to replace one 3x3 filter?

If we use two 2x2 filters, number of parameters = 2x2x2=8 Number of parameters is only reduced by 11%

With this technique, one of the new Inception modules (We call it Inception Module B here) becomes:

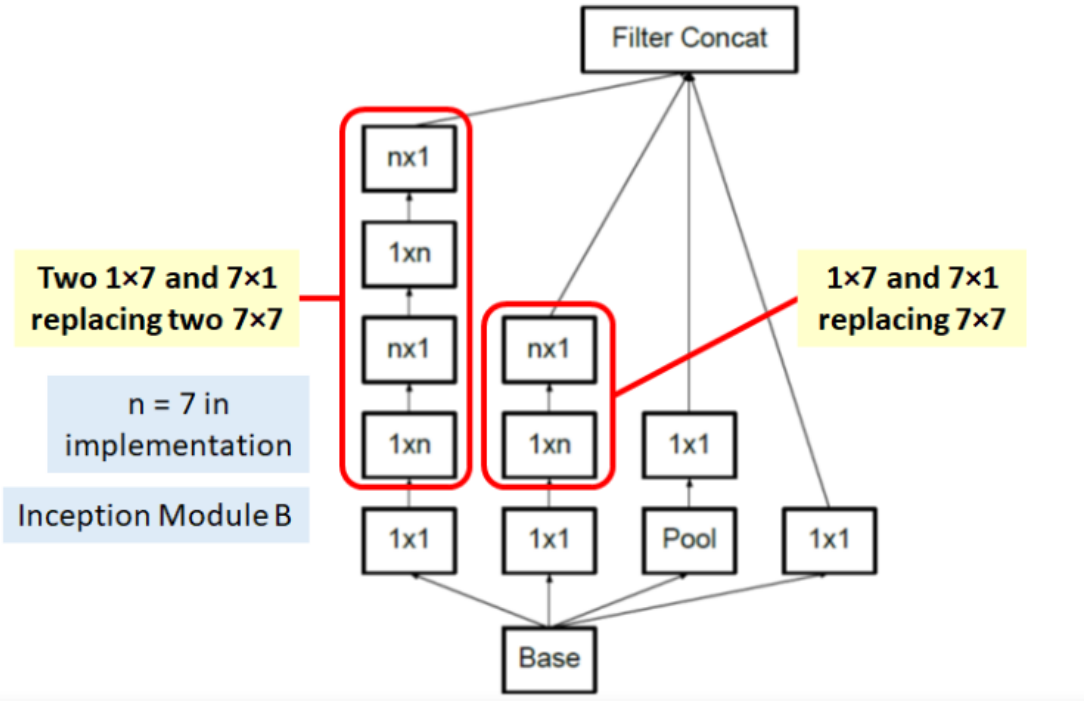


Figure 4.7: Inception Module B using asymmetric factorization

Auxiliary Classifier

Auxiliary Classifiers were already suggested in GoogLeNet / Inception-v1 . There are some modifications in Inception-v3.

Only 1 auxiliary classifier is used on the top of the last 17x17 layer, instead of using 2 auxiliary classifiers.

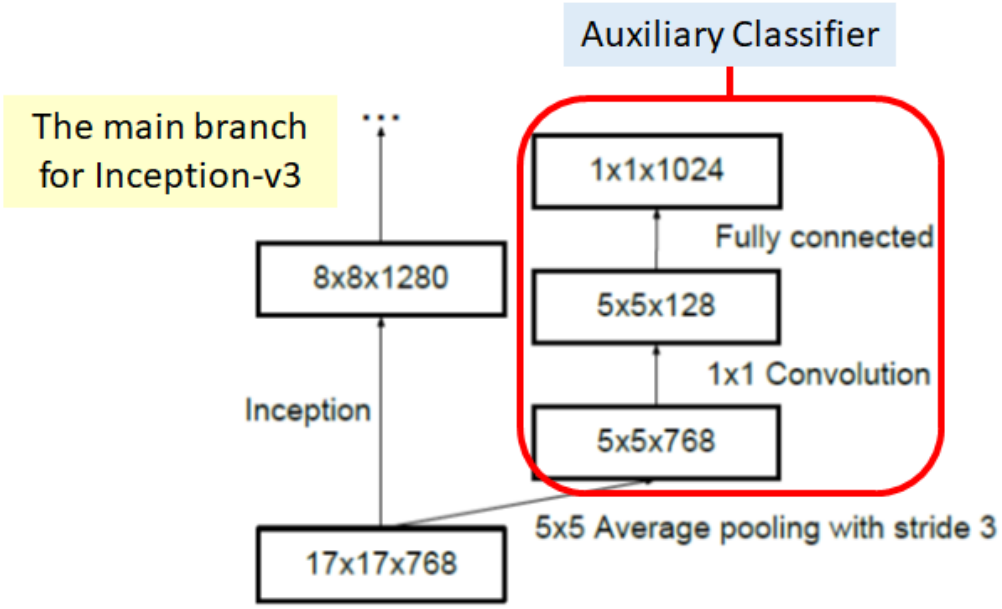


Figure 4.8: Auxiliary Classifier act as a regularization

The purpose is also different. In GoogLeNet / Inception-v1 , auxiliary classifiers are used for having deeper network. In Inception-v3, auxiliary classifier is used as regularizer. So, actually, in deep learning, the modules are still quite intuitive.

Batch normalization, suggested in Inception-v2, is also used in the auxiliary classifier.

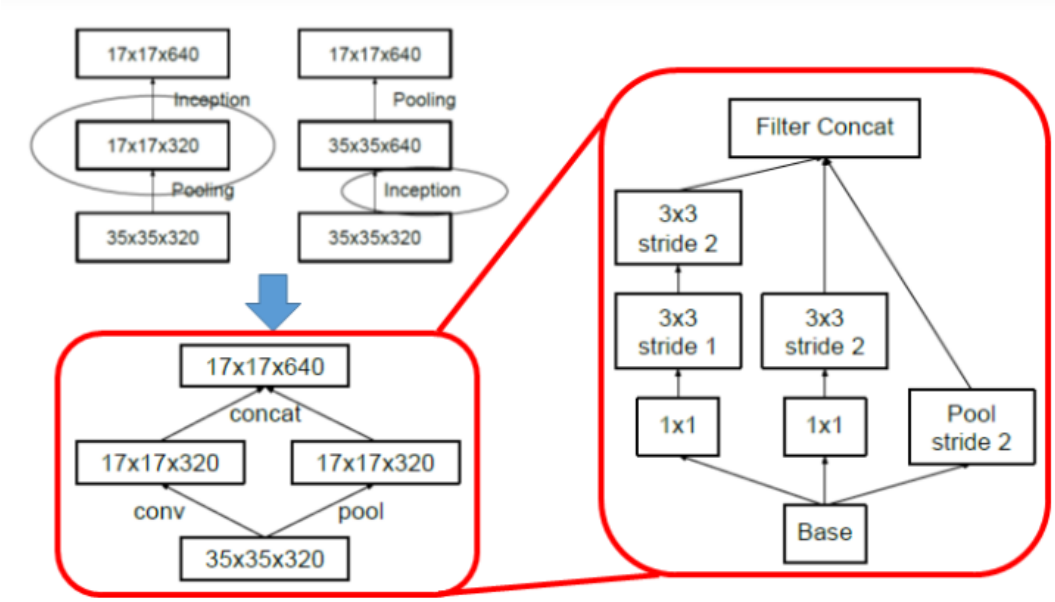


Figure 4.9: Conventional downsizing (Top Left), Efficient Grid Size Reduction (Bottom Left), Detailed Architecture of Efficient Grid Size Reduction (Right)

With the efficient grid size reduction, 320 feature maps are done by conv with

stride 2. 320 feature maps are obtained by max pooling. And these 2 sets of feature maps are concatenated as 640 feature maps and go to the next level of inception module.

Less expensive and still efficient network is achieved by this efficient grid size reduction.

Inception-v3 Architecture

With 42 layers deep, the computation cost is only about 2.5 higher than that of GoogLeNet , and much more efficient than that of VGGNet .

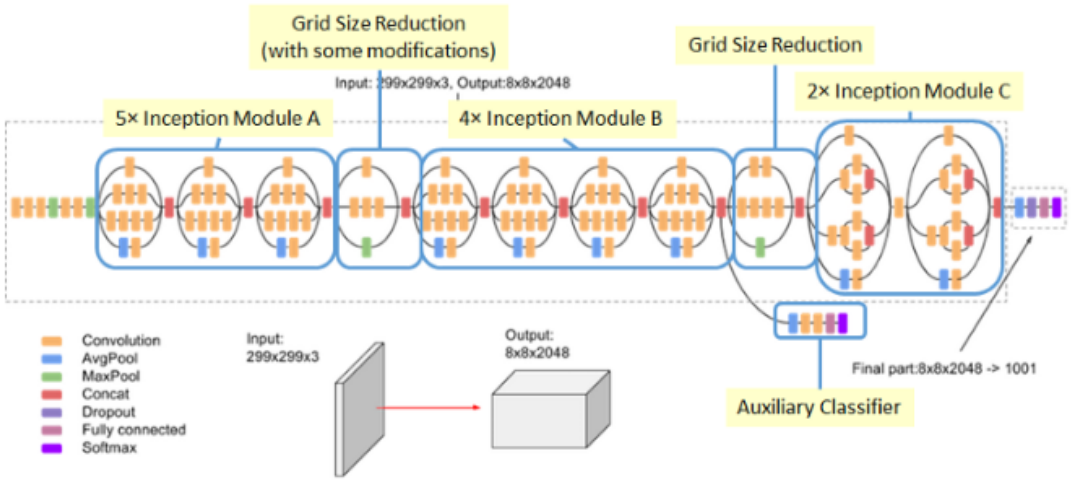


Figure 4.10: Inception-v3 Architecture (Batch Norm and ReLU are used after Conv)

4.1.4 Attention Model

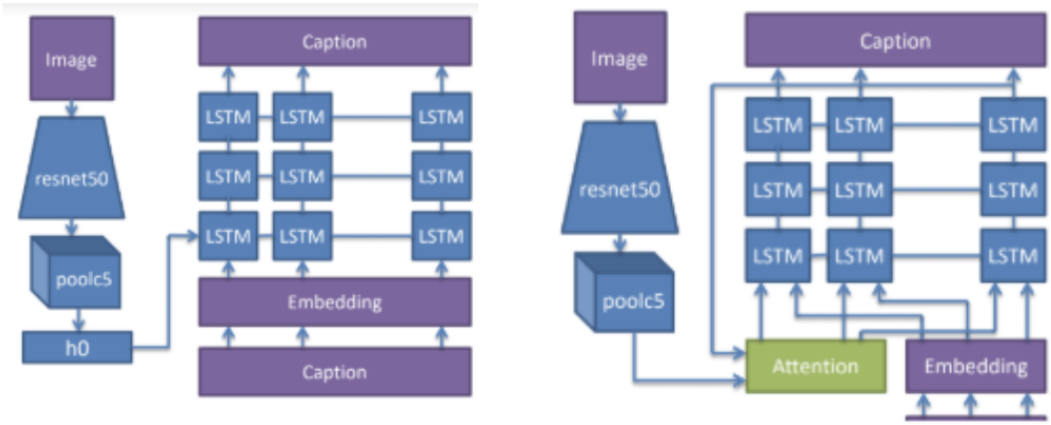


Figure 4.11: Model With and Without Attention.

Usually in Image Captioning, the spatial dimensions of the CNN image features were averaged together. Now, we describe a method to weight these spatial locations according to their perceived importance. This is known as an attention mechanism. Here, we'll use an attention-based model. This enables us to see which parts of the image the model focuses on as it generates a caption.

Efficient Grid Size Reduction

Conventionally, such as AlexNet and VGGNet, the feature map downsizing is done by max pooling. But the drawback is either too greedy by max pooling followed by conv layer, or too expensive by conv layer followed by max pooling. Here, an efficient grid size reduction is proposed as follows:

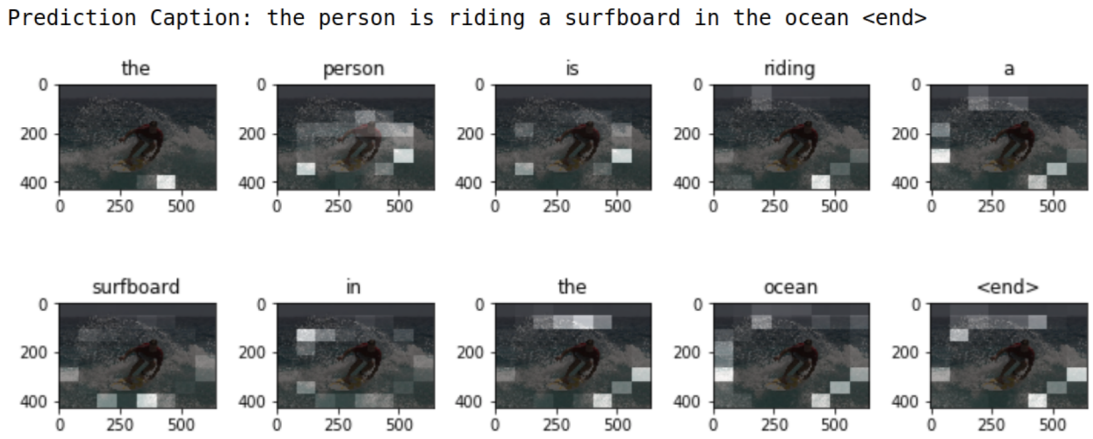


Figure 4.12: Working of Attention Model

We accomplished this by adding an attention gate to the LSTM architecture.

The attention parameters define a linear transformation of the input image. Furthermore, the softmax function ensures that for all elements we have attention parameters ¿ 0 and for the whole vector we have the absolute sum of the parameters = 1. In other words, the attention weights are positive and sum to one. The new architecture with attention is shown in figure 2.

4.2 Mathematical Model

System $S = T, O, F, Sc, Fc$

Input $I = \{i | i \in (I1)\}$

I1 = Give an image input to the model.

Output $O = \{o | o(O1)\}$

O1 = Result with the input image and the generated caption.

Function $F = \{f | f \in (F1, F2, F3, F4, F5, F6)\}$

F1 = $gru()$

F2 = $BahdanauAttention()$

F3 = $CNN\_Encoder()$

F4 = $RNN\_Decoder()$

F5 = $plot\_attention()$

F6 = $evaluate()$

Success Conditions:

1) Captions generated successfully foe the given image.

2) The generated captions correctly describe the image.

Failure Conditions:

1) Captions were not generated for the image.

2) Generated caption incorrectly describes the image.

4.3 Data Flow Diagrams

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

- **DFD Level 0:** DFD Level 0 is also called a Context Diagram.Its a basic overview of the whole system or process being analyzed or modeled.Its designed to be an at-a-glance view,showing the system as a single high-level process,with its relationship to external entities.

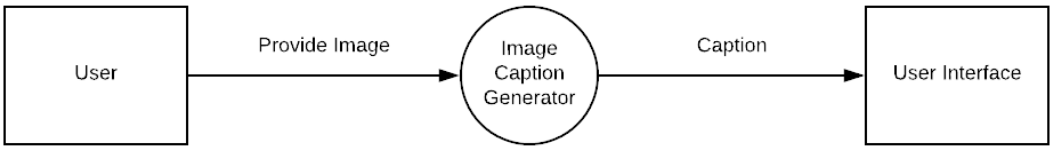


Figure 4.13: DFD Level 0

- **DFD Level 1:** DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram.It highlights the main functions carried out by the system,as you break down the high level process of the Context Diagram into its subprocesses.

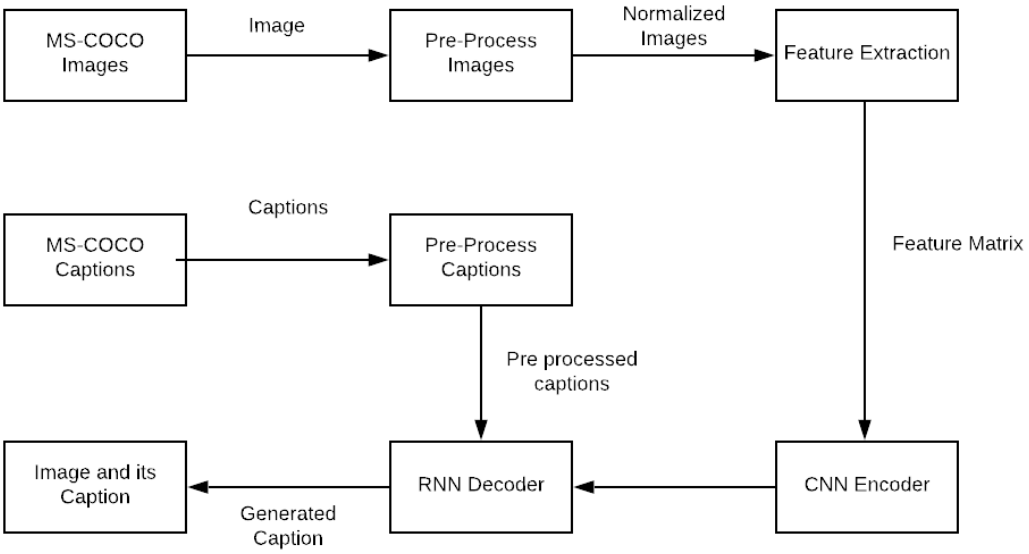


Figure 4.14: DFD Level 1

4.4 Entity Relationship Diagrams

4.5 UML Diagrams

4.5.1 Use Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. Use cases specify the expected behaviour

(what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (such as UML). A key concept of use case modeling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

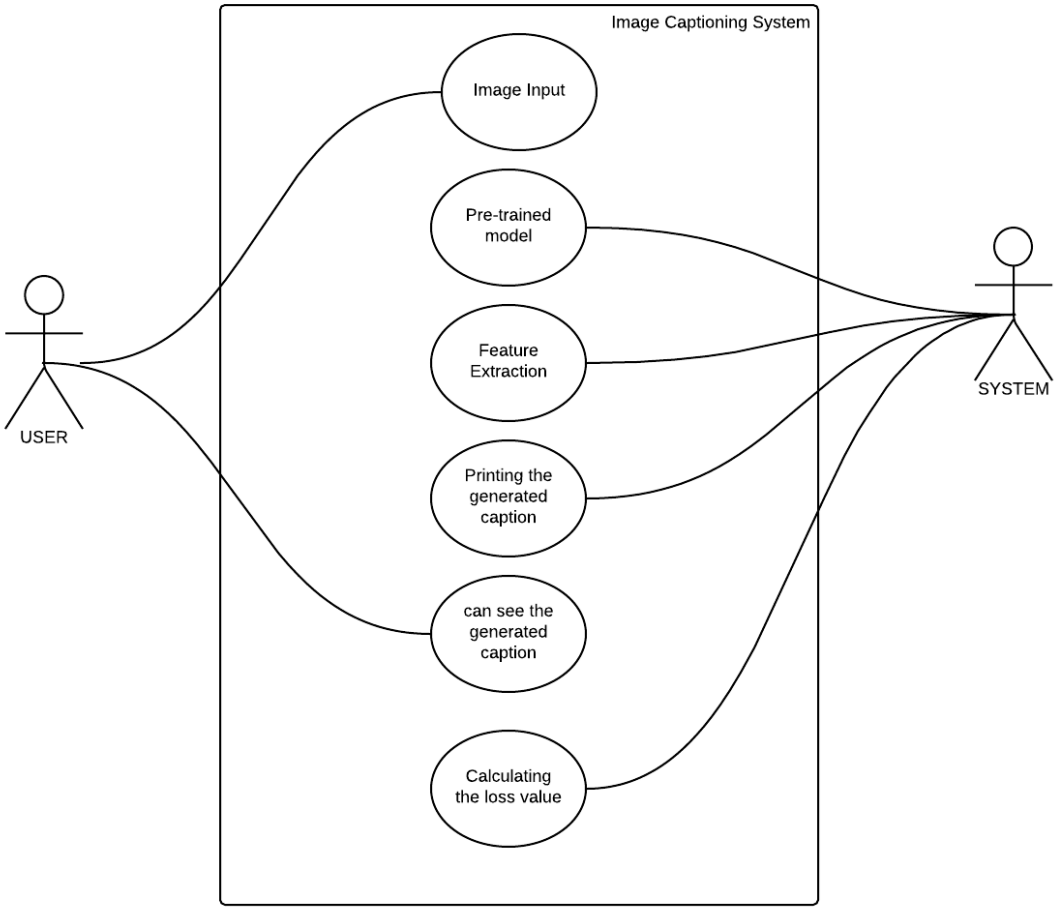


Figure 4.15: Use Case Diagram

4.5.2 Activity Diagram

Activity Diagram is basically a flowchart to represent the flow from one one activity to another activity. The activity can be described as an operation to the system. It captures the dynamic behaviour of the system.

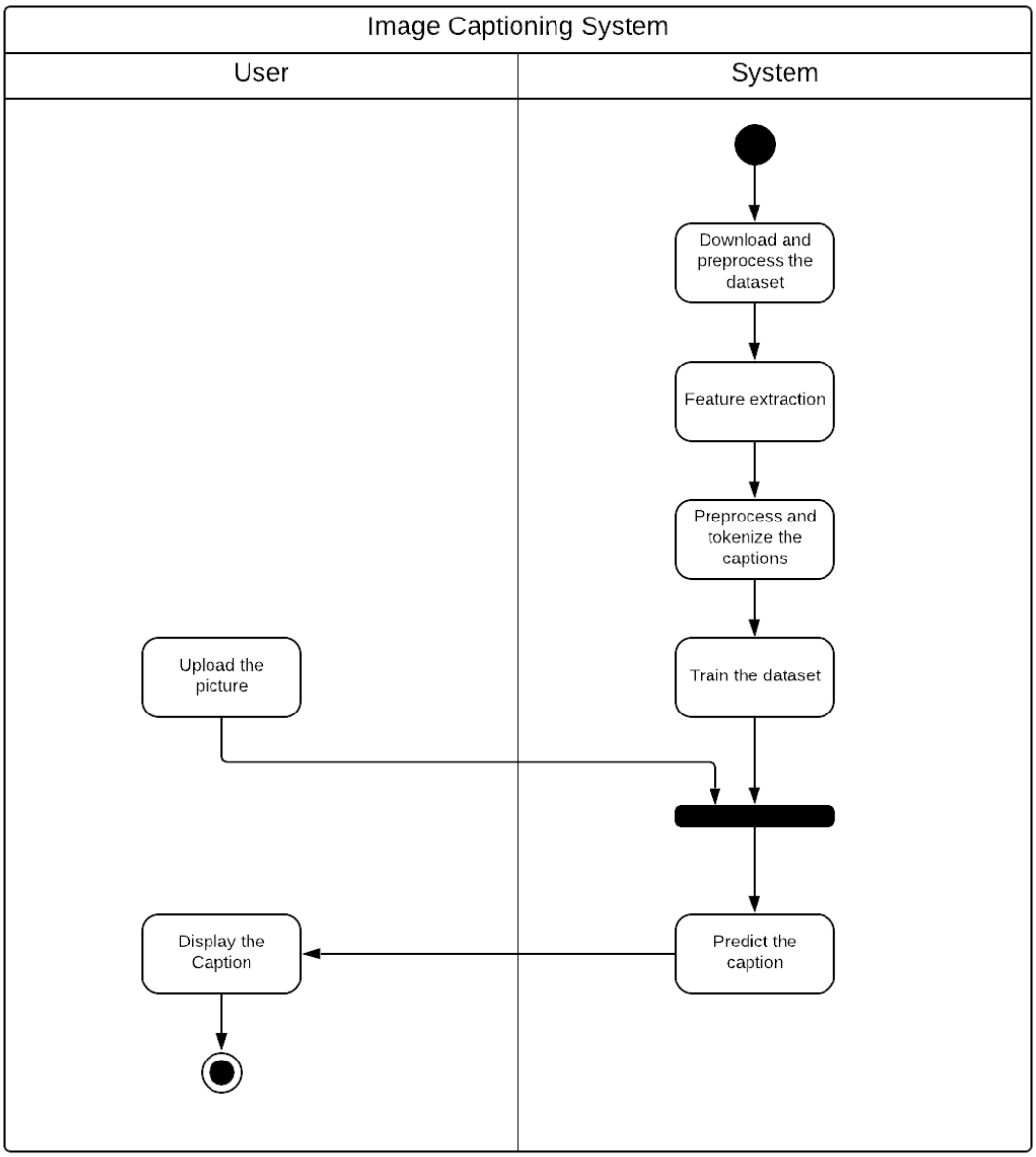


Figure 4.16: Activity Diagram

4.5.3 State Diagram

State Diagram require that the system described ins composed of finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist which differ slightly and have different semantics. State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states. Hereby each diagram usually represents objects

of a single class and track the different states of its objects through the system.

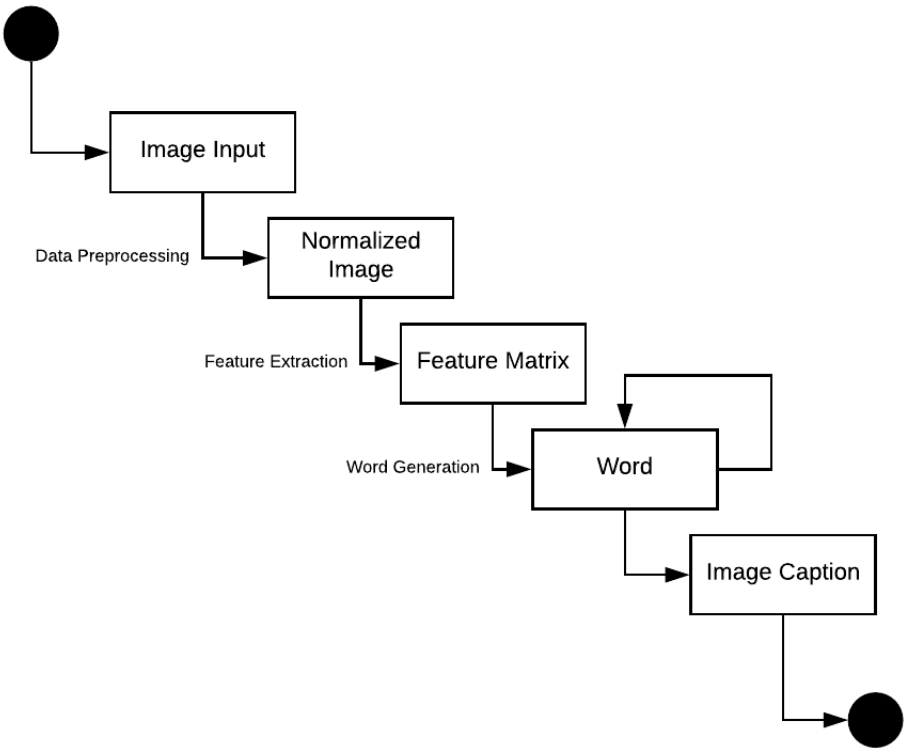


Figure 4.17: State Diagram

Chapter 5

PROJECT PLAN

5.1 Project Estimates

5.1.1 Reconciled Expenses

- **Cost Estimates:**

  The estimated cost of our project is Rs.6000. This includes Project Report, paper publication, black cover binding with golden embossing.

- **Time Estimates:**

  The estimated time for our project is 327 days ie. from 27/06/2018 to 20/05/2019.

5.1.2 Project Resources

- **People:**

  Project Guide: Prof. D.P Baviskar

  Group Members:

  Sambhav Agarwal

  Prithviraj Khelkar

  Soham Mahabaleshwarkar

  Himani Mali

- **Hardware Resources:**

    1. Pentium III Processor
    2. 4GB Graphic Card
    3. 8GB Ram

- **Software Resources:**

    1. **Platform**: Google Collab
    2. **Operating System**: Windows
    3. **Programming Language**: Python
    4. **Dependency Packages**: Numpy, SciKitLearn, Pandas, Keras, Tensorflow, Matplotlib, os,re, time, json, glob, PIL, pickle

5.2 Risk Management W.R.T. NP Hard Analysis

Risk management is the identification, assessment and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities. The process of risk management is an on-going iterative process. The mains tages of risk management includes- identify, analyze and prioritize, paln and schedule, track and report, control and learn. Each risk goes through all these steps atleast once and often cycles through numerous times.

5.2.1 Risk Identification

This is the first phase in the risk management process. It allows us to identification is the process of determining risks that could potentially prevent the program, enterprise, or investment from achieving its objectives. It includes documenting and

communicating the concern. Following are the risks in our project:

1. Machine Failure Risk
2. Inconsistent dataset Risk
3. Inaccurate Model risk
4. Network failure risk

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality. It transforms the estimates or data about specific risks that are developed during risk identification into a consistent form that can be used to make decisions around prioritization.

Table 5.1: List of Risks to the system

ID	Risk Description	Impact Schedule	Impact Quality	Impact Overall
1	Machine Failure Risk	High	High	Medium
2	Inconsistent Dataset Risk	Medium	Medium	Medium
3	Inaccurate Model Risk	High	High	High
4	Network Failure Risk	Medium	Medium	Medium

5.2.3 Overview of Risk Mitigation, Monitoring and Management

Table 5.2: Risk ID 1

ID	1
Risk Description	Machine Failure Risk
Category	Requirements
Source	Data Gathering Phase
Probability	Medium
Impact	High
Response	High
Risk Status	Occurred

Table 5.3: Risk ID 2

ID	2
Risk Description	Machine Failure Risk
Category	Requirements
Source	Data Gathering Phase
Probability	Medium
Impact	Medium
Response	Medium
Risk Status	Occurred

Table 5.4: Risk ID 3

ID	3
Risk Description	Inaccurate Model Risk
Category	Algorithm Implementation
Source	Development and Testing Phase
Probability	Medium
Impact	High
Response	High
Risk Status	Identified

Table 5.5: Risk ID 4

ID	4
Risk Description	Network Failure Risk
Category	Resource Risk
Source	Identified during implementation of the application
Probability	Medium
Impact	Medium
Response	Medium
Risk Status	Occurred

Table 5.6: Probability Reference table

Probability	Value	Description
High	Probability of occurrence is:	>75%
Medium	Probability of occurrence is:	26% - 75%
Low	Probability of occurrence is:	<25%

5.3 Project Schedule

5.3.1 Project Task Set

Major Tasks in the project stages are:

- Research on Datasets
- Analyze various Machine Learning models
- Designing the prototype
- Optimizing the model
- Improving the model
- Testing

5.3.2 Task Network

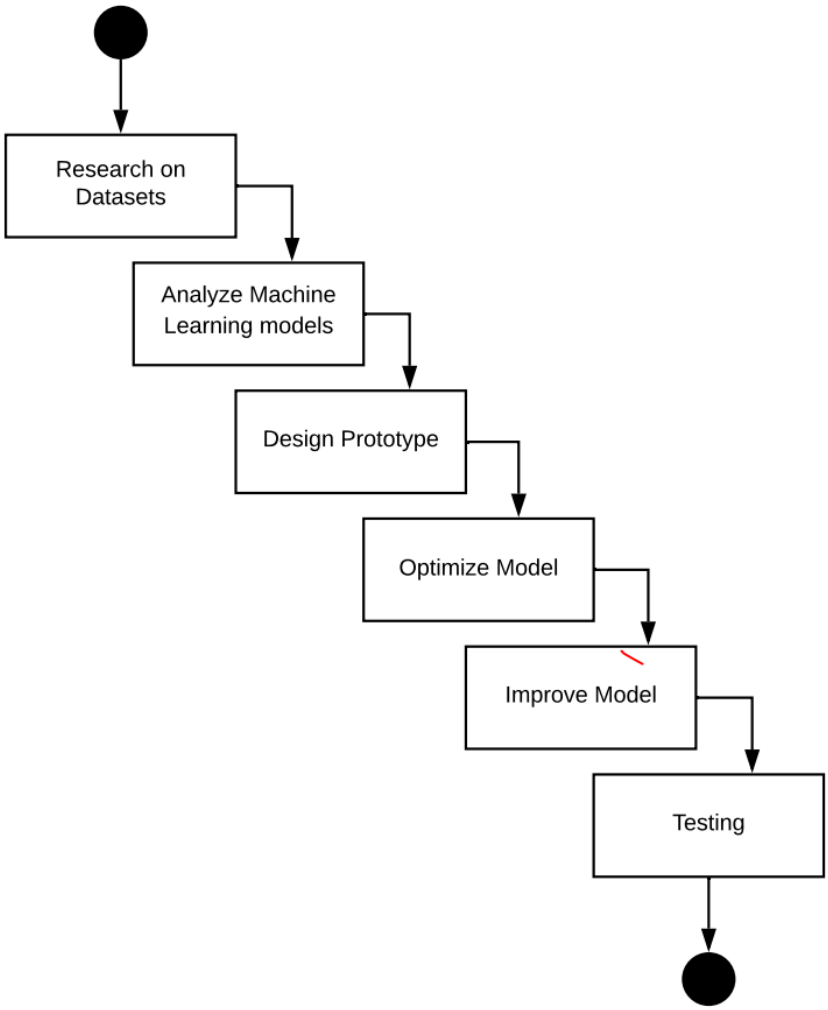


Figure 5.1: Task Network Diagram

5.3.3 Timeline Chart

Table 5.7: Timeline Chart

Sr. No	Time Span	Activity
1	20-06-18 to 30-06-18	Searching different project domains
2	04-07-18 to 14-07-18	Finalizing domain
3	15-07-18 to 21-07-18	Identifying the problem
4	22-07-18 to 26-07-18	Framing problem definition
5	27-07-18 to 20-08-18	Literature Survey
6	27-08-18 to 15-09-18	UML and Mathematical Modelling
7	16-09-18 to 26-09-18	Formulation and submission of Synopsis
8	25-10-18 to 05-10-18	Research on different datasets
9	10-12-18 to 20-12-18	Research on different Machine Learning models
10	21-12-18 to 29-12-18	Analysing and finalising the design
11	02-01-19 to 10-01-19	Designing the prototype
12	11-01-19 to 20-01-19	Designing the whole schema
13	21-01-19 to 05-02-19	Coding sub-modules
14	06-02-19 to 11-02-19	Integrating different modules
15	12-02-18 to 20-02-19	Analysing and fixing issues
16	21-02-19 to 02-03-19	Testing modules
17	12-03-19 to 20-03-19	Finalizing the system
18	22-03-19 to 30-03-19	Final documentation

5.4 Team Organization

Paper project organization is one of the key constraints to project success. There are four members in our project team.

5.4.1 Team Structure

The project team consists of four members, each of them carrying out their specific roles and responsibilities assigned in the beginning to contribute their bit to the project as a whole. Our internal guide has also invested a lot of time and efforts for the successful completion of the project.

5.4.2 Management Reporting and Communication

Progress has to be reported to the guide in a timely basis. Note for the same has to made in the workbook. Inter-team communication is done using various collaborative mechanisms like Google Docs, ShareLatex. Changes are communicated accordingly and modifications are done respectively.

Chapter 6

PROJECT IMPLEMENTATION

6.1 Overview of Project Model

The Image Captioning System is made by 5 modules. These modules are :

- **Download and Prepare the Dataset**
- **Preprocessing the MS-COCO Images**
- **Preprocessing the MS-COCO Captions**
- **Training the Model**
- **Testing the Model**

These modules are built using python programming language. After training and testing of the model,the best model is used for generating captions for the images unseen by the system.

6.2 Tools and Technologies Used

- **Platform:** Google Collaboratory
- **Operating System:** Windows or MacOSX
- **Programming Language:** Python

- **Dependency Packages:** Numpy, SciKitLearn, Pandas, Keras, Tensorflow, Matplotlib, os,re, time, json, glob, PIL, pickle

6.3 Algorithm Details

- **Download and Prepare the Dataset**

  We will use the [MS-COCO dataset](http://cocodataset.org/home) to train our model. This dataset contains ¿82,000 images, each of which has been annotated with at least 5 different captions. The code will download and extract the dataset automatically.

  We'll select a subset of 30,000 captions and use these and the corresponding images to train our model. As always, captioning quality will improve if you choose to use more data.

- **Pre-processing the MS-COCO Dataset**

  Next, we will use InceptionV3 (pretrained on Imagenet) to classify each image. We will extract features from the last convolutional layer.

  First, we will need to convert the images into the format inceptionV3 expects by:

  Resizing the image to (299, 299) Using the preprocess_input method to place the pixels in the range of -1 to 1 (to match the format of the images used to train InceptionV3).

  Initialize InceptionV3 and load the pretrained Imagenet weights

  To do so, we'll create a tf.keras model where the output layer is the last convolutional layer in the InceptionV3 architecture.

  * Each image is forwarded through the network and the vector that we get at the end is stored in a dictionary (image_name –¿ feature_vector).

  * We use the last convolutional layer because we are using attention in this

example. The shape of the output of this layer is “‘8x8x2048“‘.

* We avoid doing this during training so it does not become a bottleneck.

* After all the images are passed through the network, we pickle the dictionary and save it to disk.

We will pre-process each image with InceptionV3 and cache the output to disk. Caching the output in RAM would be faster but memory intensive, requiring 8 * 8 * 2048 floats per image. At the time of writing, this would exceed the memory limitations of Colab (although these may change, an instance appears to have about 12GB of memory currently).

Performance could be improved with a more sophisticated caching strategy (e.g., by sharding the images to reduce random access disk I/O) at the cost of more code.

- **Pre-processing the MS-COCO Captions**

* First, we'll tokenize the captions (e.g., by splitting on spaces). This will give us a vocabulary of all the unique words in the data (e.g., "surfing", "football", etc).

* Next, we'll limit the vocabulary size to the top 5,000 words to save memory. We'll replace all other words with the token "UNK" (for unknown).

* Finally, we create a word –¿ index mapping and vice-versa.

* We will then pad all sequences to the be same length as the longest one.

- **Training the Model**

The Model architecture is inspired by the [Show, Attend and Tell] (https://arxiv.org/pdf/1502.03044.pdf) paper.

* In this example, we extract the features from the lower convolutional layer of InceptionV3 giving us a vector of shape (8, 8, 2048).

* We squash that to a shape of (64, 2048).

* This vector is then passed through the CNN Encoder(which consists of a single Fully connected layer).

* The RNN(here GRU) attends over the image to predict the next word.

Training

* We extract the features stored in the respective '.npy' files and then pass those features through the encoder.

* The encoder output, hidden state(initialized to 0) and the decoder input (which is the start token) is passed to the decoder.

* The decoder returns the predictions and the decoder hidden state.

* The decoder hidden state is then passed back into the model and the predictions are used to calculate the loss.

* Use teacher forcing to decide the next input to the decoder.

* Teacher forcing is the technique where the target word is passed as the next input to the decoder.

* The final step is to calculate the gradients and apply it to the optimizer and backpropagate.

- **Testing the Model**

  * The evaluate function is similar to the training loop, except we don't use teacher forcing here. The input to the decoder at each time step is its previous predictions along with the hidden state and the encoder output.

  * Stop predicting when the model predicts the end token.

  * And store the attention weights for every time step.

Chapter 7

SOFTWARE TESTING

7.1 Type of Testing

7.1.1 Unit Testing

It is a level of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software performs as designed. Models used are:

- Image Feature Extractor
- Language Generation Model
- Model Trainer
- Model Testing

7.1.2 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group.Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates,applies tests denied in an integration test plan to those aggregates,and delivers its output as an integrated system ready for system testing. Modules used are:

- Image Captioning Model

7.1.3 Manual Testing

Manual testing is the process of manually testing software for defects.It requires a tester to play the role of an end user whereby they utilize most of the application features to ensure correct behaviour.To guarantee completeness of testing, the tester often follows a written test plan that leads them to a set of important test cases.

7.2 Test Cases and Results

7.2.1 Test ID 1:Image Feature Extractor

Table 7.1: Image Feature Extractor

Sr.No	Test Case Objective	Input Data	Expected Results	Actual Result	Status
1	Check if correct object is identified	Image with single object	Correct Object is identified	Object is identified correctly	Pass
2	Check if correct object is identified	Image with single object	Correct Object is identified	Object is not identified correctly	Fail
3	Check if input is image	File other than image	Input file is not an image	Input file is not an image	Pass

7.2.2 Test ID 2:Language Generation

Table 7.2: Language Generation

Sr.No	Test Case Objective	Input Data	Expected Results	Actual Result	Status
1	Check if correct generated captions describe the image correctly	Image	Captions correctly describe the input image	Captions correctly describe the image image	Pass
2	Check if correct generated captions describe the image correctly	Image	Captions correctly describe the input image	Captions do not describe input image as expected	Fail
3	Check if generated captions are grammatically correct	Image	Captions are grammatically correct	Captions are grammatically correct	Pass

7.2.3 Test ID 3:Model Trainer

Table 7.3: Model Trainer

Sr.No	Test Case Objective	Input Data	Expected Results	Actual Result	Status
1	Check if weights are improved for given learning rate	MS-COCO dataset, learning rate	Weights are improved and loss is reduced	Weights are improved and loss is reduced	Fail
2	Check if weights are improved for given learning rate	MS-COCO dataset, learning rate	Weights are improved and loss is reduced	Weights are improved and loss is reduced	Fail
3	Check if time per epoch changes corresponding to batch size	MS-COCO dataset, learning rate	Time per epoch changes with batch size	Time per epoch changes with batch size	Pass

7.2.4 Test ID 4:Model Tester

Table 7.4: Model Tester

Sr.No	Test Case Objective	Input Data	Expected Results	Actual Result	Status
1	Check if the test results correspond to the weights given	MS-COCO dataset, Weights	Test results as predicted for a given image	Test results as predicted for a given image	Pass
2	Check if the test results correspond to the weights given	MS-COCO dataset, Weights	Test results as predicted for a given image	Test results different than the expected result	Fail

Chapter 8

RESULTS

8.1 Outcomes

- The capitons generated by the Image caption generator are very much similar to the actual captions of the corresponding images.
- The system works for any image be it from the testing data set or any other random image.

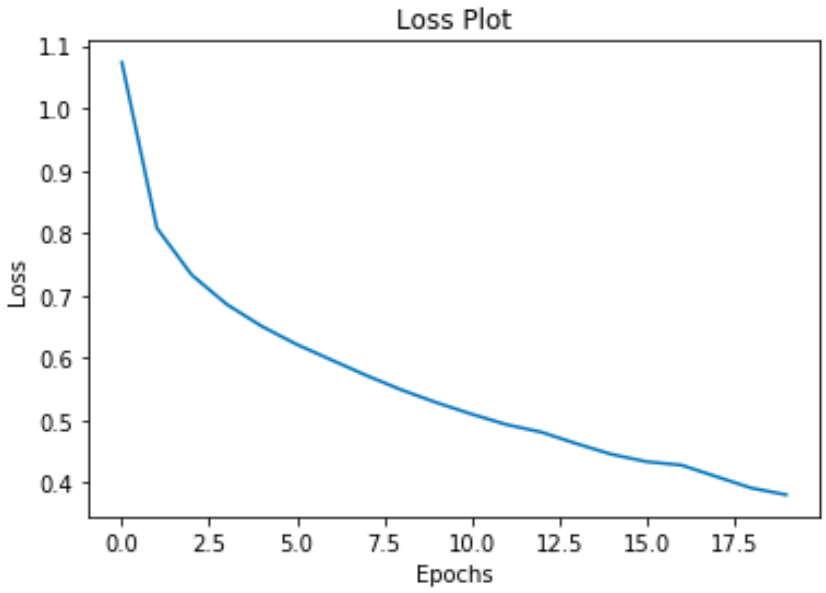


Figure 8.1: Loss Plot

8.2 Screen Shots

8.2.1 Sample Result 1

Figure 8.2: Input Image 1

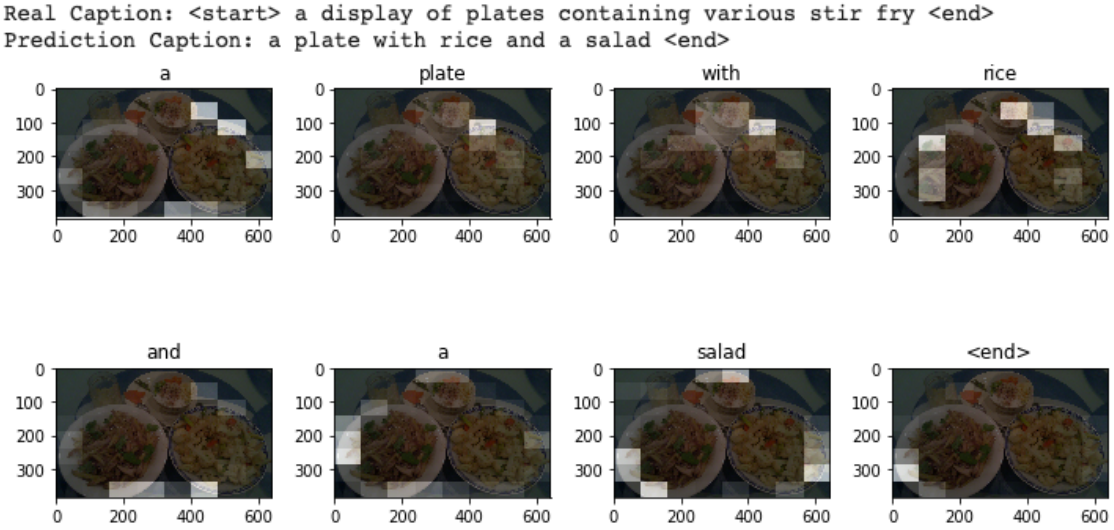


Figure 8.3: Result of Image 1

8.2.2 Sample Result 2

Figure 8.4: Input Image 2

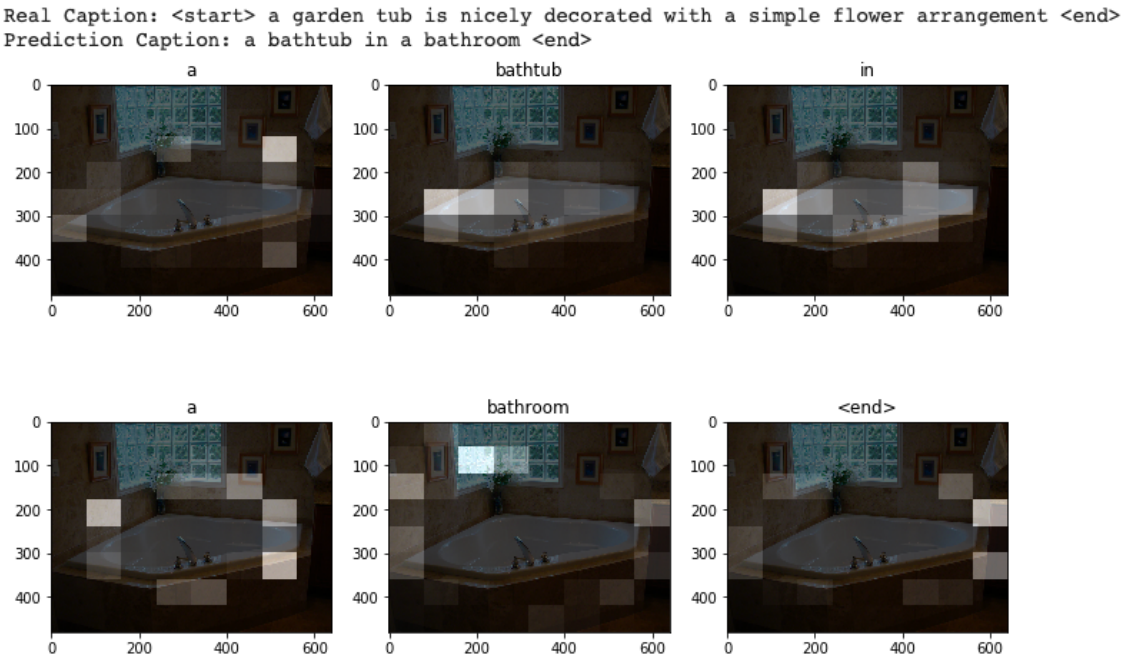


Figure 8.5: Result of Image 2

8.2.3 Sample Result 3

Figure 8.6: Input Image 3

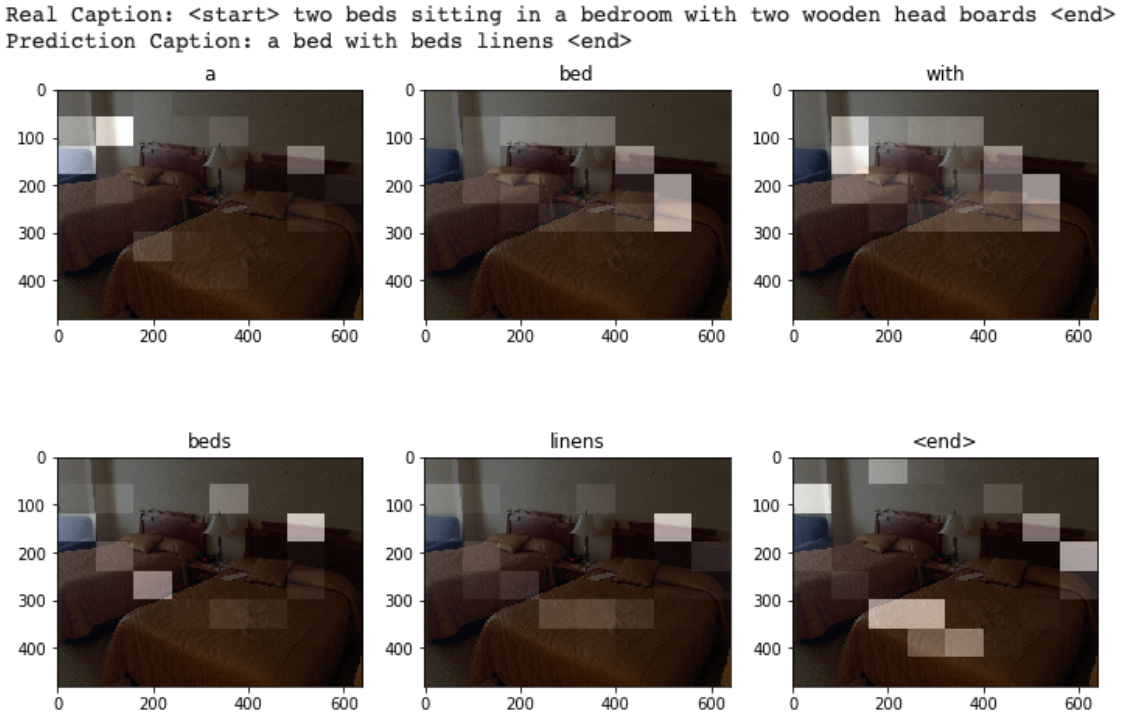


Figure 8.7: Result of Image 3

8.2.4 Sample Result 4

Figure 8.8: Input Image 4

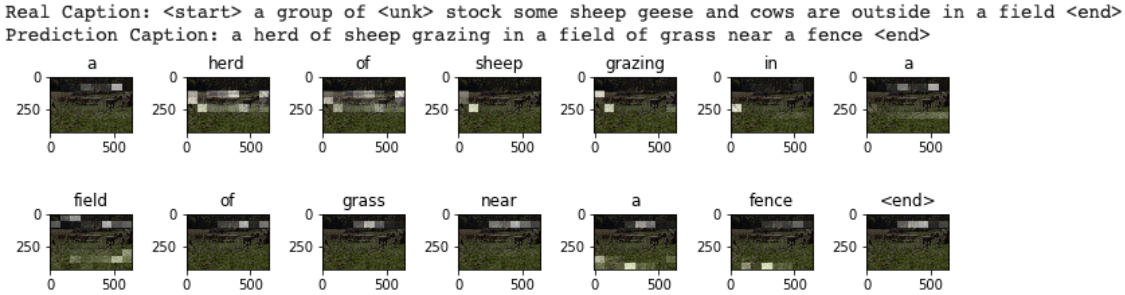


Figure 8.9: Result of Image 4

8.2.5 Sample Result 5

Figure 8.10: Input Image 5

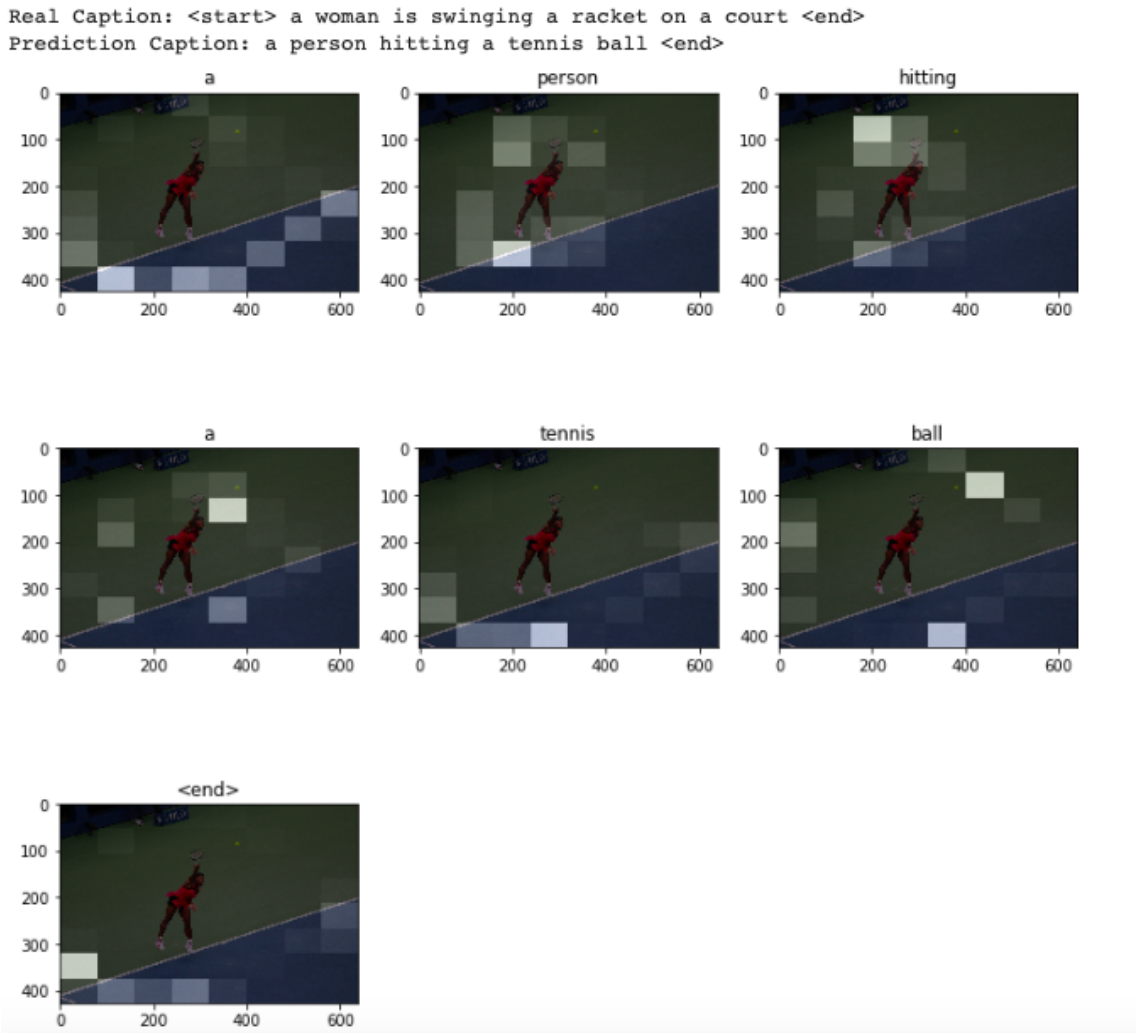


Figure 8.11: Result of Image 5

8.2.6 Sample Result 6

Figure 8.12: Input Image 6

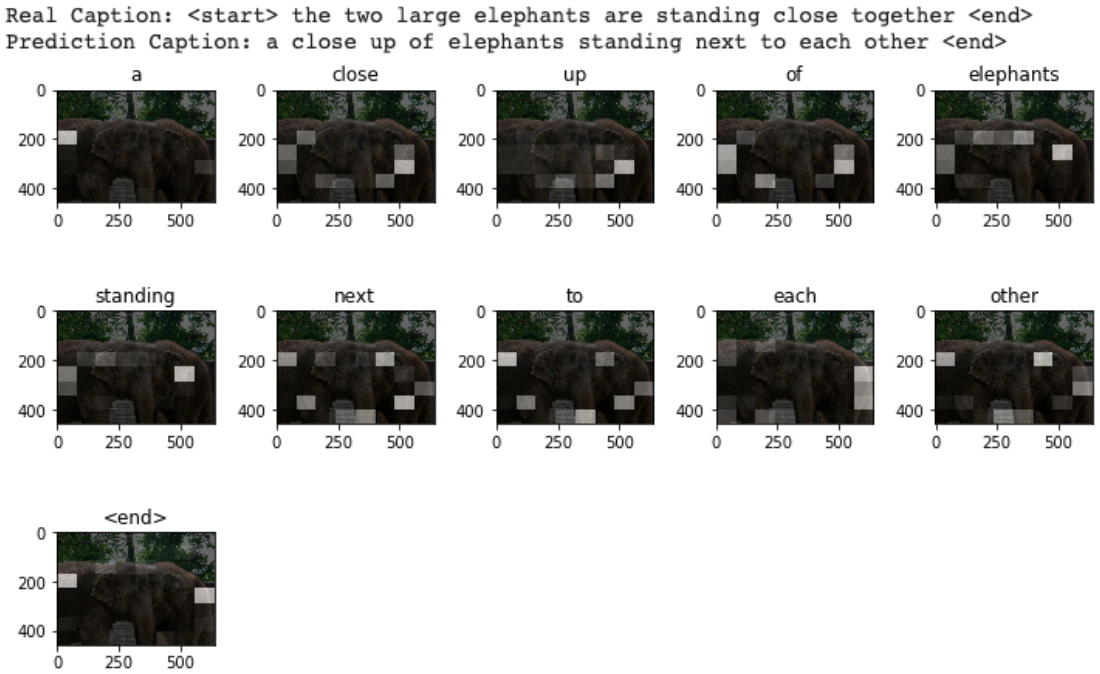


Figure 8.13: Result of Image 6

8.2.7 Sample Result 7

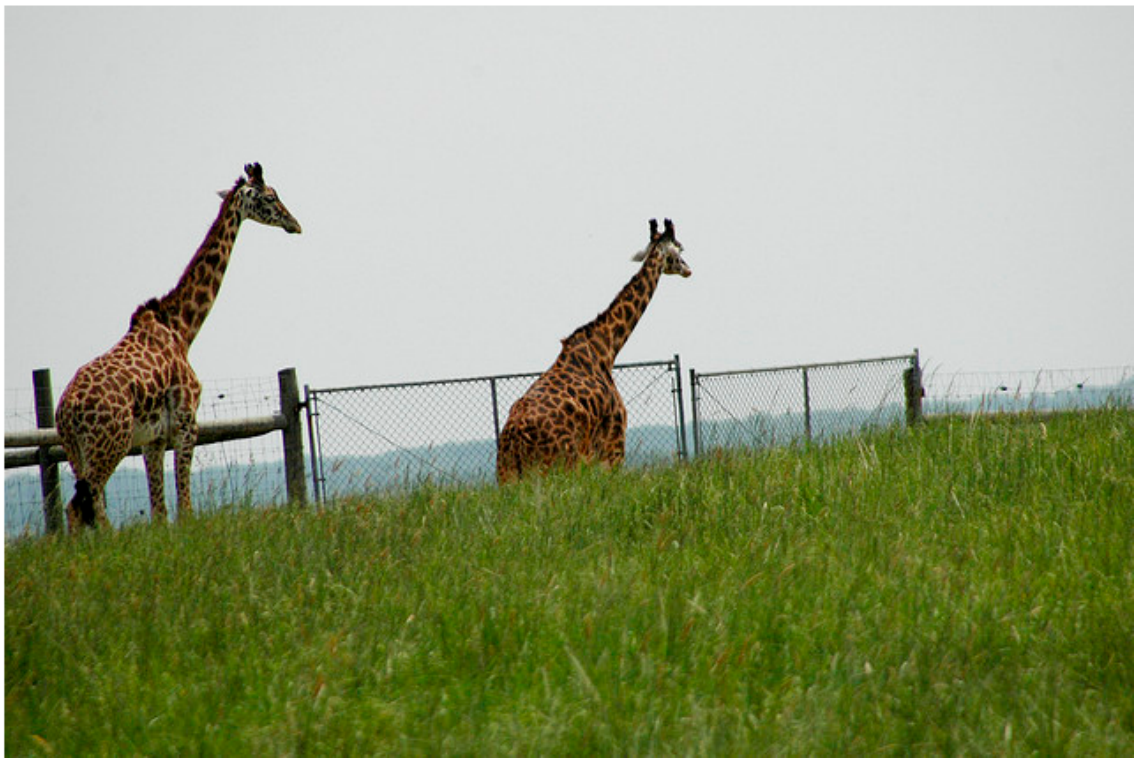


Figure 8.14: Input Image 7

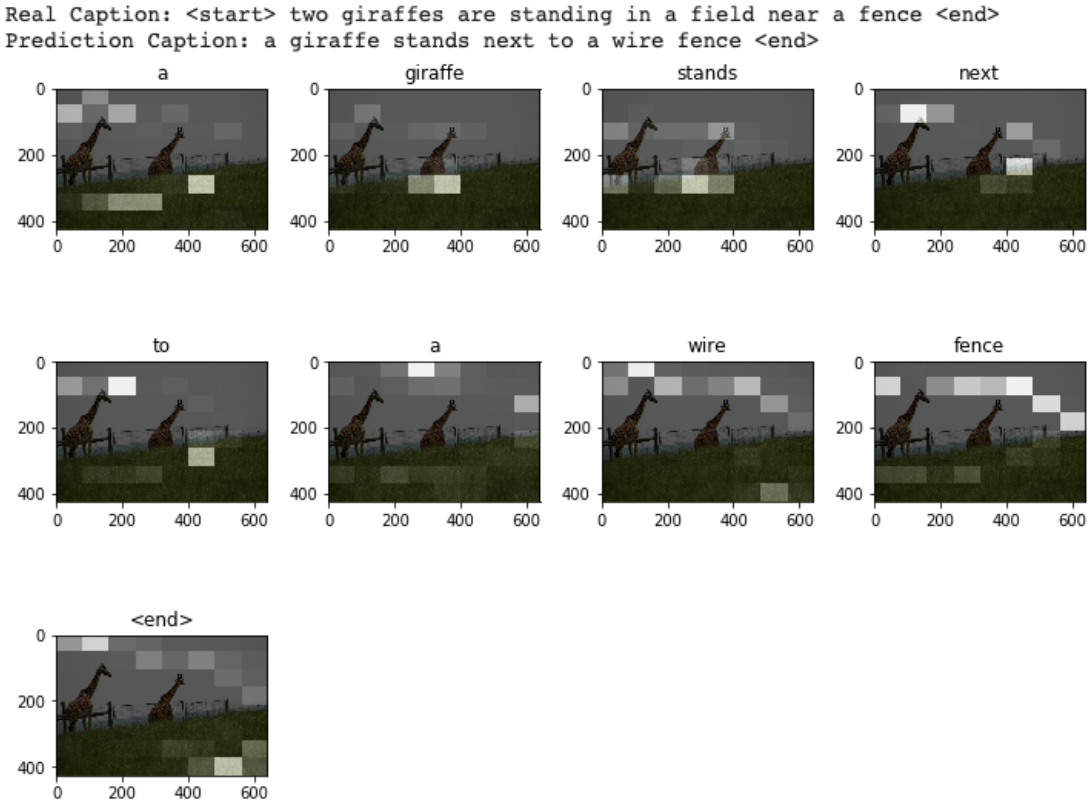


Figure 8.15: Result of Image 7

Chapter 9

CONCLUSION

9.1 Conclusion

We have developed a neural network architecture for image captioning. We have used technologies such as Convolutional Neural Networks, Recurrent Neural Networks with attention model to increase the accuracy and viability of this project. The current experiments are encouraging, providing significant scope in expansion and future research.

9.2 Future Work

- The project can be scaled up to operating on a batch of images at once, to generate a large group of captions for each image in the batch.
- We can also apply a language model on the generated captions to generate a fully fledged story out of it i.e. the model could be fine-tuned additionally for longer captions (stories) instead of captions which are only between 100 and 300 words. A neural network which is able to generate characters and create their backstories and create a cohesive long form story with no human help would one of the major breakthroughs in the field of Image Captioning.
- The same technique can be applied to generate captions/stories for videos, i.e.

using each frame in the video as an image input and passing them through a language model.

- The system can further incorporate multilingual ability.
- A Graphical User Interface can be developed to ease user interaction with the system.

9.3 Applications

- Probably, will be useful in cases/fields where text is most used and with the use of this, you can infer/generate text from images. As in, use the information directly from any particular image in a textual format automatically.
- There are many NLP applications right now, which extract insights/summary from a given text data or an essay etc. The same benefits can be obtained by people who would benefit from automated insights from images.
- A slightly (not-so) long term use case would definitely be, explaining what happens in a video, frame by frame.
- Would serve as a huge help for visually impaired people. Lots of applications can be developed in that space.
- Social Media Platforms like facebook can infer directly from the image, where you are (beach, cafe etc), what you wear (color) and more importantly what youre doing also (in a way).
- Image Captioning can increase the engagement with audiences and are therefore very significant to a lot of campaigns or advertisements.
- Can be used in industries requiring content generation such as Writing, Television and Cinema.

- Effective method of automating interpretation of images, lowering the need for manpower.
- Relatively easy to implement with high levels of accuracy.
- These systems can be trained to operate in a smoother manner and the descriptions and captions are more accurate and insightful than any other technique.

Appendix A

A.1 Feasibility Assessment

Automatic generation of image captioning is a very difficult task, however due to the necessity of computer visions and NLP technologies and its vast applications including early childhood education, blind navigation, image retrieval there has been a vast amount of research done on the topic. The particular project being undertaken is to generate fictional image captions based on uploaded images by the user. This project can be undertaken by the implementation of Neural Networks (CNN or RNN). Other potential alternatives to Neural Networks could be using Machine Learning algorithms for classification and NLP. Another potential alternative that could be used is Reinforcement Learning. However Neural Networks are the more effective choice as demonstrated with their high level of accuracy and effectiveness. To generate the satisfactory output a high level of artificial intelligence is required and hence the use of complex and advanced techniques such as Deep Learning is required.

The Evaluation Criteria for other potential solutions are:

- Effectiveness
- Speed of Computation
- Accuracy
- Reliability

Based on the criteria set above, we can determine that the most technically feasible solution is the use of Deep Learning for Image Captioning. The projects purpose is to develop a image captioning system for caption generation that will generate fictional image captions based on uploaded image by the user. The most feasible solution for the project has been chosen and is now ready for further elaboration.

A.1.1 Technical Feasibility

- Diverse Types of datasets are freely available for use.
- Machine learning libraries necessary for the project are open source.
- Algorithms required for the project have a free to use license.

A.1.2 Economic Feasibility

- The dataset used to train the CNN is freely available online.
- Keras is an open source neural network library.
- Computing power required for training the algorithm was available using the Param-shavak supercomputer.

A.1.3 Time Feasibility

The project has been implemented in 6-7 months.

A.1.4 Privacy Feasibility

Dataset used in open source so no confidential data was acquired.

A.2 Satisfiability Analysis

Informally an algorithm is any well-defined computational procedure that takes some value or a set of vlues as input and produces some value or a set of values as output.

An algorithm is thus a sequence of computational steps that transform the input into output.

To analyze an algorithm is to determine the amount of resources such as time and storage necessary to execute it. In order to choose the best algorithm for a particular task, we need to be able to judge for how long a particular solution will take to run; or how long two solutions will take to run and choose the best of the two. We don't need to know how many minutes and seconds they will take, but we need some way to compare algorithms against one another. This is why we need to analyze an algorithm.

Time Complexity:

P, NP, NP-Hard, NP-Complete

P:

If the running time is some polynomial function of the size of input for instance if the algorithm runs in linear time or quadratic time or cubic time,then we say that the algorithm runs in polynomial time and the problem solves in class P.

NP:

Now there are a lot of programs that don't run in polynomial time on regular computer, but do run in polynomial time time on non-deterministic Turing machine. These programs solve the problems in NP, which stands for non-deterministic polynomial time. A non deterministic Turing machine can do everything a regular computer can do and more. This means there is not necessarily a polynomial-time way to find a solution, but once you have a solution it takes only polynomial time to verify that it is correct.

NP-Complete:

NP is a complexity class which represents the set of all problems X for which it is possible to reduce any other NP problem Y to X in polynomial time. Intuitively it

means that we can solve Y quickly if we know how to solve X quickly. Precisely, Y is reducible to X, if there is a polynomial time algorithm f to transform instances y of Y to instances x = f(y) of X in polynomial time, with the property that the answer to y is yes, if and only if the answer to f(y) is yes.

NP-Hard:

Intuitively, these are the problems that are even harder than the NP-Complete problems. Note that the NP-Hard problems don't have to be in NP and they do not have to be decision problems. The precise definition here is that a problem X is NP-Hard, if there is an NP-Complete problem Y, such that Y is reducible to X in polynomial time. But since any NP-Complete problem can be reducible to any other NP-Complete problem in polynomial time, all NP-Complete problems can be reduced to any NP-Hard problem in polynomial time. Then if there is a solution to one NP-Hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

Algorithm Type:

Current project problem statement is an NP-Complete problem since, it is directly dependent upon the nature of the data and the database capability. This means that it will take non-deterministic polynomial time based on the given image database, algorithms used, and the time taken for processing. Hence a particular P-type time dependent expression cannot be predicted or calculated. Thus the problem sums up to be a NP type, where time dependency is variable, but reaching an expected state is known as end result.

Appendix B

B.1 Paper Publication Details

- **Name of the Conference:** ICICSE 2019 : 7th International Conference on Innovations in Computer Science Engineering at Guru Nanak Institutions (GNI), Hyderabad from 16th - 17th August, 2019
- **Conference Link:** http://www.icicse2019.org
- **Journal:** All accepted registered high quality papers will be published in Scopus Indexed LNNS Springer Series. Original research reported in proceedings and post-proceedings represents the core of Springer series.

  The other papers will be considered for Special Sessions and those papers will be published in UGC Approved journal "Journal of Innovations in Computer Science and Engineering" (JICSE) of volumes 7(2) 8(1) and Special Issue with ISSN 2278-0947.
- **Paper title:** Image Captioning with Visual Attention
- **Author Names:**
  - Sambhav Agarwal (sambhav03@gmail.com)
  - Himani Mali (himani.mali1996@gmail.com)
  - Prithviraj Khelkar (pkhelkar26@gmail.com)
  - Soham Mahabaleshwarkar (soham.mkar@gmail.com)

Appendix C

C.1 Plagiarism Report

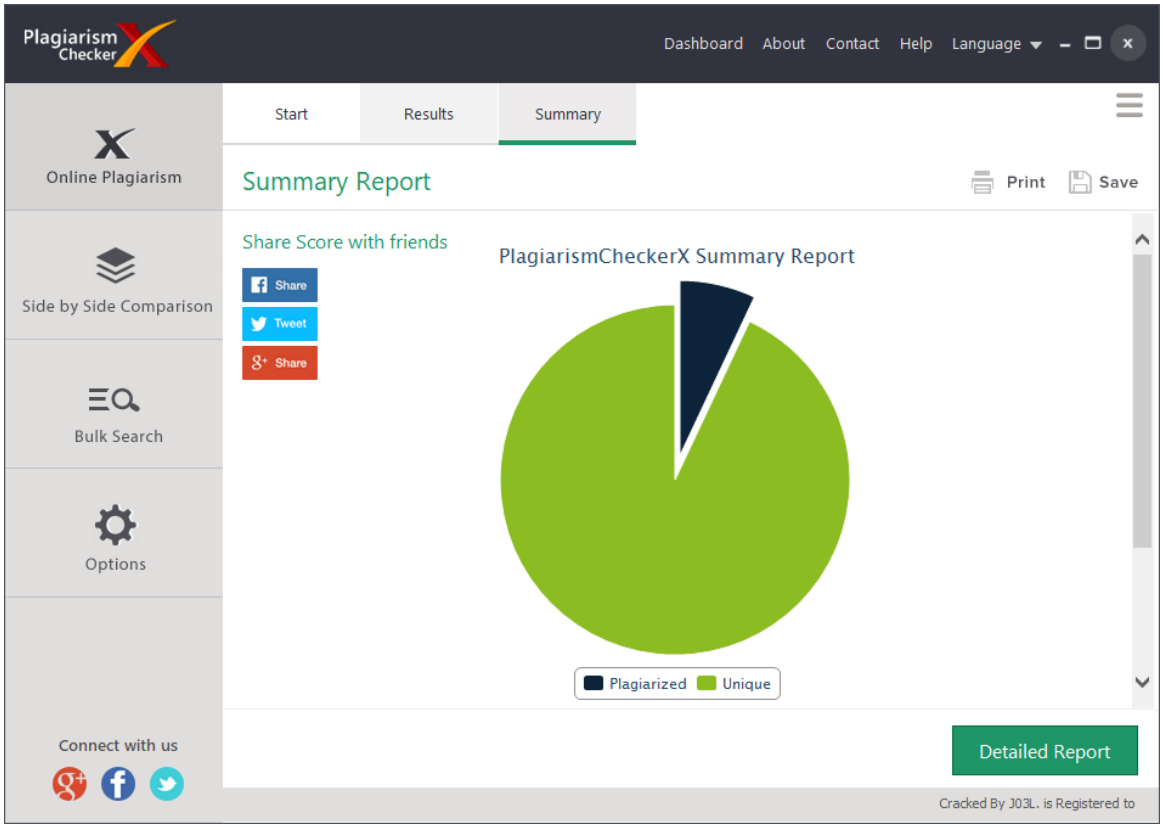


Figure C.1: Plagiarism Report

Bibliography

[1] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. "Show, attend and tell: Neural image caption generation with visual attention."In ICML, 2015. [Online]. Available:https://arxiv.org/abs/1502.03044

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. "Show and tell: A neural image caption generator." In CVPR, 2015. [Online]. Available https://arxiv.org/abs/1411.4555

[3] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick. "Microsoft coco captions: Data collection and evaluation server." In arXiv:1504.00325, 2015. [Online]. Available https://arxiv.org/abs/1504.00325

[4] Q. You, H. Jin, Z.Wang, C. Fang, and J. Luo. "Image captioning with semantic attention." In CVPR, 2016. [Online]. Available https://arxiv.org/abs/1603.03925?context=cs

[5] Ba, Jimmy Lei, Mnih, Volodymyr, and Kavukcuoglu, Koray : Multiple object recognition with visual attention. arXiv:1412.7755, December 2014. [Online]. Available https://dl.acm.org/citation.cfm?id=3045336

[6] J. Johnson, A. Karpathy, and L. Fei-Fei. "Densecap: Fully convolutional localization networks for dense captioning." In CVPR, 2016. [Online]. Available https://arxiv.org/abs/1511.07571

[7] L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. Mooney,K. Saenko, and T. Darrell. "Deep compositional captioning: Describing novel object categories without paired training data." In CVPR,2016. [Online]. Available https://arxiv.org/abs/1511.05284

[8] J. Donahue, L. A. Hendricks, S. Guadarrama,M. Rohrbach, S. Venugopalan,K. Saenko, and T. Darrell." Long-term recurrent convolutional networks for visual recognition and description." In CVPR,2016 [Online]. Available https://arxiv.org/abs/1411.4389

[9] Marco Pedersoli, Thomas Lucas, Cordelia Schmid, Jakob Verbeek Areas of Attention for Image Captioning, 2017 IEEE International Conference on Computer Vision (ICCV).
[Online]. Available https://ieeexplore.ieee.org/document/8237402

[10] Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger , IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2017. [Online]. Available https://arxiv.org/abs/1612.08242

[11] Andrej Karpathy, Li Fei-Fei Deep Visual-Semantic Alignments for Generating Image Descriptions. , IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2015. [Online]. Available https://ieeexplore.ieee.org/document/8237402

[12] Moses Soh Learning CNN-LSTM Architectures for Image Caption Generation. , IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2016. [Online]. Available https://cs224d.stanford.edu/reports/msoh.pdf

[13] Ryan Kiros, Ruslan Salakhutdinov, Richard S. Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models, NIPS 2014 deep learning workshop. [Online]. Available https://arxiv.org/abs/1411.2539

[14] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollr, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence

Zitnick, Geoffrey Zweig. From Captions to Visual Concepts and Back, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Online]. Available https://ieeexplore.ieee.org/document/7298754

[15] Jifeng Dai, Kaiming He, Jian Sun.

BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation, ICCV '15 Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). [Online]. Available https://dl.acm.org/citation.cfm?id=2919723

[16] Christian Szegedy Alexander Toshev Dumitru Erhan.Deep Neural Networks for Object Detection, Published in NIPS 2013. [Online]. Available https://www.semanticscholar.org/

[17] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding , MM '14 Proceedings of the 22nd ACM international conference on Multimedia. [Online]. Available https://dl.acm.org/citation.cfm?id=2654889