MAJOR PROJECT

SUBMITTED in PARTIAL FULFILLMENT of

THE REQUIREMENTS FOR THE AWARD OF THE DEGREE of

BACHELOR OF TECHNOLOGY

(INFORMATION TECHNOLOGY)

SUBMITTED BY

Prakul Sharma - (02115003116)

Ankur Palmia - (00515003116)

Under the Guidance of

(Dr. Prabhjot Kaur)

Department of Information Technology

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY,

JANAKPURI DELHI-58

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

DELHI, INDIA

MAY-2020

Acknowledgement

We truly acknowledge the cooperation and help made by Dr. Prabhjot Kaur, Associate Professor, Department of Information Technology, Maharaja Surajmal Institute of Technology, C-4 Janakpuri, Delhi, Guru Gobind Singh Indraprastha University, Delhi. She has been a constant source of guidance throughout the course of this project. We would also like to thank Mr Mridul Tuteja for his help and guidance in understanding some concepts used in the project. We are also thankful to my friends and family whose silent support led me to complete my project.

Prakul Sharma - (02115003116)

Ankur Palmia - (00515003116)

Certificate

This is to certify that the project entitled Smart STORY TELLER is a bonafide work carried out by Mr. Prakul Sharma and Mr. Ankur Palmia, under my guidance and supervision and submitted in partial fulfillment of B.Tech degree in Information Technology of Maharaja Surajmal Institute of Technology affiliated by Guru Gobind Singh Indraprastha University, Delhi. The work embodied in this project has not been submitted for any other degree or diploma.

Dr. Tripti Sharma	Dr. Prabhjot Kaur
HOD	Associate Professor
Department of Information Technology	Department of Information Technology
MSIT, Delhi	MSIT, Delhi

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER ONE: INTRODUCTION

1.1 COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions.Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

Image processing focuses on, well, processing images. A computer vision system inputs an image and outputs task-specific knowledge, such as object labels and coordinates. Computer vision and image processing work together in many cases. Many computer vision systems rely on image processing algorithms.

Deep Learning in Computer Vision. Deep learning added a huge boost to the already rapidly developing field of computer vision. With deep learning, a lot of new applications of computer vision techniques have been introduced and are now becoming parts of our everyday lives.

The goal of computer vision is to extract useful information from images. This has proved a surprisingly challenging task; it has occupied thousands of intelligent and creative minds over the last four decades, and despite this we are still far from being able to build a general-purpose "seeing machine."

1.2 NATURAL LANGUAGE PROCESSING (NLP)

NLP is a field in machine learning with the ability of a computer to understand, analyze, manipulate, and potentially generate human language.
The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.
Most NLP techniques rely on machine learning to derive meaning from human languages.
NLP algorithms are typically based on machine learning algorithms. Instead of hand-coding large sets of rules, NLP can rely on machine learning to automatically learn these rules by analyzing a set of examples (i.e. a large corpus, like a book, down to a collection of sentences), and making a statical inference.
By utilizing NLP and its components, one can organize the massive chunks of text data, perform numerous automated tasks and solve a wide range of problems such as – automatic summarization,

machine translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation etc.

Before moving further, some terms used are:

- Tokenization – process of converting a text into tokens
- Tokens – words or entities present in the text
- Text object – a sentence or a phrase or a word or an article

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing.

It is predominantly comprised of three steps:

- Noise Removal
- Lexicon Normalization
- Object Standardization

Noise Removal

Any piece of text which is not relevant to the context of the data and the end-output can be specified as the noise.

For example – language stopwords (commonly used words of a language – is, am, the, of, in etc), URLs or links, social media entities (mentions, hashtags), punctuations and industry specific words. This step deals with removal of all types of noisy entities present in the text.

A general approach for noise removal is to prepare a dictionary of noisy entities, and iterate the text object by tokens (or by words), eliminating those tokens which are present in the noise dictionary.

Lexicon Normalization

Another type of textual noise is about the multiple representations exhibited by single word.

For example – “play”, “player”, “played”, “plays” and “playing” are the different variations of the word – “play”, Though they mean different but contextually all are similar. The step converts all the disparities of a word into their normalized form (also known as lemma). Normalization is a pivotal step for feature engineering with text as it converts the high dimensional features (N different features) to the low dimensional space (1 feature), which is an ideal ask for any ML model.

The most common lexicon normalization practices are :

- Stemming: Stemming is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.
- Lemmatization: Lemmatization, on the other hand, is an organized & step by step procedure of obtaining the root form of the word, it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

Object Standardization

Text data often contains words or phrases which are not present in any standard lexical dictionaries. These pieces are not recognized by search engines and models.

Some of the examples are – acronyms, hashtags with attached words, and colloquial slangs. With the help of regular expressions and manually prepared data dictionaries, this type of noise can be fixed, the code below uses a dictionary lookup method to replace social media slangs from a text.

Text to Features (Feature Engineering on text data)

To analyse a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted techniques – Syntactical Parsing, Entities / N-grams / word-based features, Statistical features, and word embeddings.

1.3 DEEP LEARNING

Deep learning (also known as **deep structured learning** or **hierarchical learning**) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised

Deep learning architectures such as deep neural networks, deep belief ne0tworks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

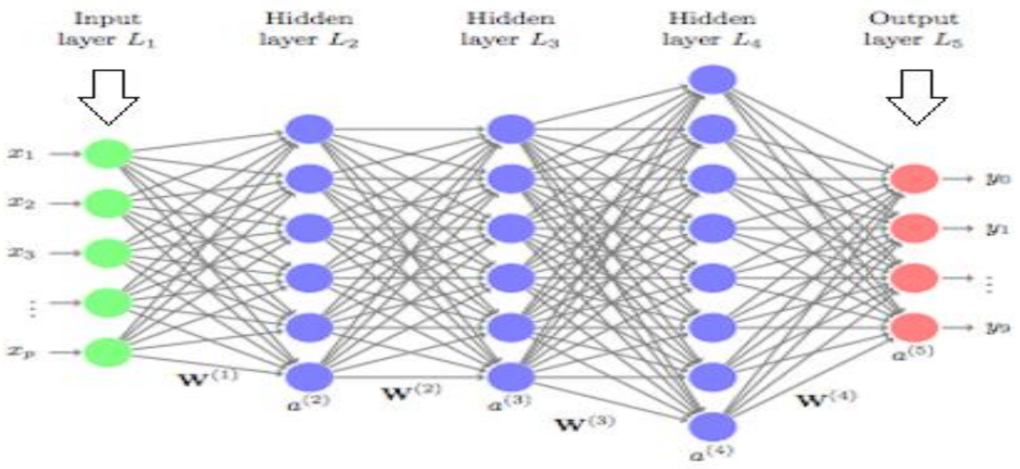


Figure 1.1 DEEP LEARNING MODEL

1.4 CONVOLUTION NEURAL NETWORKS (CNN)

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization.

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. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

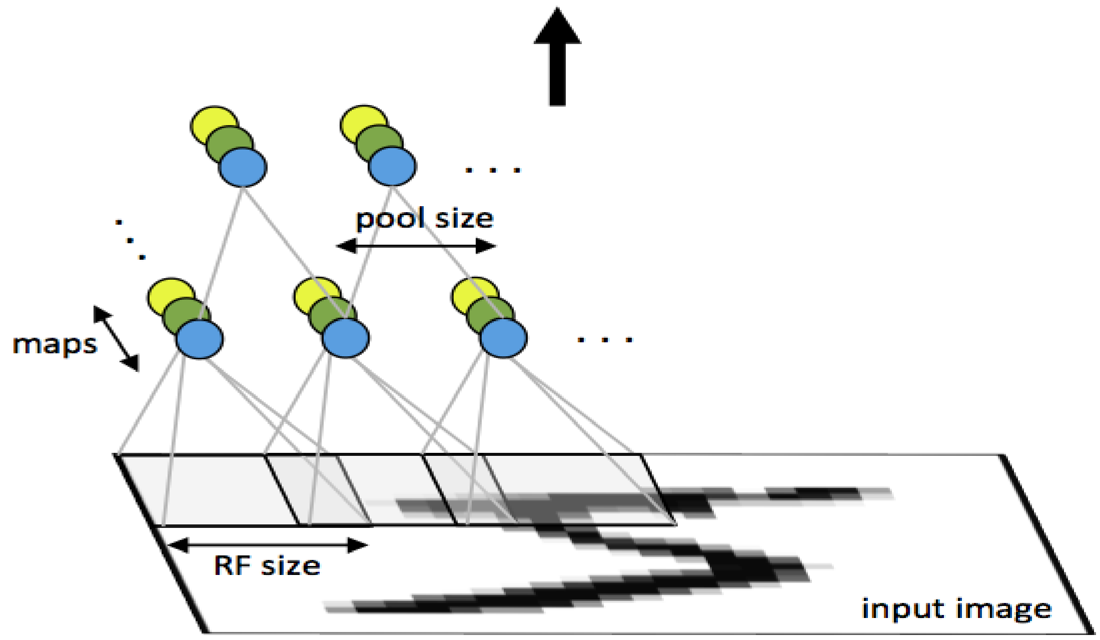


Figure 1.2: First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

1.5 RECURRENT NEURAL NETWORKS (RNN)

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.
Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

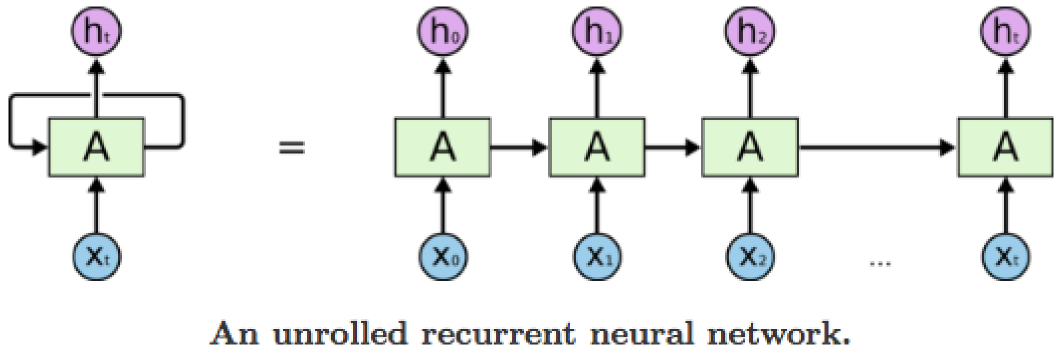


Figure 1.3: RNN basic model

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning.

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other.

RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state.

Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps.

CHAPTER TWO : LITERATURE REVIEW

Ask Your Neurons: A Neural-Based Approach to Answering Questions About Images Mateusz Malinowski, Marcus Rohrbach, Mario Fritz

The Author proposes a novel approach based on recurrent neural networks for the challenging task of answering of questions about images. It combines a CNN with a LSTM into an end-to-end architecture that predict answers conditioning on a question and an image.

Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books , Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, Sanja Fidler

To align movies and books the author exploits a neural sentence embedding that is trained in an unsupervised way from a large corpus of books, as well as a video-text neural embedding for computing similarities between movie clips and sentences in the book.

Local Convolutional Features with Unsupervised Training for Image Retrieval Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronnin, Cordelia Schmid

This paper introduces a deep kernel-based convolutional approach for the description of image patches that does not require supervision. The kernel-based feature representation can be effectively approximated using a simple stochastic gradient optimization procedure, yielding a patch-level descriptor that can be used for image retrieval tasks. Image retrieval is a challenging problem as different images of the same object/scene may exhibit large variations in viewpoint, illumination, scaling, occlusion, etc

A CLOSED-FORM SOLUTION TO PHOTOREALISTIC IMAGE STYLIZATION, BY YIJUN LI, MING-YU LIU, XUETING LI, MING-HSUAN YANG, JAN KAUTZ

Photorealistic image stylization concerns transferring style of a reference photo to a content photo with the constraint that the stylized photo should remain photorealistic. While several photorealistic image stylization methods exist, they tend to generate spatially inconsistent stylizations with noticeable artifacts. In this paper, we propose a method to address these issues. The proposed method consists of a stylization step and a smoothing step.

The goal of photorealistic image stylization is to transfer style of a reference photo to a content photo while keeping the stylized image photorealistic.

Learning to Answer Questions from Image Using Convolutional Neural Network

Lin Ma, Zhengdong Lu, Hang Li

In this paper, we propose to employ the convolutional neural network (CNN) for the image question answering (QA) task. Our proposed CNN provides an end-to-end framework with convolutional architectures for learning not only the image and question representations, but also their inter-modal interactions to produce the answer. More specifically, our model consists of three CNNs: one image CNN to encode the image content, one sentence CNN to compose the words of the question, and one multimodal convolution layer to learn their joint representation for the classification in the space of candidate answer words. We demonstrate the efficacy of our proposed model on the DAQUAR and COCO-QA datasets, which are two benchmark datasets for image QA, with the performances significantly outperforming the state-of-the-art.

Theme-Based Cause-Effect Planning for Multiple-Scene Story Generation Karen Ang, Sherie Yu and Ethel Ong

Early literacy in children begins through picture drawing and the subsequent sharing of an orally narrated story out of the drawn picture. This is the basis for the Picture Books story generation system whose motivation is to produce a textual counterpart of the input picture in order for the child to associate words with images. However, stories are comprised of sequences of events that occur in a cause-effect loop, and the single scene input picture approach may lead to a story whose event flow may not match the child's original intended story. In this paper, we present Picture Books 2, which provides an environment for a child to creatively define a sequence of scenes for his input picture and then uses a theme-based cause-effect planner to generate a fable story narrating the flow of events across the scenes for children age 6-8 years old.

A Survey of Current Datasets for Vision and Language Research

Integrating vision and language has long been a dream in work on artificial intelligence (AI). In the past two years, we have witnessed an explosion of work that brings together vision and language from images to videos and beyond. The available corpora have played a crucial role in advancing this area of research. In this paper, we propose a set of quality metrics for evaluating and analyzing the vision & language datasets and categorize them accordingly. Our analyses show that the most recent datasets have been using more complex language and more abstract concepts, however, there are different strengths and weaknesses .

Thirty-Five Years of Research on Neuro-Linguistic Programming. NLP Research Data

The analysis of the NLP Research Data Base (state of the art) by all measures was like peeling an onion. To reach its core, first I had to remove some useless layers, and once I arrived, I was close to tears. Today, after 35 years of research devoted to the concept, NLP reminds one more of an unstable house built on the sand rather than an edifice founded on the empirical rock. In 1988 Heap passed a verdict on NLP. As the title of his article indicated, it was an interim one. In the conclusions he wrote: If it turns out to be the case that these therapeutic procedures are indeed as rapid and powerful as is claimed, no one will rejoice more than the present author. If however these claims fare no better than the ones already investigated then the final verdict on NLP will be a harsh one indeed .

Translation universals: do they exist? A corpus-based NLP study of convergence and simplification

Convergence and simplification are two of the so-called universals in translation studies. The first one postulates that translated texts tend to be more similar than nontranslated texts. The second one postulates that translated texts are simpler, easier-tounderstand than non-translated ones. This paper discusses the results of a project which applies NLP techniques over comparable corpora of translated and nontranslated texts in Spanish seeking to establish whether these two universals hold Corpas Pastor (2008).

With respect to the simplification hypothesis, it appears to be validated on some, but not all parameters. Indeed, we find that translated texts often exhibit significantly lower lexical density and richness, and seem to be more readable that nontranslated texts (however, statistical significance for the readability differences for this could be established only for one corpus pair). Unexpectedly, translated texts displayed significantly smaller proportion of simple sentences and their sentences turned out to also be significantly shorter. With respect to discourse markers, we find that in two pairs out of three, non-translated texts use discourse markers significantly more often. Interestingly, simplification traits are more visible on the technical translation corpora and, to a somewhat lesser degree, on corpora of professionallyproduced medical translations (where all the features, except sentence length and simple sentences ratio, indicate simpler wordings and formulations), while simplification cannot be found in texts produced by student translators.

Loss of function mutation in glutamic pyruvate transaminase 2 (GPT2) causes developmental encephalopathy

Intellectual disability is genetically heterogeneous, and it is likely that many of the responsible genes have not yet been identified. We describe three siblings with isolated, severe developmental encephalopathy. After extensive uninformative genetic and metabolic testing, whole exome sequencing identified a homozygous novel variant in glutamic pyruvate transaminase 2 (GPT2) or alanine transaminase 2 (ALT2), c.459 C > G p.Ser153Arg that segregated with developmental encephalopathy in the family. This variant was predicted to be damaging by all in silico prediction algorithms. GPT2 is the gene encoding ALT2 which is responsible for the reversible transamination of alanine and 2-oxoglutarate to form pyruvate and glutamate. GPT2 is expressed in brain and is in the pathway to generate glutamate, an excitatory neurotransmitter. Functional assays of recombinant wild-type and mutant ALT2 proteins demonstrated the p.Ser153Arg mutation resulted in a severe loss of enzymatic function. We suggest that recessively inherited loss of function GPT2 mutations are a novel cause of intellectual disability.

Training Optimus Prime, M.D.: Generating Medical Certification Items by Fine-Tuning OpenAI's gpt2 Transformer Model

Objective: Showcasing Artificial Intelligence, in particular deep neural networks, for language modeling aimed at automated generation of medical education test items.
Materials and Methods: OpenAI’s gpt2 transformer language model was retrained using PubMed’s open access text mining database. The retraining was done using toolkits based on tensorflow-gpu available on GitHub, using a workstation equipped with two GPUs.
Results: In comparison to a study that used character based recurrent neural networks trained on open access items, the retrained transformer architecture allows generating higher quality text that can be used as draft input for medical education assessment material. In addition, prompted text generation can be used for production of distractors suitable for multiple choice items used in certification exams. Discussion: The current state of neural network based language models can be used to develop tools in supprt of authoring medical education exams using retrained models on the basis of corpora consisting of general medical text collections.
Conclusion: Future experiments with more recent transformer models (such as Grover, TransformerXL) using existing medical certification exam item pools is expected to further improve results and facilitate the development of assessment materials.

Inception-v3 for flower classification

The study of flower classification system is a very important subject in the field of Botany. A classifier of flowers with high accuracy will also bring a lot of fun to people's lives. However, because of the complex background of flowers, the similarity between the different species of flowers, and the differences among the same species of flowers, there are still some challenges in the recognition of flower images. The traditional flower classification is mainly based on the three features: color, shape and texture, this classification requires people to select features for classification, and the accuracy is not very high. In this paper, based on Inception-v3 model of

TensorFlow platform, we use the transfer learning technology to retrain the flower category datasets, which can greatly improve the accuracy of flower classification.

Fruit Recognition and Grade of Disease Detection using Inception V3 Model

In India, crop yield is declined due to the post-recognition of diseases in fruits/vegetables by the farmers. Farmers face great economic loss worldwide. Diseases in fruits and plants are the main reasons for the agricultural loss. Knowing the health status of fruits/vegetables helps farmers to improve their productivity. This motivates us to design and develop a tool to help farmers detect the diseases in the early stage itself. This work focuses on developing a user-friendly tool which recognizes the level of the disease and grades them accordingly. Inception model uses convolutional neural networks for the classification, which is again retrained using transfer learning technique. The proposed system also grades the fruit based on the percentage of infection. The system is developed in Tensor flow platform. For the proposed work banana, apple and cherry fruits have been considered.

Identification of Plant Leaf Diseases Based on Inception V3 Transfer Learning and Fine-Tuning

Crop disease is a major factor currently to jeopardize agricultural production activities. In recent years, with the great success of deep learning technology in the field of image classification and image recognition, and with the convenient acquisition of crop leaf images, it is possible to automatically identify crop disease through deep learning based on plant leaf disease images. This paper mainly completed the research and analysis of leaf disease identification of agricultural plants based on Inception-V3 neural network model transfer learning and fine-tuning. A large number of model accuracy tests are carried out by training neural networks with different parameters. When the network parameter Batch is set to 100 and the learning rate is set to 0.01, the training precision and test precision of the network reach the maximum. Its training precision rate for crop disease image recognition in the PlantVillage DataSet is 95.8%, and the precision rate on the test set is as high as 93%, and far exceeding the accuracy of manual recognition. This fully proves that the deep learning model based on Inception-V3 neural network can effectively distinguish crop disease.

CHAPTER THREE : PROBLEM STATEMENT

The main aim of this project is to create a deep learning model for Image captioning and image story telling. It will generate a suitable caption in one liner and they it will generate a specific story based on the keywords from the captioner as input.

Being able to automatically describe the content of an image using properly formed English sentences is a very challenging task, but it could have great impact, for instance by helping visually impaired people better understand the content of images on the web. This task is significantly harder, for example, than the well-studied image classification or object recognition tasks, which have been a main focus in the computer vision community.It can be useful in a lot of different areas of interest such as movie story writing , creating a news from a image, write a poem, paragraph.

3.1 PROPOSED SOLUTION ;

To solve this problem we introduce IMAGE STORY TELLER , which uses INCEPTION V3 for image captioning and GPT2 for story generation.

Convolutional neural network architecture, Inception v3, used in this study. Inception v3 network stacks 11 inception modules where each module consists of pooling layers and convolutional filters with rectified linear units as activation function. The input of the model is two-dimensional images of 16 horizontal sections of the brain placed on 4 3 4 grids as produced by the preprocessing step. Three fully connected layers of size 1024, 512, and 3 are added to the final concatenation layer. A dropout with rate of 0.6 is applied before the fully connected layers as means of regularization. The model is pretrained on ImageNet dataset and further fine-tuned with a batch size of 8 and learning rate of 0.0001.

So, can we take advantage of the existence of this model for a custom image feature extraction task like the present one . Well, the concept has a name: **Transfer learning**. It may not be as efficient as a full training from scratch, but is surprisingly effective for many applications. It allows model creation with significantly reduced training data and time by modifying existing rich deep learning models.

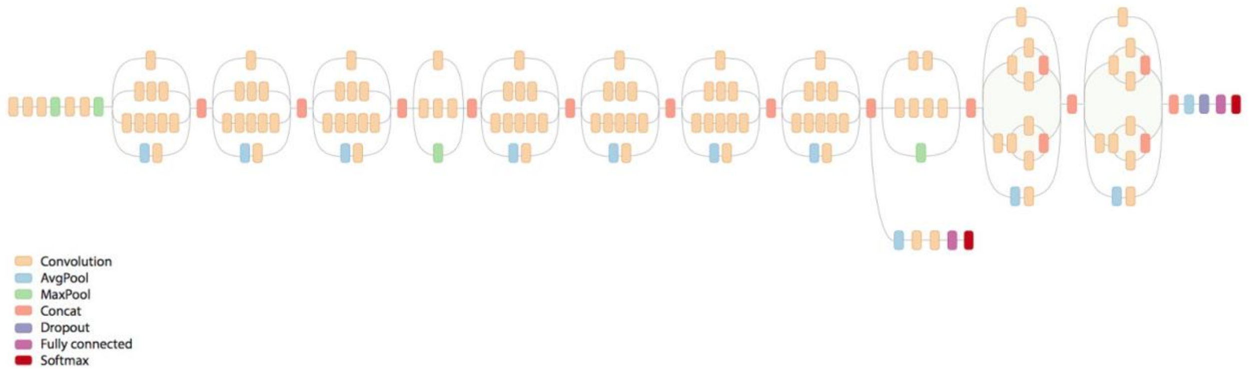


Figure 3.1.1 Inception V3 schematic diagram.

In a neural network, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (input), to the last one (output), possibly after traversing the layers multiple times. As the last hidden layer, the “bottleneck” has enough summarized information to provide the next layer which does the actual extraction task.

The reason our final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1000 classes in ImageNet is often also useful to distinguish between new kinds of objects.

The output of the captioner is fed to the story generator.

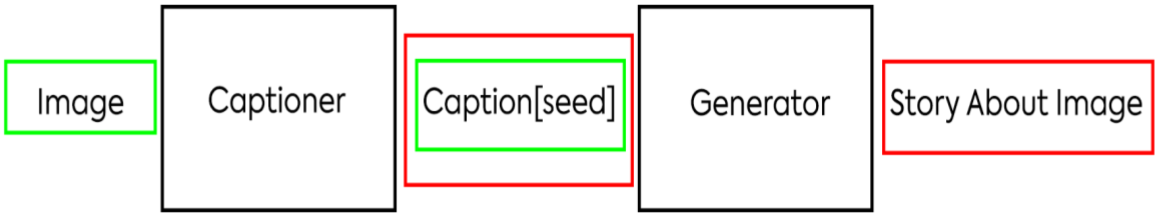


Figure 3.1.2: The output of the captioner is fed to the story generator.

GPT2 Model is used for the story generation part.

GPT-2 is like a black box, taking in inputs and providing outputs. Like previous forms of text generators, the inputs are a sequence of tokens, and the outputs are the probability of the next token in the sequence, with these probabilities serving as weights for the AI to pick the next token in the sequence. In this case, both the input and output tokens are byte pair encodings, which instead of using character tokens (slower to train but includes case/formatting) or word tokens (faster to train but does not include case/formatting) like most RNN approaches, the inputs are “compressed” to the shortest combination of bytes including case/formatting, which serves as a compromise between both approaches but unfortunately adds randomness to the final generation length. The byte pair encodings are later decoded into readable text for human generation.

The pretrained GPT-2 models were trained on websites linked from Reddit. As a result, the model has a very strong grasp of the English language, allowing this knowledge to transfer to other datasets and perform well with only a minor amount of additional finetuning. Due to the English bias in encoder construction, languages with non-Latin characters like Russian and CJK will perform poorly in finetuning.

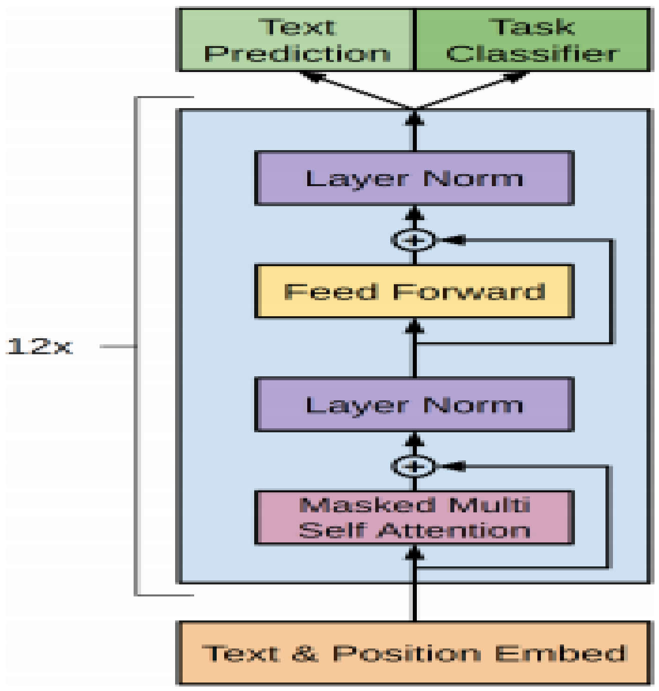


Figure 3.1.3: GPT2 architecture

GPT-2 achieves state-of-the-art scores on a variety of domain-specific language modeling tasks. Our model is not trained on any of the data specific to any of these tasks and is only evaluated on them as a final test; this is known as the “zero-shot” setting. GPT-2 outperforms models trained on domain-specific data sets (e.g. Wikipedia, news, books) when evaluated on those same data sets.

GPT-2 has only a few architecture modification besides having many more parameters and Transformers layers:

- The model uses larger context and vocabulary size
- After the final self-attention block, an additional normalization layer is added
- Similar to a residual unit of type “building block”, layer normalization is moved to the input of each sub-block. It has batch normalization applied before weight layers, which is different from the original type “bottleneck”

3.2 ALGORITHM USED :

3.2.1 SELF ATTENTION THROUGH TRANSFER LEARNING:

Self-attention is a sequence-to-sequence operation: a sequence of vectors goes in, and a sequence of vectors comes out. Let's call the input vectors **x**1,**x**2,…**x**t and the corresponding output vectors **y**1,**y**2,…,**y**t. The vectors all have dimension k.
To produce output vector **y**i, the self attention operation simply takes a weighted average over all the input vectors **yi=∑jwijx**j.
Where j indexes over the whole sequence and the weights sum to one over all j. The weight wij is not a parameter, as in a normal neural net, but it is derived from a function over **x**i and **x**j. The simplest option for this function is the dot product:
w′ij=**x**iT**x**j.
xi is the input vector at the same position as the current output vector **y**i. For the next output vector, we get an entirely new series of dot products, and a different weighted sum.

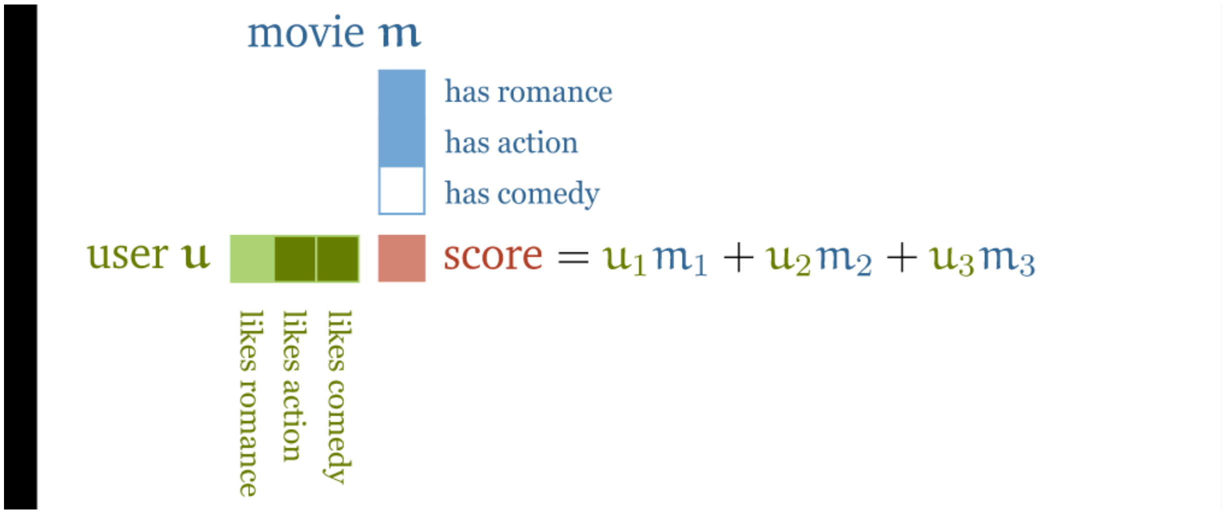


Figure 3.2.1 Self Attention Working

If the signs of a feature match for the user and the movie—the movie is romantic and the user loves romance or the movie is unromantic and the user hates romance—then the resulting dot product gets a positive term for that feature. If the signs don't match—the movie is romantic and the user hates romance or vice versa—the corresponding term is negative.

Furthermore, the magnitudes of the features indicate how much the feature should contribute to the total score: a movie may be a little romantic, but not in a noticeable way, or a user may simply prefer no romance, but be largely ambivalent.

3.3 DATASETS USED

3.3.1. FLICKR

The Flickr30k dataset has become a standard benchmark for sentence-based image description.

The dataset contains 30,000 images , where each image has around 5 captions related to them.

This dataset is built by forming links between images sharing common metadata from Flickr. Edges are formed between images from the same location, submitted to the same gallery, group, or set, images sharing common tags, images taken by friends, etc.

Dataset statistics	
Nodes	105938
Edges	2316948
Nodes in largest WCC	105722 (0.998)
Edges in largest WCC	2316668 (1.000)
Nodes in largest SCC	105722 (0.998)
Edges in largest SCC	2316668 (1.000)
Average clustering coefficient	0.0891
Number of triangles	107987357
Fraction of closed triangles	0.1828
Diameter (longest shortest path)	9
90-percentile effective diameter	4.8

Fig 3.3.1 Flickr Dataset Statistics

3.3.2. IMAGENET

Inception V3 Model is trained on ImageNet.

ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). In ImageNet, we aim to provide on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated. In its completion, we hope ImageNet will offer tens of millions of cleanly sorted images for most of the concepts in the WordNet hierarchy.

The ImageNet project is inspired by a growing sentiment in the image and vision research field – the need for more data. Ever since the birth of the digital era and the availability of web-scale data exchanges, researchers in these fields have been working hard to design more and more sophisticated algorithms to index, retrieve, organize and annotate multimedia data. But good research needs good resource. To tackle these problem in large-scale (think of your growing personal collection of digital images, or videos, or a commercial web search engine’s database), it would be tremendously helpful to researchers if there exists a large-scale image database. This is the motivation for us to put together ImageNet. We hope it will become a useful resource to our research community, as well as anyone whose research and education would benefit from using a large image database.

The goal of developing the dataset was to provide a resource to promote the research and development of improved methods for computer vision.

3.3.3. GUTENBERG

GPT2 Model is trained on Gutenberg.

This is a collection of 3,036 English books written by 142 authors. This collection is a small subset of the Project Gutenberg corpus. All books have been manually cleaned to remove metadata, license information, and transcribers' notes, as much as possible.

Project Gutenberg is a volunteer effort to digitize and archive cultural works, to "encourage the creation and distribution of eBooks". It was founded in 1971 by American writer Michael S. Hart and is the oldest digital library. Most of the items in its collection are the full texts of public domain books.

The use of Project Gutenberg (PG) as a text corpus has been extremely popular in statistical analysis of language for more than 25 years. However, in contrast to other major linguistic datasets of similar importance, no consensual full version of PG exists to date. In fact, most PG studies so far either consider only a small number of manually selected books, leading to potential biased subsets, or employ vastly different pre-processing strategies (often specified in insufficient details), raising concerns regarding the reproducibility of published results. In order to address these shortcomings, here we present the Standardized Project Gutenberg Corpus (SPGC), an open science approach to a curated version of the complete PG data containing more than 50,000 books and more than 3×109 word-tokens.

3.4 LIBRARIES USED

3.4.1 NLTK

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

3.4.2 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML.[1][2] Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System),[3] and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model[

3.4.3 Pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.[3]

3.4.4 Open CV

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source BSD license.

OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

3.5 SOFTWARE USED

3.5.1 GOOGLE COLAB

Google Colab is a free cloud service and now it supports free GPU!

You can;

i) improve your Python programming language coding skills.

ii) develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

The most important feature that distinguishes Colab from other free cloud services is; Colab provides GPU and is totally free.

If you want to create a machine learning model but say you don't have a computer that can take the workload, Google Colab is the platform for you. Even if you have a GPU or a good computer creating a local environment with anaconda and installing packages and resolving installation issues are a hassle. Colaboratory is a free Jupyter notebook environment provided by Google where you can use free GPUs and TPUs which can solve all these issues.

3.5.2 JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3.5.3 KERAS

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model .

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

3.5.4 PYTORCH

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing. It is primarily developed by Facebook's artificial intelligence research group. It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ frontend. Furthermore, Uber's Pyro probabilistic programming language software uses PyTorch as a backend.
PyTorch provides two high-level features:
Tensor computing (like NumPy) with strong acceleration via GPU’sDeep
neural networksbuilt on a tape-based autodiff system.

3.5.5 INCEPTION V3

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

3.5.6 GPT2

Developed by OpenAI, GPT-2 is a pre-trained language model which we can use for various NLP tasks, such as: Text generation. Language translation. Building question-answering systems, and so on.
GPT-2 stands for “Generative Pretrained Transformer 2”: “Generative” means the model was trained to predict (or “generate”) the next token in a sequence of tokens in an unsupervised way.

3.6 DATA FLOW DIAGRAMS

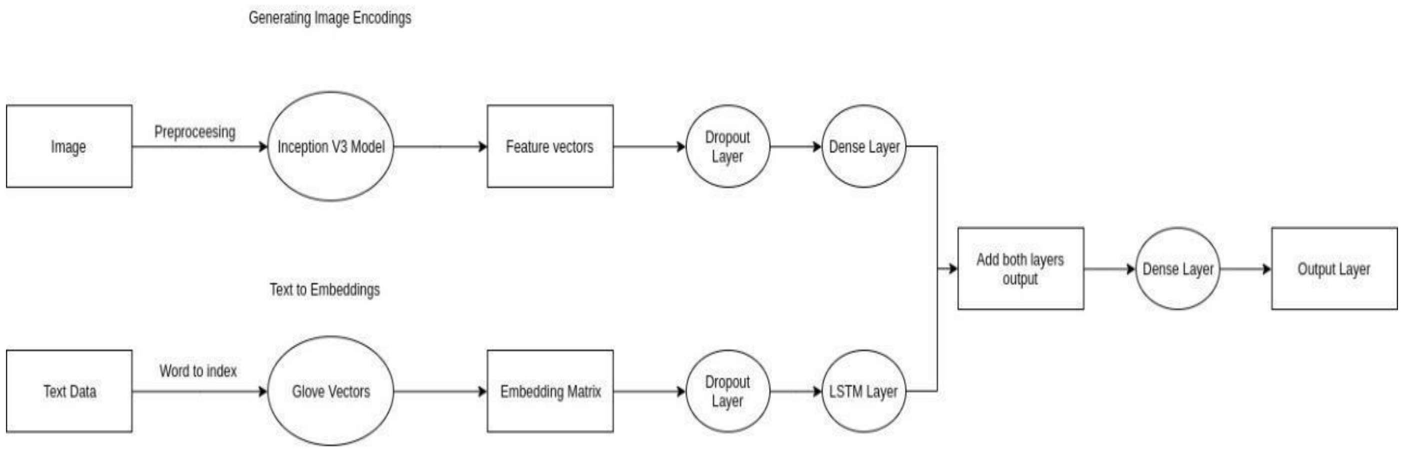


Figure 3.6.1 IMAGE CAPTIONING DFD

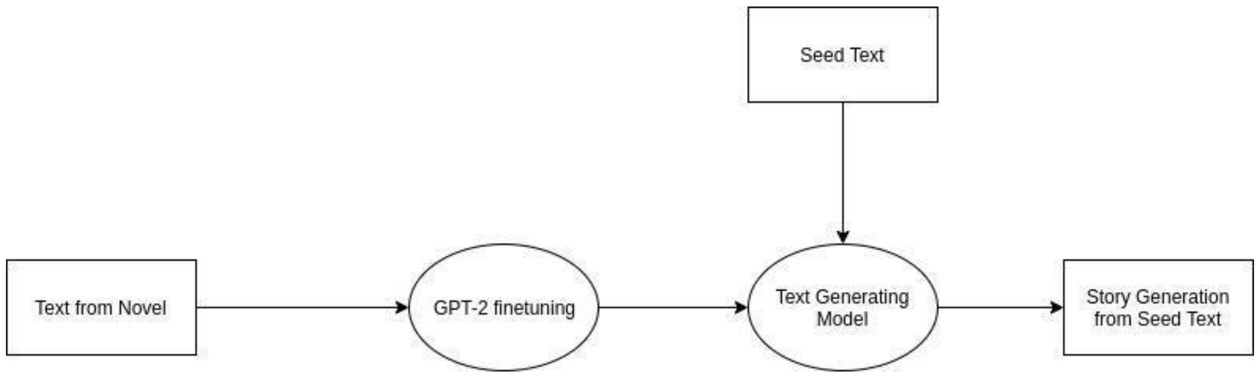


Figure 3.6.2 TEXT GENERATOR DFD

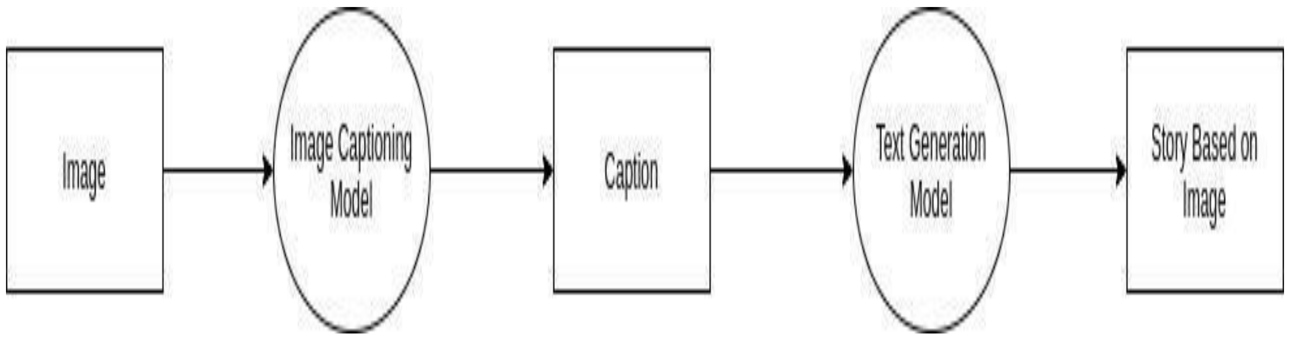
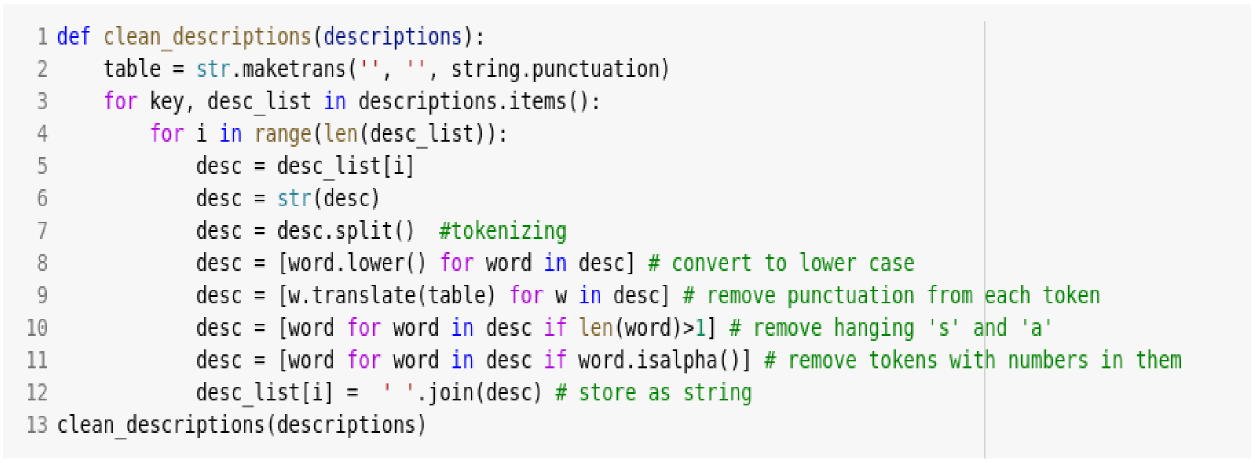


Figure 3.6.2 COMPLETE DFD

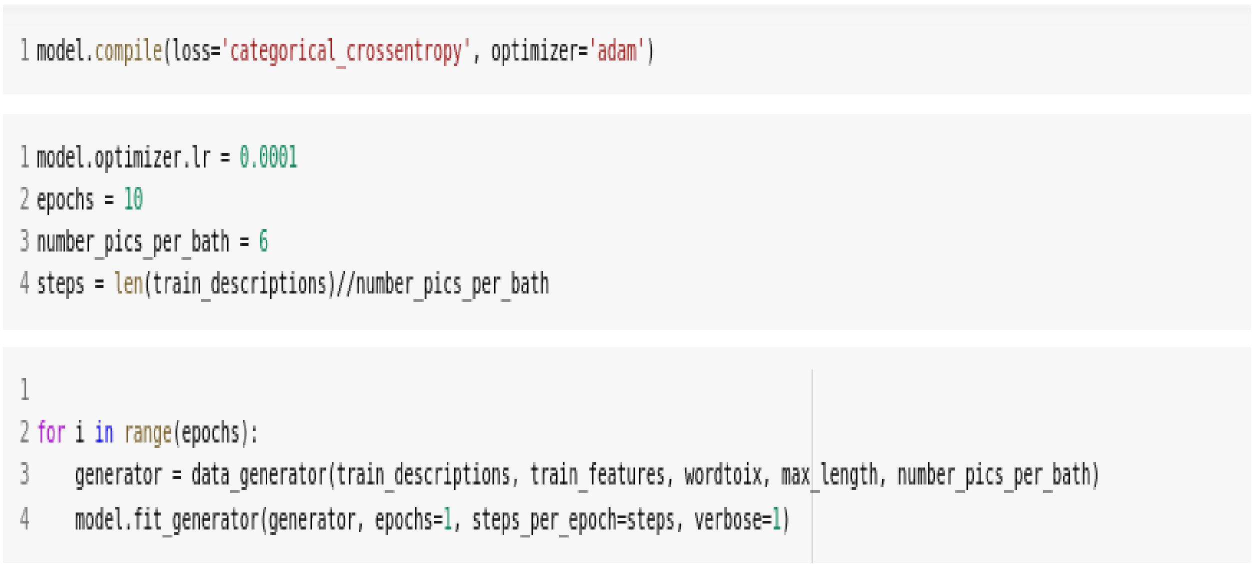
CHAPTER FOUR: SCREENSHOTS

4.1 CODE SNIPETS



```python
def clean_descriptions(descriptions):
    table = str.maketrans('', '', string.punctuation)
    for key, desc_list in descriptions.items():
        for i in range(len(desc_list)):
            desc = desc_list[i]
            desc = str(desc)
            desc = desc.split()  #tokenizing
            desc = [word.lower() for word in desc] # convert to lower case
            desc = [w.translate(table) for w in desc] # remove punctuation from each token
            desc = [word for word in desc if len(word)>1] # remove hanging 's' and 'a'
            desc = [word for word in desc if word.isalpha()] # remove tokens with numbers in them
            desc_list[i] =  ' '.join(desc) # store as string
clean_descriptions(descriptions)
```

Figure 4.1 CLEANING THE TEXT FOR IMAGE CAPTIONER

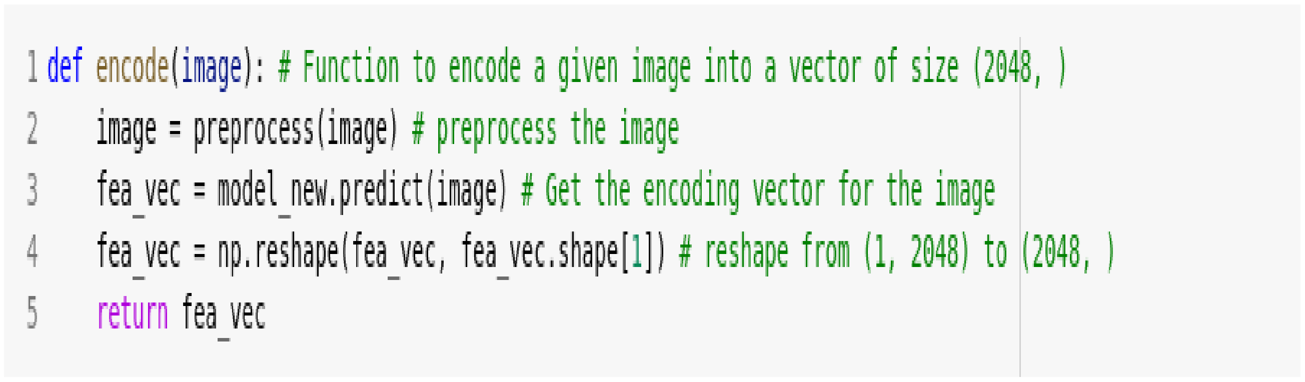


```python
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

```python
model.optimizer.lr = 0.0001
epochs = 10
number_pics_per_bath = 6
steps = len(train_descriptions)//number_pics_per_bath
```

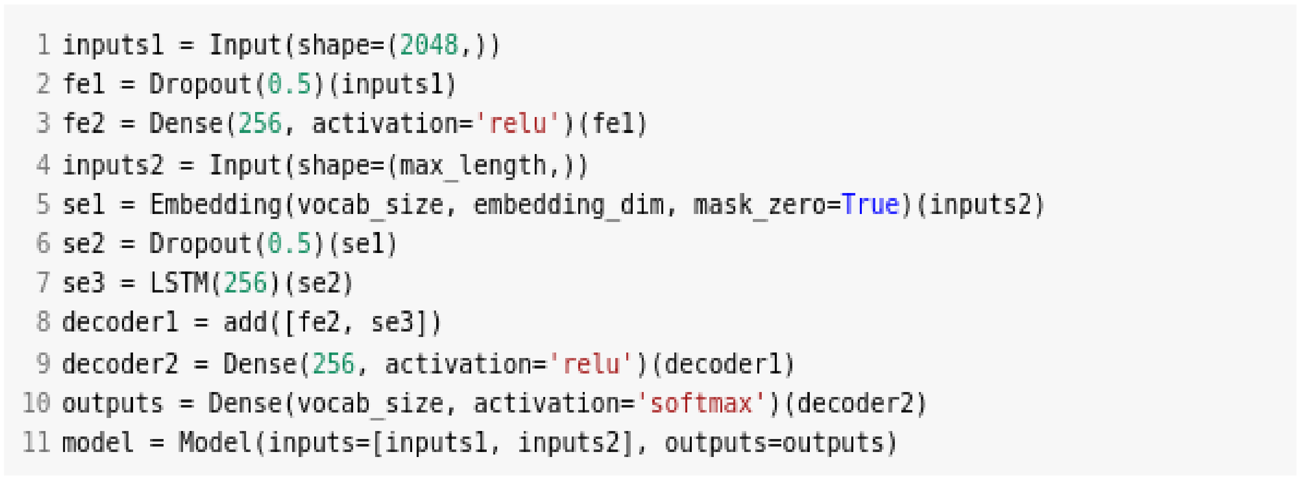
```python

for i in range(epochs):
    generator = data_generator(train_descriptions, train_features, wordtoix, max_length, number_pics_per_bath)
    model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
```

Figure 4.2 COMPILING MODEL , SETTING HYPERPARAMETER AND TRAINING



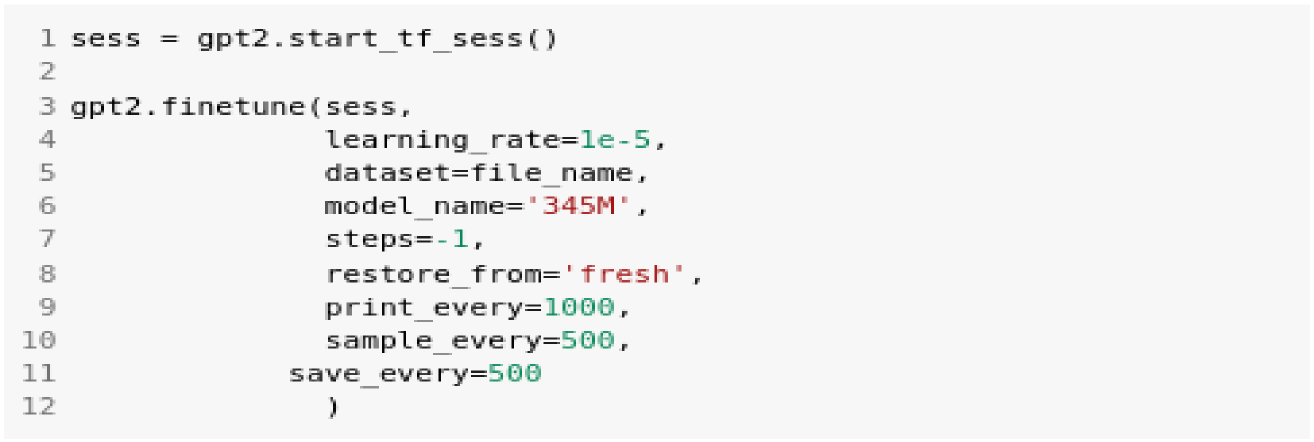
```python
def encode(image): # Function to encode a given image into a vector of size (2048, )
    image = preprocess(image) # preprocess the image
    fea_vec = model_new.predict(image) # Get the encoding vector for the image
    fea_vec = np.reshape(fea_vec, fea_vec.shape[1]) # reshape from (1, 2048) to (2048, )
    return fea_vec
```

Figure 4.3 CREATING FEATURE VECTORS FROM IMAGE



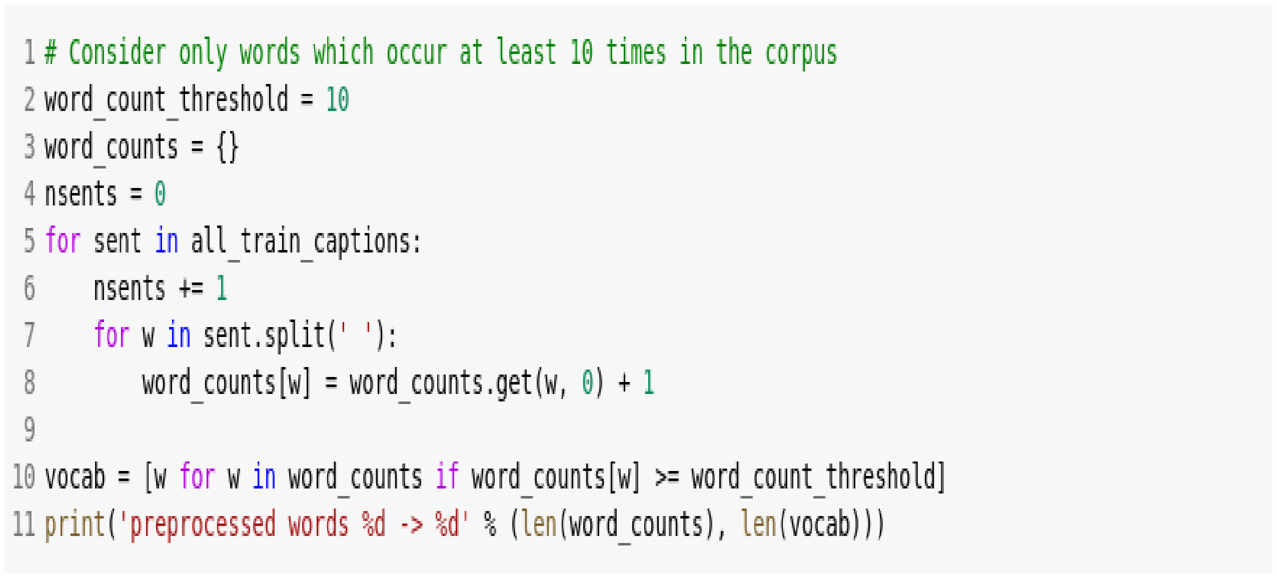
```python
inputs1 = Input(shape=(2048,))
fe1 = Dropout(0.5)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs2)
se2 = Dropout(0.5)(se1)
se3 = LSTM(256)(se2)
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)
model = Model(inputs=[inputs1, inputs2], outputs=outputs)
```

Figure 4.4 CREATING MODEL FOR IMAGE CAPTIONER



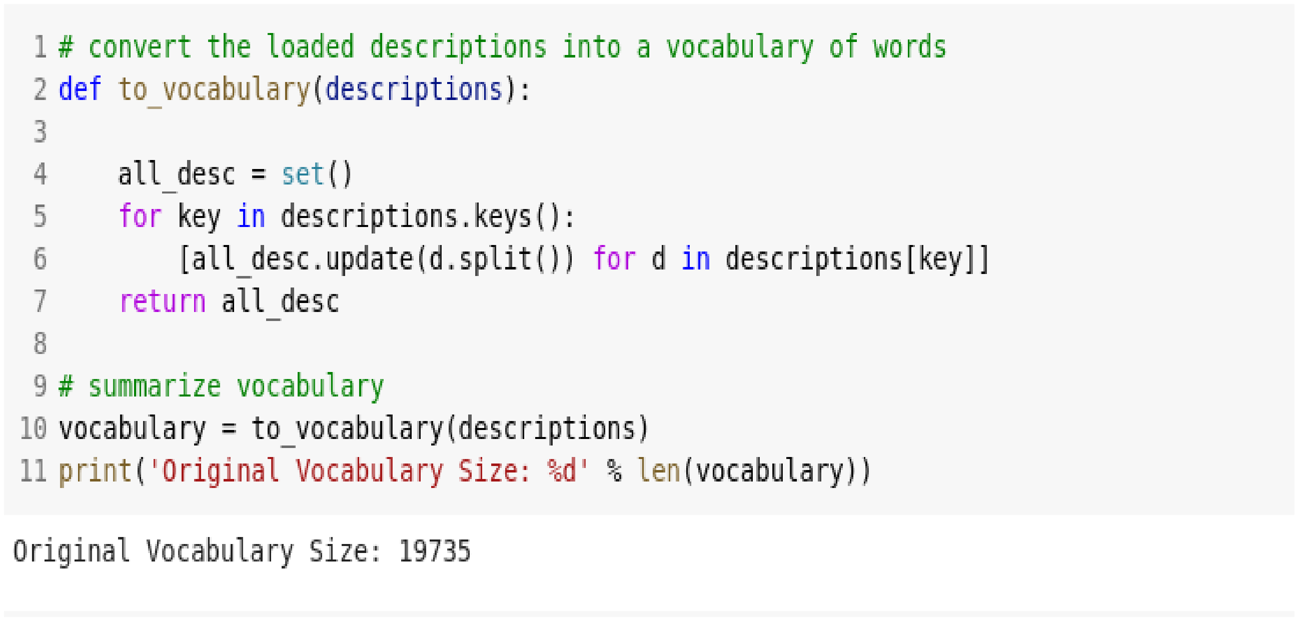
```python
sess = gpt2.start_tf_sess()

gpt2.finetune(sess,
              learning_rate=1e-5,
              dataset=file_name,
              model_name='345M',
              steps=-1,
              restore_from='fresh',
              print_every=1000,
              sample_every=500,
            save_every=500
              )
```

Figure 4.5 CREATING SESSION FOR GPT2



```python
# Consider only words which occur at least 10 times in the corpus
word_count_threshold = 10
word_counts = {}
nsents = 0
for sent in all_train_captions:
    nsents += 1
    for w in sent.split(' '):
        word_counts[w] = word_counts.get(w, 0) + 1

vocab = [w for w in word_counts if word_counts[w] >= word_count_threshold]
print('preprocessed words %d -> %d' % (len(word_counts), len(vocab)))
```

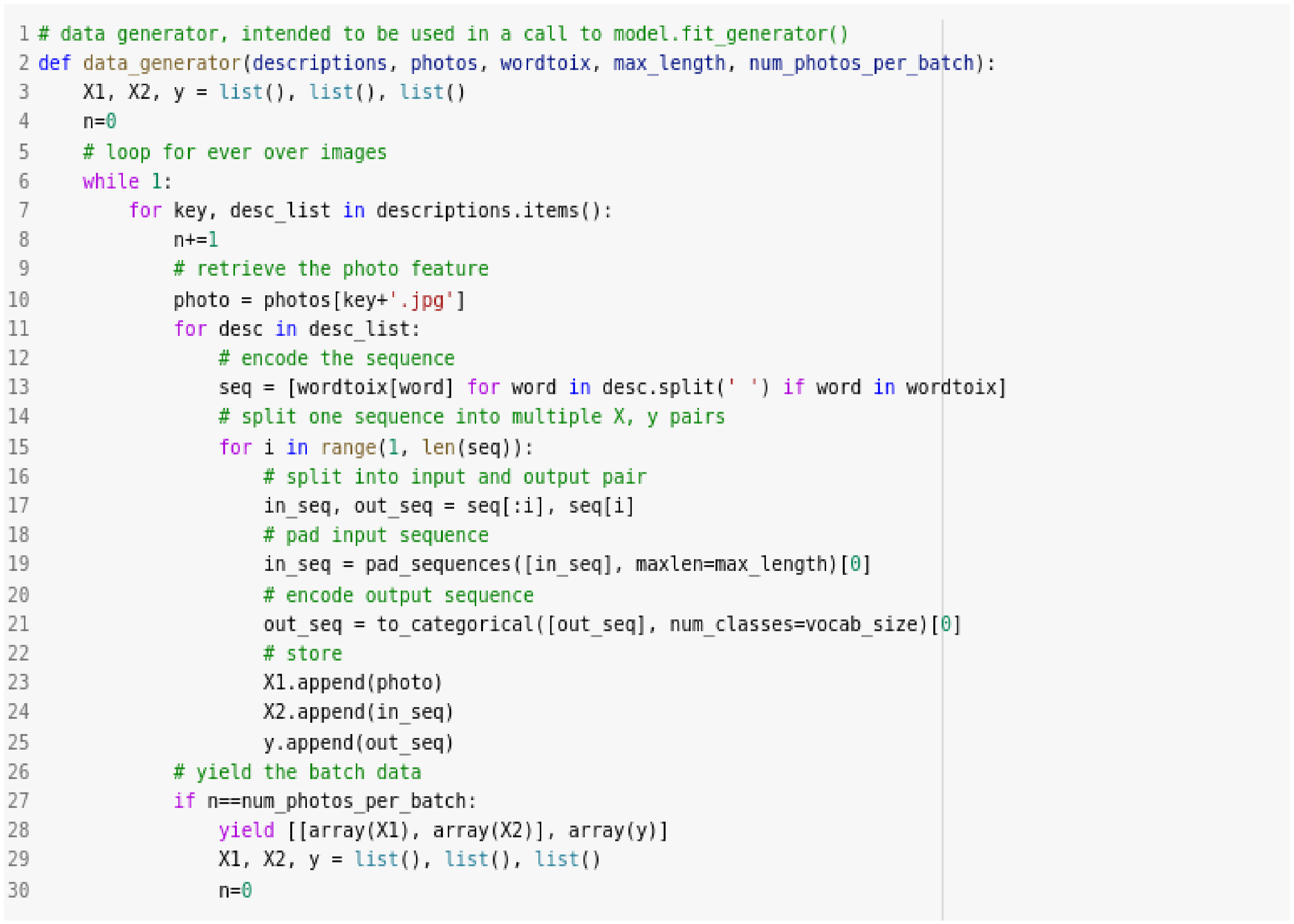
Figure 4.6 CREATING VOCAB WITH THRESHOLD



```python
# convert the loaded descriptions into a vocabulary of words
def to_vocabulary(descriptions):

    all_desc = set()
    for key in descriptions.keys():
        [all_desc.update(d.split()) for d in descriptions[key]]
    return all_desc

# summarize vocabulary
vocabulary = to_vocabulary(descriptions)
print('Original Vocabulary Size: %d' % len(vocabulary))
```

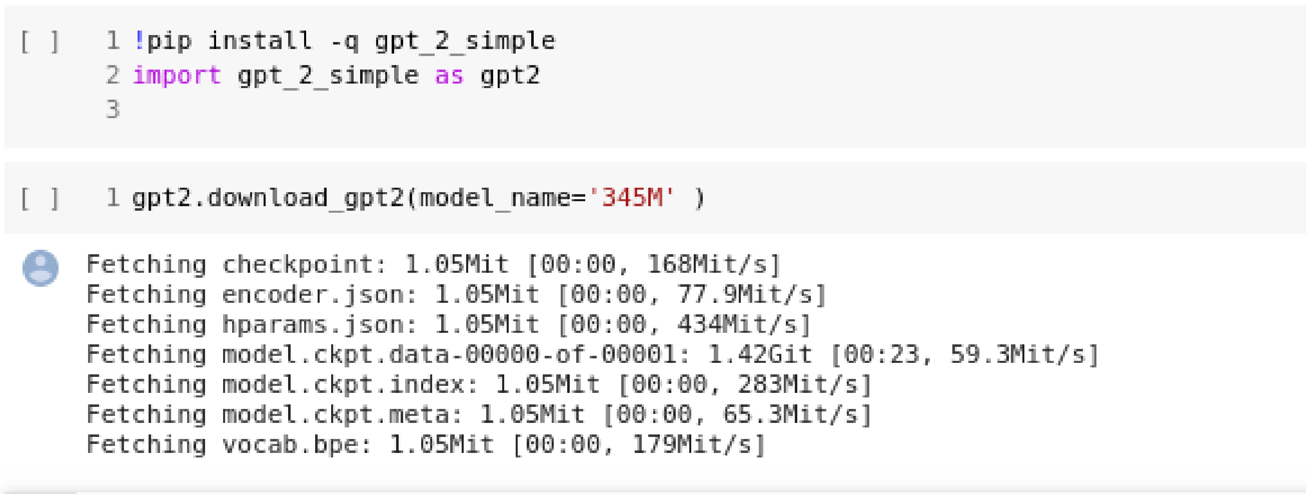
```
Original Vocabulary Size: 19735
```

Figure 4.7 CREATING VOCAB FROM DESCRIPTION



```python
# data generator, intended to be used in a call to model.fit_generator()
def data_generator(descriptions, photos, wordtoix, max_length, num_photos_per_batch):
    X1, X2, y = list(), list(), list()
    n=0
    # loop for ever over images
    while 1:
        for key, desc_list in descriptions.items():
            n+=1
            # retrieve the photo feature
            photo = photos[key+'.jpg']
            for desc in desc_list:
                # encode the sequence
                seq = [wordtoix[word] for word in desc.split(' ') if word in wordtoix]
                # 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(photo)
                    X2.append(in_seq)
                    y.append(out_seq)
            # yield the batch data
            if n==num_photos_per_batch:
                yield [[array(X1), array(X2)], array(y)]
                X1, X2, y = list(), list(), list()
                n=0
```

Figure 4.8 DATA GENERATOR FOR MODEL TRAINING

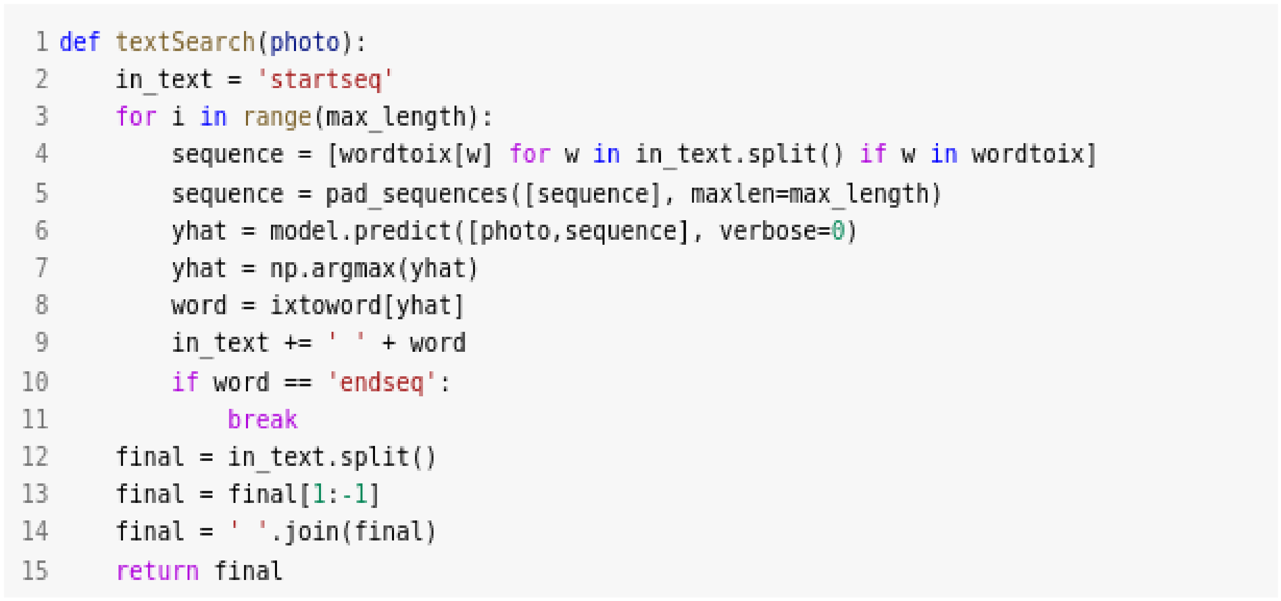


```
[ ]   !pip install -q gpt_2_simple
      import gpt_2_simple as gpt2

```

```
[ ]   gpt2.download_gpt2(model_name='345M' )
```

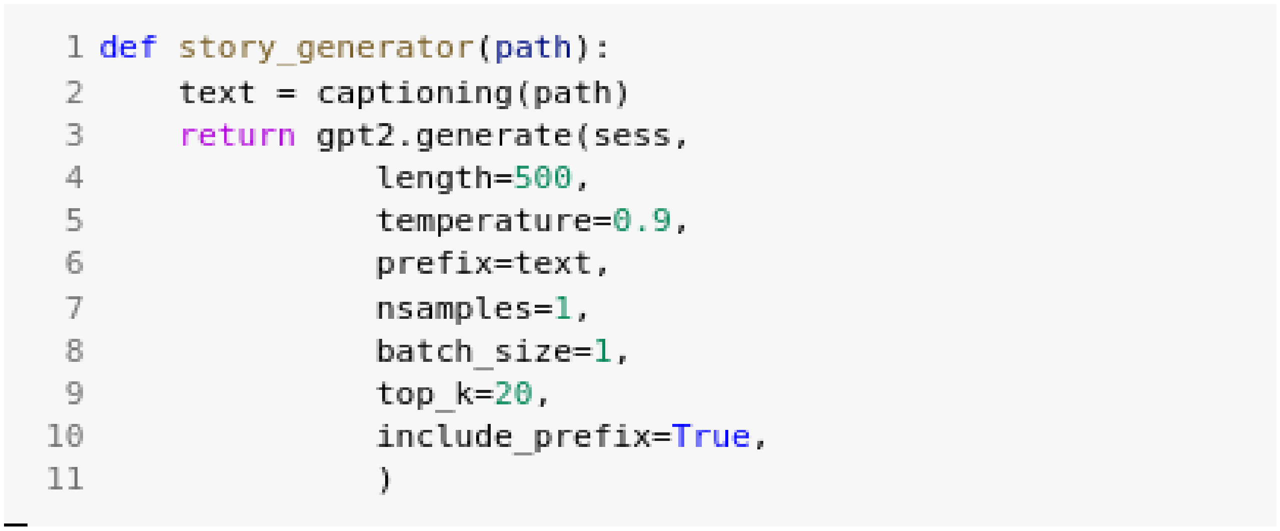
```
Fetching checkpoint: 1.05Mit [00:00, 168Mit/s]
Fetching encoder.json: 1.05Mit [00:00, 77.9Mit/s]
Fetching hparams.json: 1.05Mit [00:00, 434Mit/s]
Fetching model.ckpt.data-00000-of-00001: 1.42Git [00:23, 59.3Mit/s]
Fetching model.ckpt.index: 1.05Mit [00:00, 283Mit/s]
Fetching model.ckpt.meta: 1.05Mit [00:00, 65.3Mit/s]
Fetching vocab.bpe: 1.05Mit [00:00, 179Mit/s]
```

Figure 4.9 DOWNLOADING AND INSTALLING NECESSARY GPT2 FILES



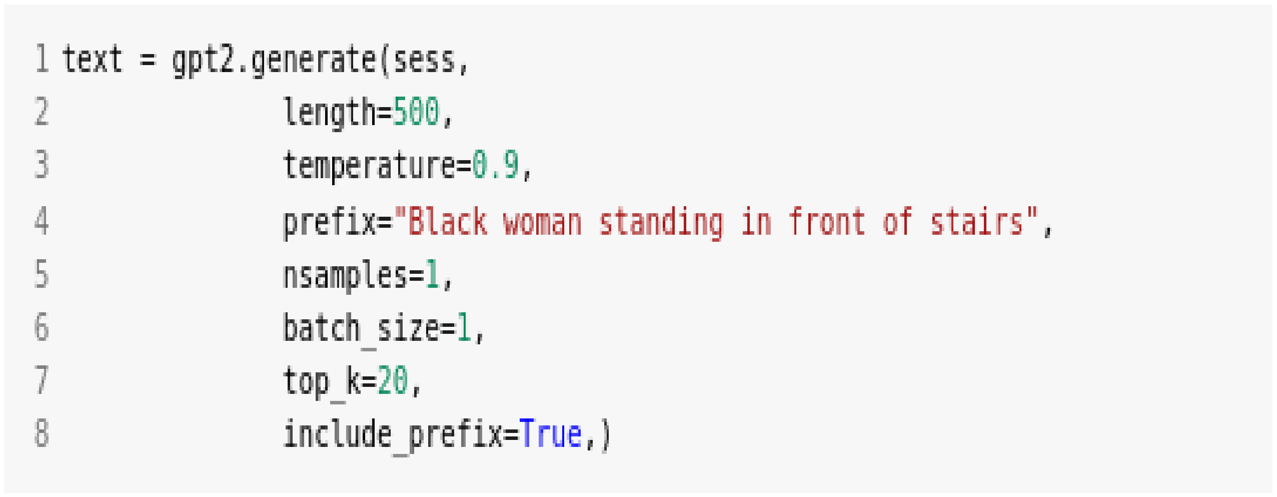
```python
def textSearch(photo):
    in_text = 'startseq'
    for i in range(max_length):
        sequence = [wordtoix[w] for w in in_text.split() if w in wordtoix]
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = model.predict([photo,sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = ixtoword[yhat]
        in_text += ' ' + word
        if word == 'endseq':
            break
    final = in_text.split()
    final = final[1:-1]
    final = ' '.join(final)
    return final
```

Figure 4.10 FUNCTION TO CONVERT MODEL'S OUTPUT TO WORDS



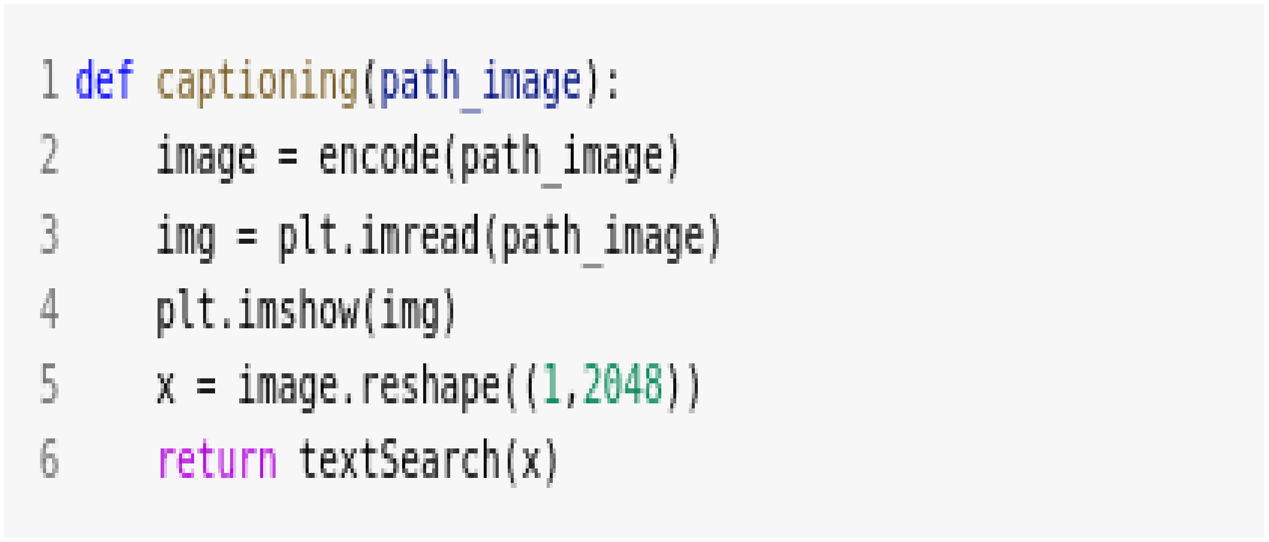
```python
def story_generator(path):
    text = captioning(path)
    return gpt2.generate(sess,
                  length=500,
                  temperature=0.9,
                  prefix=text,
                  nsamples=1,
                  batch_size=1,
                  top_k=20,
                  include_prefix=True,
                  )
```

Figure 4.11 FUNCTION TO CREATE STORY DIRECTLY FROM IMAGES



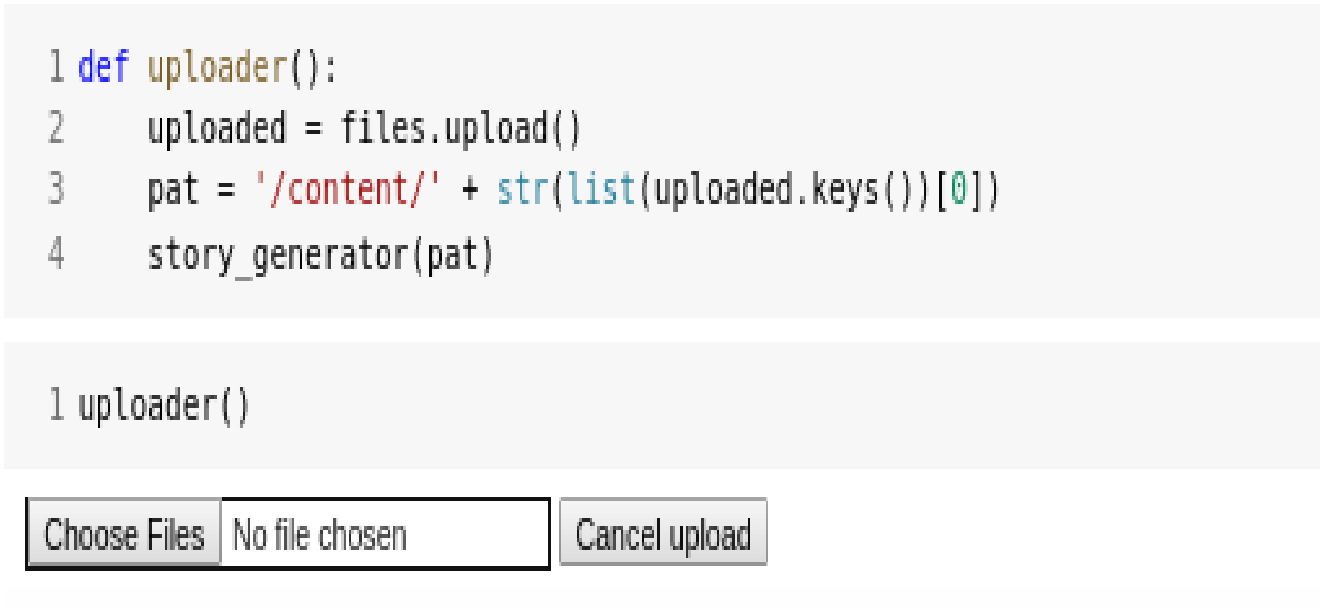
```python
text = gpt2.generate(sess,
              length=500,
              temperature=0.9,
              prefix="Black woman standing in front of stairs",
              nsamples=1,
              batch_size=1,
              top_k=20,
              include_prefix=True,)
```

Figure 4.12 FUNCTION TO CREATE TEXT FROM SEED



```python
def captioning(path_image):
    image = encode(path_image)
    img = plt.imread(path_image)
    plt.imshow(img)
    x = image.reshape((1,2048))
    return textSearch(x)
```

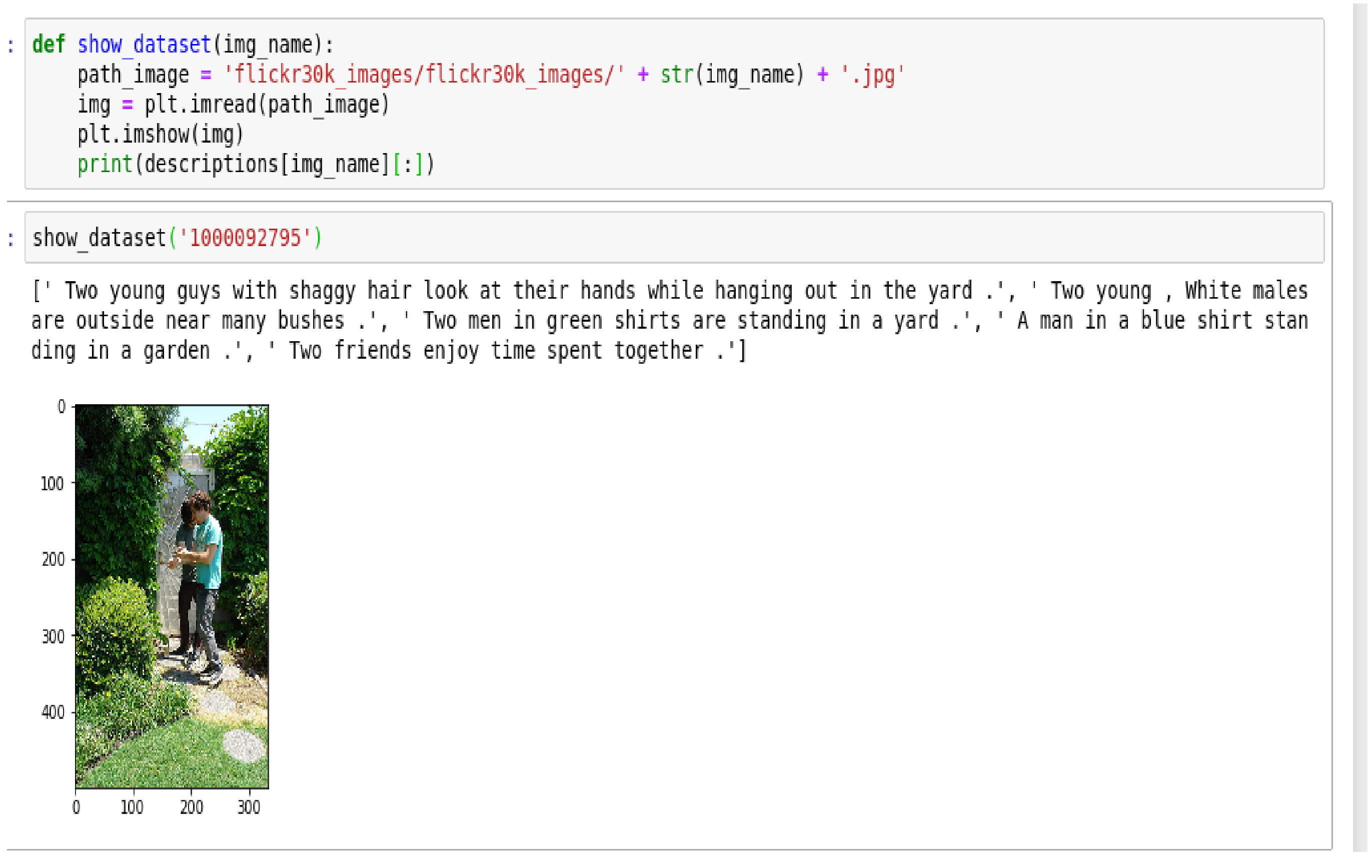
Figure 4.13 FUNCTION TO DIRECTLY SHOW IMAGES AND MODEL GENERATED CAPTIONS



```python
def uploader():
    uploaded = files.upload()
    pat = '/content/' + str(list(uploaded.keys())[0])
    story_generator(pat)
```

```python
uploader()
```

Figure 4.14 GENERATING STORY BY DIRECTLY UPLOADING IMAGE

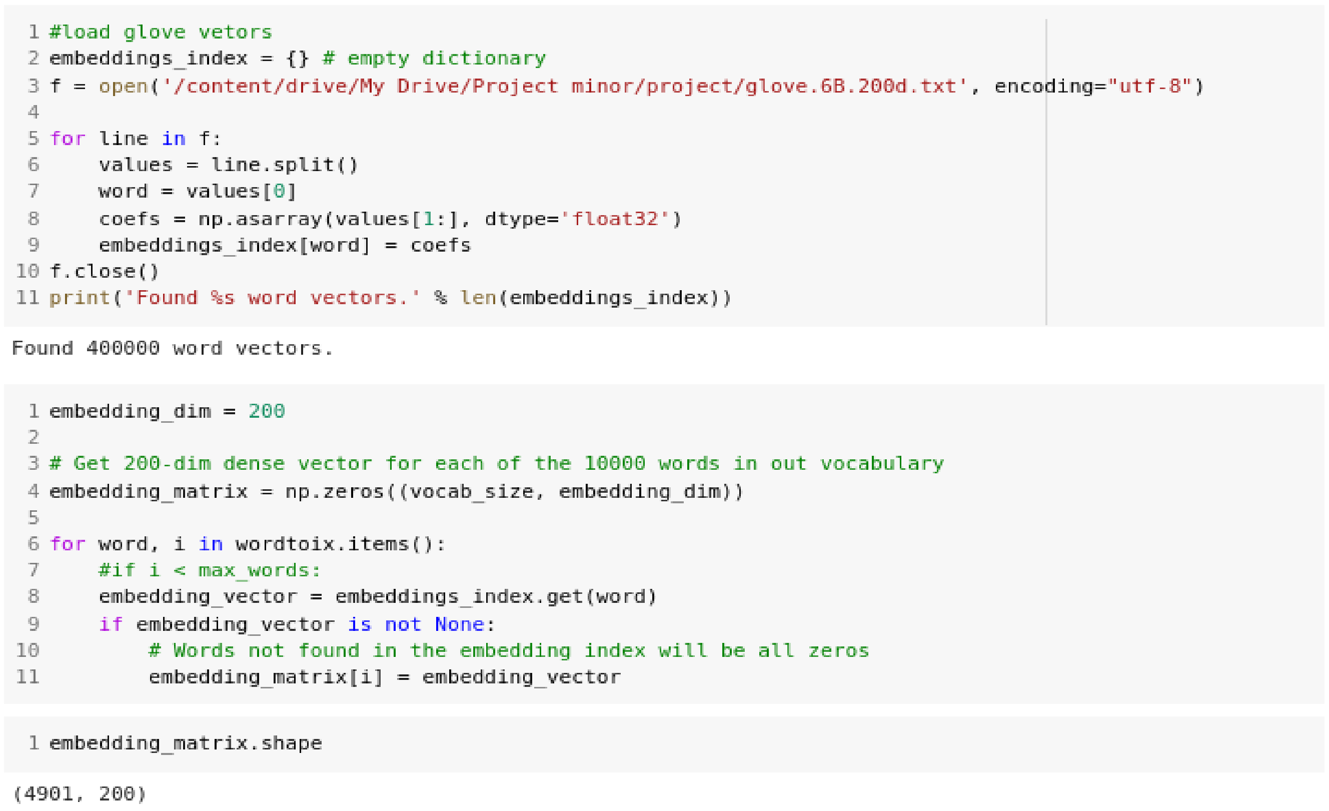


```python
def show_dataset(img_name):
    path_image = 'flickr30k_images/flickr30k_images/' + str(img_name) + '.jpg'
    img = plt.imread(path_image)
    plt.imshow(img)
    print(descriptions[img_name][:])
```

```python
show_dataset('1000092795')
```

```
[' Two young guys with shaggy hair look at their hands while hanging out in the yard .', ' Two young , White males
are outside near many bushes .', ' Two men in green shirts are standing in a yard .', ' A man in a blue shirt stan
ding in a garden .', ' Two friends enjoy time spent together .']
```

Figure 4.15 IMAGE OF FLICKR DATASET



```
#load glove vetors
embeddings_index = {} # empty dictionary
f = open('/content/drive/My Drive/Project minor/project/glove.6B.200d.txt', encoding="utf-8")

for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Found %s word vectors.' % len(embeddings_index))
```

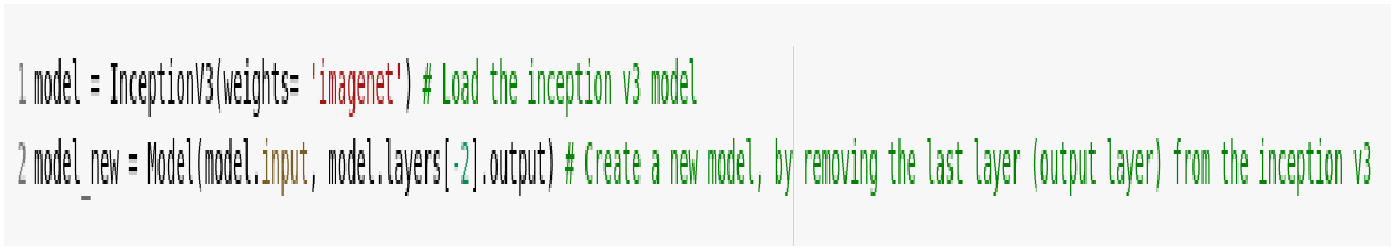
```
Found 400000 word vectors.
```

```
embedding_dim = 200

# Get 200-dim dense vector for each of the 10000 words in out vocabulary
embedding_matrix = np.zeros((vocab_size, embedding_dim))

for word, i in wordtoix.items():
    #if i < max_words:
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in the embedding index will be all zeros
        embedding_matrix[i] = embedding_vector
```

```
embedding_matrix.shape
```

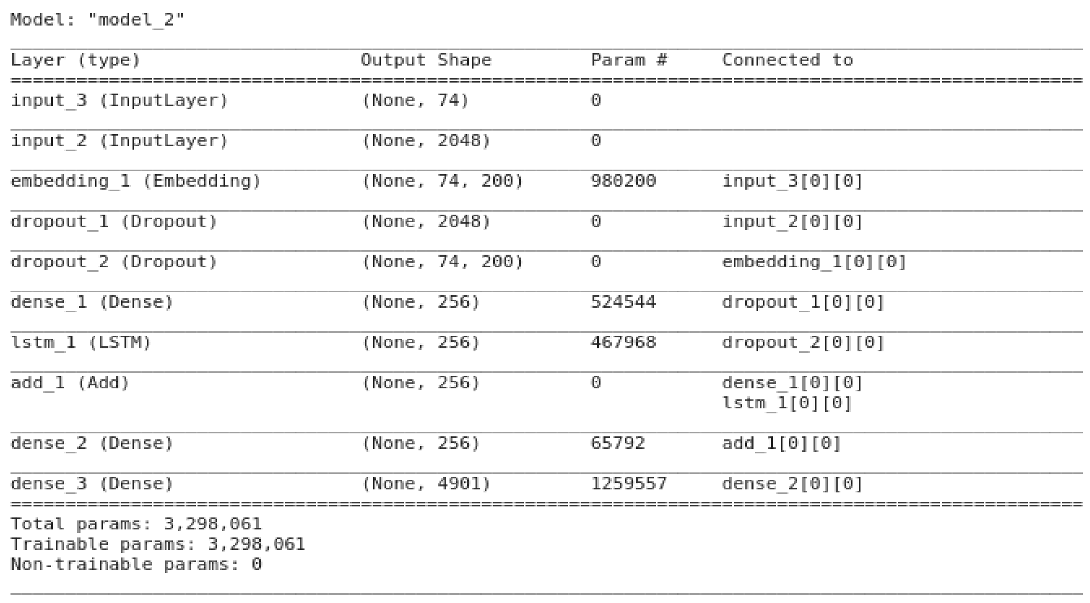
```
(4901, 200)
```

Figure 4.16 LOADING GLOVE VECTOR AND CREATING EMBEDDING MATRIX



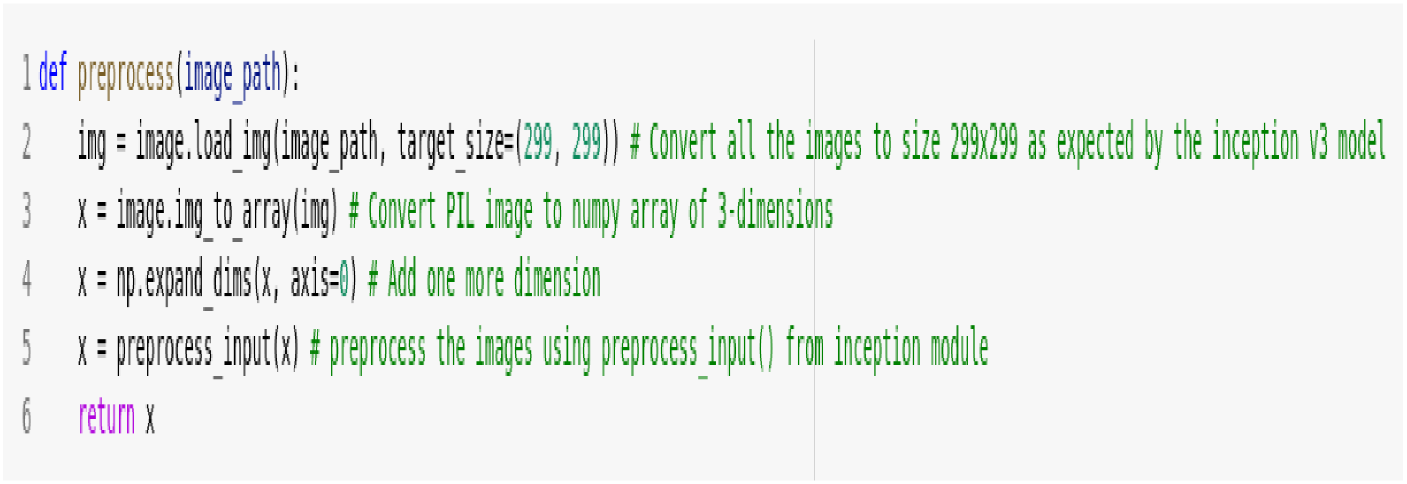
```
model = InceptionV3(weights='imagenet') # Load the inception v3 model
model_new = Model(model.input, model.layers[-2].output) # Create a new model, by removing the last layer (output layer) from the inception v3
```

Figure 4.17 LOADING INCEPTION V3



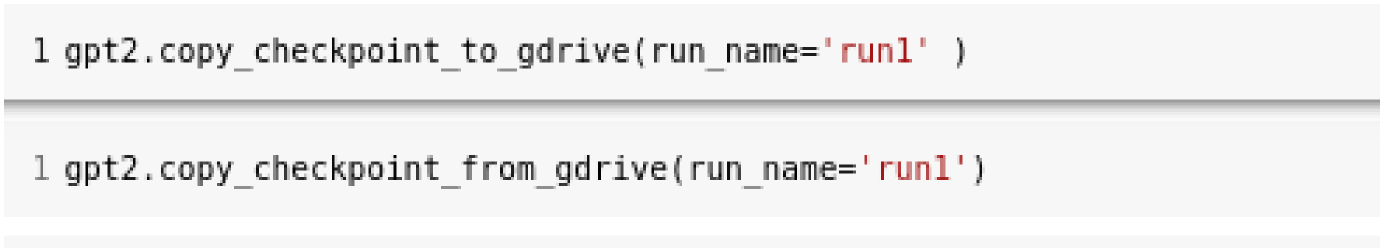
```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            (None, 74)           0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 2048)         0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 74, 200)      980200      input_3[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 2048)         0           input_2[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 74, 200)      0           embedding_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          524544      dropout_1[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 256)          467968      dropout_2[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 256)          0           dense_1[0][0]
                                                                 lstm_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          65792       add_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 4901)         1259557     dense_2[0][0]
==================================================================================================
Total params: 3,298,061
Trainable params: 3,298,061
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 4.18 IMAGE CAPTIONING MODEL SUMMARY



```python
def preprocess(image_path):
    img = image.load_img(image_path, target_size=(299, 299)) # Convert all the images to size 299x299 as expected by the inception v3 model
    x = image.img_to_array(img) # Convert PIL image to numpy array of 3-dimensions
    x = np.expand_dims(x, axis=0) # Add one more dimension
    x = preprocess_input(x) # preprocess the images using preprocess_input() from inception module
    return x
```

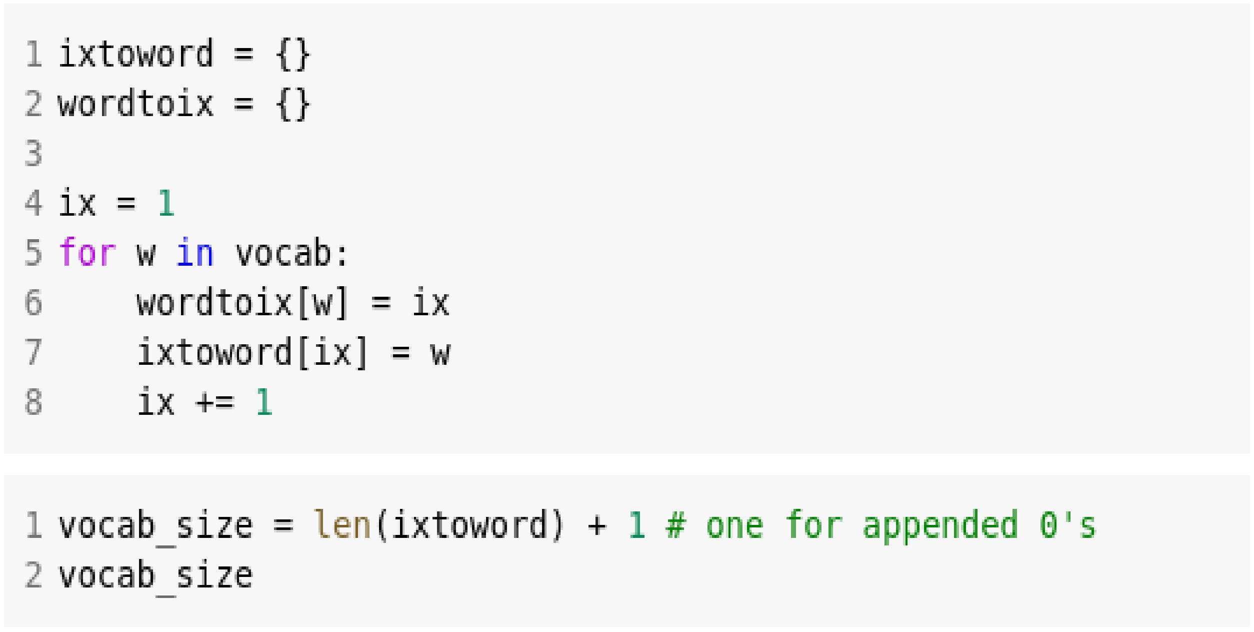
Figure 4.19 PREPROCESSING IMAGES BEFORE INCEPTION V3



```python
gpt2.copy_checkpoint_to_gdrive(run_name='run1' )
```

```python
gpt2.copy_checkpoint_from_gdrive(run_name='run1')
```

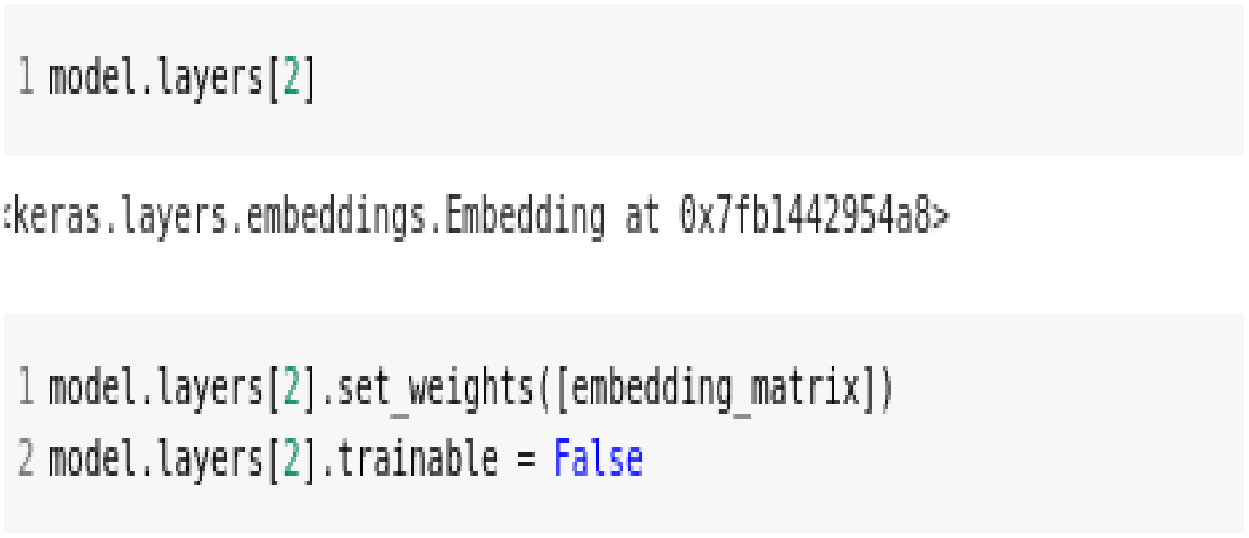
Figure 4.20 SAVING TEXT GENERATOR MODEL TO DRIVE AND LOADING IT



```python
ixtoword = {}
wordtoix = {}

ix = 1
for w in vocab:
    wordtoix[w] = ix
    ixtoword[ix] = w
    ix += 1
```

```python
vocab_size = len(ixtoword) + 1 # one for appended 0's
vocab_size
```

Figure 4.21 SAVING VOCAB AND CONVERTING IT TO INDICES



```python
model.layers[2]
```

```
keras.layers.embeddings.Embedding at 0x7fb1442954a8>
```

```python
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False
```

Figure 4.22 SETTING EMBEDDING LAYER TO UNTRAINABLE

4.2 RESULTS

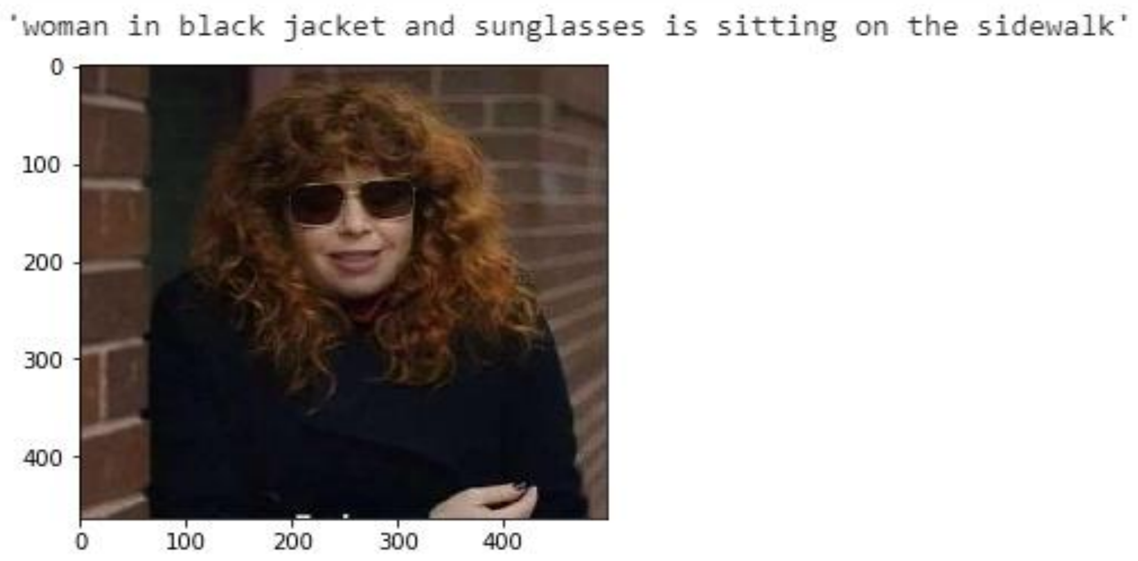

'woman in black jacket and sunglasses is sitting on the sidewalk'
0
100
200
300
400
0
100
200
300
400

Figure 4.23 IMAGE CAPTIONER OUTPUT

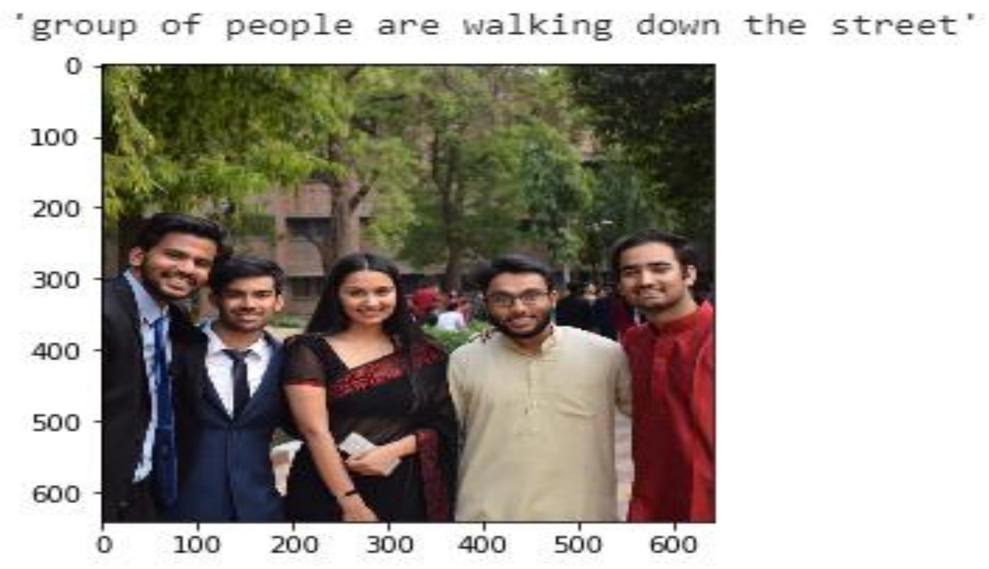
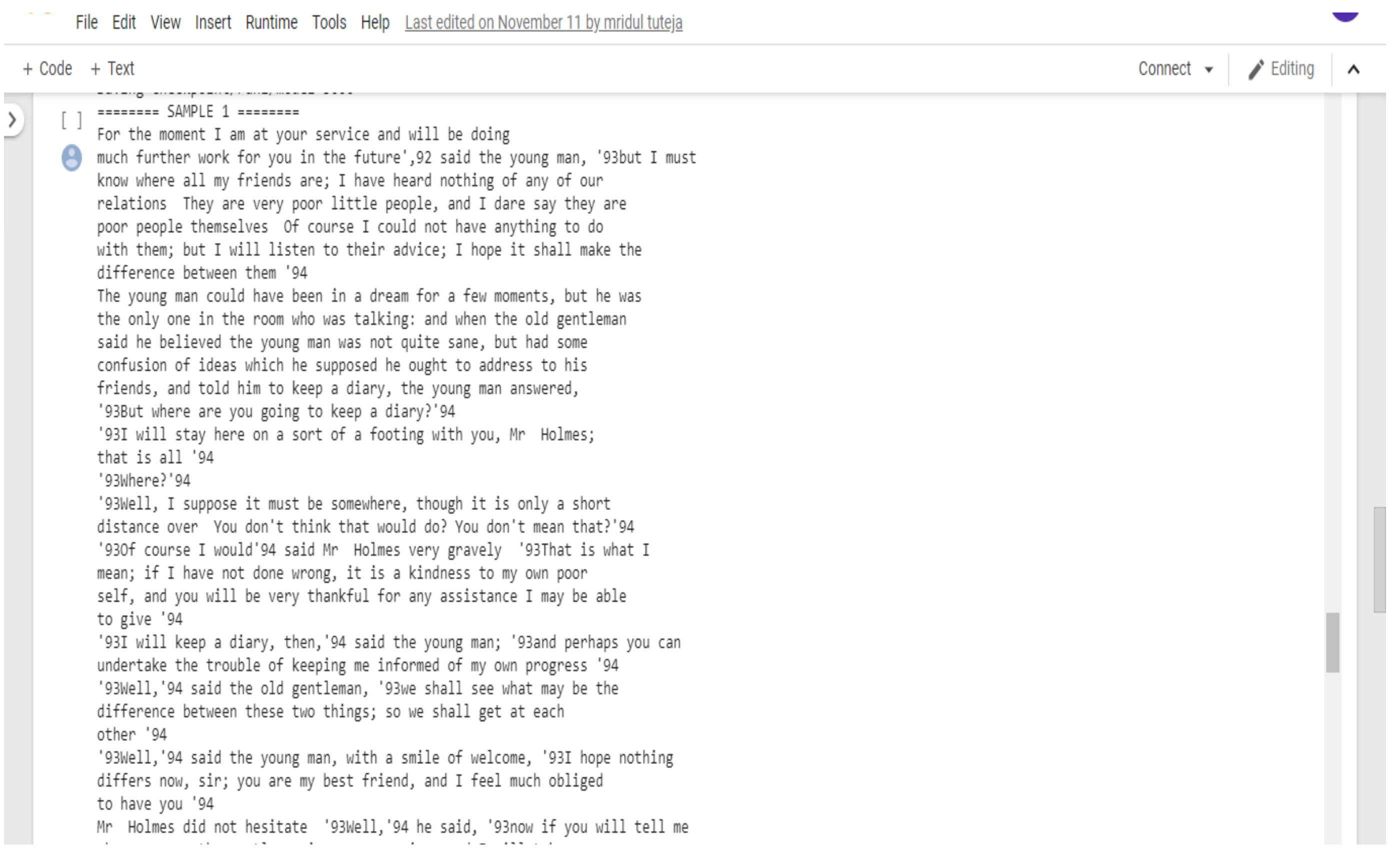


Figure 4.24 IMAGE CAPTIONER OUTPUT


File Edit View Insert Runtime Tools Help Last edited on November 11 by mridul tuteja
+ Code + Text
Connect
Editing

```
======== SAMPLE 1 ========
For the moment I am at your service and will be doing
much further work for you in the future',92 said the young man, '93but I must
know where all my friends are; I have heard nothing of any of our
relations  They are very poor little people, and I dare say they are
poor people themselves  Of course I could not have anything to do
with them; but I will listen to their advice; I hope it shall make the
difference between them '94
The young man could have been in a dream for a few moments, but he was
the only one in the room who was talking: and when the old gentleman
said he believed the young man was not quite sane, but had some
confusion of ideas which he supposed he ought to address to his
friends, and told him to keep a diary, the young man answered,
'93But where are you going to keep a diary?'94
'93I will stay here on a sort of a footing with you, Mr  Holmes;
that is all '94
'93Where?'94
'93Well, I suppose it must be somewhere, though it is only a short
distance over  You don't think that would do? You don't mean that?'94
'93Of course I would'94 said Mr  Holmes very gravely  '93That is what I
mean; if I have not done wrong, it is a kindness to my own poor
self, and you will be very thankful for any assistance I may be able
to give '94
'93I will keep a diary, then,'94 said the young man; '93and perhaps you can
undertake the trouble of keeping me informed of my own progress '94
'93Well,'94 said the old gentleman, '93we shall see what may be the
difference between these two things; so we shall get at each
other '94
'93Well,'94 said the young man, with a smile of welcome, '93I hope nothing
differs now, sir; you are my best friend, and I feel much obliged
to have you '94
Mr  Holmes did not hesitate  '93Well,'94 he said, '93now if you will tell me
```

Figure 4.25 IMAGE GENERATOR OUTPUTS

CHAPTER FIVE : CONCLUSION

Thus after a lot of hard work and study on related topics , we were finally able to create and train a model that takes image as an input , generates a caption from the input image using INCEPTION V3 model which is a pretrained model on IMAGENET dataset.
The output of the captioner model is stored and passed to story generator model.
The generator model is Open AI's GPT2 model which we trained on GUTTENBERG dataset.
By using the self attention algorithm and transfer learning we were able to extract the word embeddings and develop a working model predicting stories from the input image or an input text.

5.1 FUTURE SCOPE

Image story teller can be further improvised and trained so well that it can start predicting perfect captions and stories for pictures on Instagram.
It can help writers to help in predicting and writing further stories from feeding the previous sentence as an input.
It can help in preparing movie scripts if trained on a very large amount of data.

CHAPTER SIX: REFERNCES

- https://www.d2.mpi-inf.mpg.de/sites/default/files/iccv15-neural_qa.pdf
- https://arxiv.org/pdf/1506.06724v1.pdf
- https://hal.inria.fr/hal-01207966/document
- https://www.topbots.com/most-important-ai-computer-vision-research/
- https://www.degruyter.com/downloadpdf/j/ppb.2010.41.issue-2/v10059-010-0008-0/v10059-010-0008-
- https://link.springer.com/article/10.1007/s10545-015-9824-x
- https://arxiv.org/abs/1908.08594
- https://ieeexplore.ieee.org/abstract/document/7984661
- https://ieeexplore.ieee.org/abstract/document/8822095
- https://link.springer.com/chapter/10.1007/978-981-15-1301-5_10
- https://www.kaggle.com/hsankesara/flickr-image-dataset
- https://towardsdatascience.com/is-a-picture-worth-a-thousand-words-d640425c0029
- https://towardsdatascience.com/is-a-picture-worth-a-thousand-words-d640425c0029
- https://medium.com/@igrandic03/ai-writing-stories-based-on-pictures-52b3ddbcd7d
- https://openai.com/blog/better-language-models/
- https://www.analyticsvidhya.com/blog/2019/07/openai-gpt2-text-generator-python/
- https://zpascal.net/cvpr2016/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf
- https://arxiv.org/pdf/1605.07678.pdf
- https://arxiv.org/pdf/1706.03762.pdf
- https://software.intel.com/en-us/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic
- https://en.wikipedia.org/wiki/Keras
- https://www.analyticsvidhya.com/blog/2019/07/openai-gpt2-text-generator-python/
- https://bigthink.com/stephen-johnson/could-we-ever-train-ai-to-become-master-storytellers

- https://medium.com/abraia/first-steps-with-transfer-learning-for-custom-image-classification-with-keras-b941601fcad5

- https://cloud.google.com/tpu/docs/inception-v3-advanced
- https://minimaxir.com/2019/09/howto-gpt2/
- https://medium.com/@igrandic03/ai-writing-stories-based-on-pictures-52b3ddbcd7d
- https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202