IMAGE CAPTIONING WITH PRE-TRAINED MODELS

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

INUKONDA VIDYA GAYATHRI

21B91A1251

Under Supervision of

IIDT ……

DEPARTMENT OF INFORMATION TECHNOLOGY

S.R.K.R. ENGINEERING COLLEGE

(Autonomous)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(Autonomous)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Summer Internship Report titled "**IMAGE CAPTIONING WITH PRE-TRAINED MODELS**" is the bonafide work done by Ms. **INUKONDA VIDYA GAYATHRI** bearing 21B91A1251 at the end of third year second semester at from

… ..... in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

Department Internship Coordinator **Dean -T & P Cell** **Head of the Department**

ABSTRACT

In the modern era, image captioning has become one of the most widely required tools. Moreover, there are inbuilt applications that generate and provide a caption for a certain image, all these things are done with the help of deep neural network models. The process of generating a description of an image is called image captioning. It requires recognizing the important objects, their attributes, and the relationships among the objects in an image. It generates syntactically and semantically correct sentences. In this paper, we present a deep learning model to describe images and generate captions using computer vision and machine translation. This paper aims to detect different objects found in an image, recognize the relationships between those objects and generate captions.

The dataset used is Flickr8k and the programming language used was Python3, and an ML technique called Transfer Learning will be implemented with the help of the Xception model, to demonstrate the proposed experiment. This paper will also elaborate on the functions and structure of the various Neural networks involved. Generating image captions is an important aspect of Computer Vision and Natural language processing. Image caption generators can find applications in Image segmentation as used by Facebook and Google Photos, and even more so, its use can be extended to video frames. They will easily automate the job of a person who has to interpret images.

TABLE OF CONTENTS

1 INTRODUCTION

Whenever an image appears in front of us, our brain can annotate or label it. But what about computers? How can a machine process and label an image with a

highly relevant and accurate caption? It seemed quite impossible a few years back. Still, with the enhancement of Computer Vision and Deep Learning algorithms, the availability of relevant datasets, and AI models, it becomes easier to build a relevant caption generator for an image. Even Caption generation is growing worldwide, and many data annotation firms are earning billions. In this guide, we will build one such annotation tool capable of generating relevant captions for the image with the help of datasets. Basic knowledge of two Deep learning techniques, including LSTM and CNN, is required.

1.1 OBJECTIVE

1. The project aims to work on one of the ways to context a photograph in simple English sentences using Deep Learning (DL).
2. The need to use CNN and LSTM instead of working with RNN

1.2 PROBLEM STATEMENT

In our world, information is considered valuable and some humans face a serious problem regarding visualizing an image. We hence dig into this matter, considering blindness as a major factor, and generate a sentence by allowing users to upload or scan a visual image.

Advantages:

- Recommendations in Editing Applications
- Assistance for visually impaired
- Social Media posts
- Self-Driving cars
- Robotics

● Easy to implement and connect to new data sources

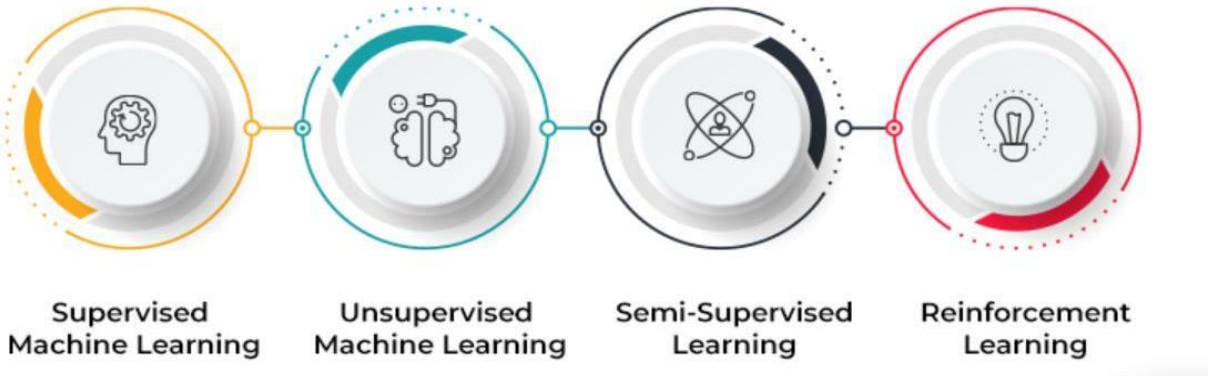
Disadvantages:

● Do not make intuitive feature observations on objects or actions in the image

● Nor do they give an end-to-end mature general model to solve this problem

What is Machine Learning?

A branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data. As intelligence requires knowledge, it is necessary for computers to acquire knowledge. ML uses two types of techniques: **Supervised learning**, which trains a model on known input and output data so that it can predict future outputs, and **Unsupervised learning**, which finds hidden patterns or intrinsic structures in



input data. **semi-Supervised**: combination of above. **Reinforcement**: Learning from the mistakes and the feedback.

2 LITERATURE SURVEY

Literature survey is the most important step in the software development process. Before developing the tool, it is necessary to determine the time factor, economy, and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool, they

programmers need a lot of external support. This support can be obtained from senior programmers, books, or websites. Before building the system, the above considerations are taken into account for developing the proposed system.

The major part of the project development sector considers and fully surveys all the required needs for developing the project. For every project, a Literature survey is the most important sector in the software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, manpower, economy, and company strength.

To improve and tailor the user experience on its products, photos use image classification. Intraclass variation, occlusion, deformation, size variation, perspective variation, and lighting are all frequent issues in computer vision that are represented by the picture classification problem.

Methods that work well for picture classification are likely to work well for other important computer vision tasks like detection, localization, and segmentation as well.

Image captioning is a great illustration of this. Given an image, the image captioning challenge is to generate a sentence description of the image. The picture captioning problem is comparable to the image classification problem in that it expects more detail and has a bigger universe of possibilities. Image classification is used as a black box system in modern picture captioning systems, therefore greater image classification leads to better captioned.

The image captioning problem is intriguing in and of itself because it brings together two significant AI fields: computer vision and natural language processing. An image captioning system demonstrates that it understands both image semantics and natural language.

Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what all the necessary software is needed to proceed with the next step such as developing the tools, and the associated operations.

To construct an image sentence, image classification is a key stage in the object recognition and picture analysis process. The final output of the image categorization phase might be a statement.

To date, a variety of image captioning techniques have been presented. Several studies have been carried out in attempt to determine the best image captioning technique. It's difficult to pick one approach as the finest of them all because the results and accuracy are dependent on a variety of circumstances.

In order to achieve the most accurate results, traditional approaches have been constantly modified as well as new image captioning techniques invented during the previous few decades. Each caption generator technique has its own set of benefits and drawbacks. The focus of the research today is on combining the desired qualities of various techniques in order to boost efficiency.

Many high-level tasks, such as image classification, object detection, and, more recently, semantic segmentation, have recently been proven to obtain outstanding results using convolutional neural networks with many layers. A two-stage technique is frequently used, especially for semantic segmentation. Convolutional networks are trained in this way to offer good local pixel-wise data for the second stage, which is often a more global graphical analysis model.

We will use Long short-term memory (LSTM), which is a subset of RNNs, to tackle the problem of Vanishing Gradient. The main goal of LSTM is to solve the problem of Vanishing Gradients. The unique feature of LSTM is that it can keep data values for long periods, allowing it to address the vanishing gradient problem.

When compared to applying RNN, the results revealed that using a mixture of LSTM generated better outcomes.CNNs employ multilayer convolution to accomplish feature engineering and integrate these features internally, unlike traditional image recognition algorithms. It also employs the pooling and fully connected (FC) layers, as well as SoftMax.

3 AIM AND SCOPE

3.1 Aim

Development of automatically describing an image with more than natural language sentences which leads to faster information transfer.

3.2 Scope

The application of image captioning in deep learning and securing to become common computing power is one of the main factors that led to techniques for analysis of new and diverse digital data spreading information and response toward users for quick understanding of information with just an image.

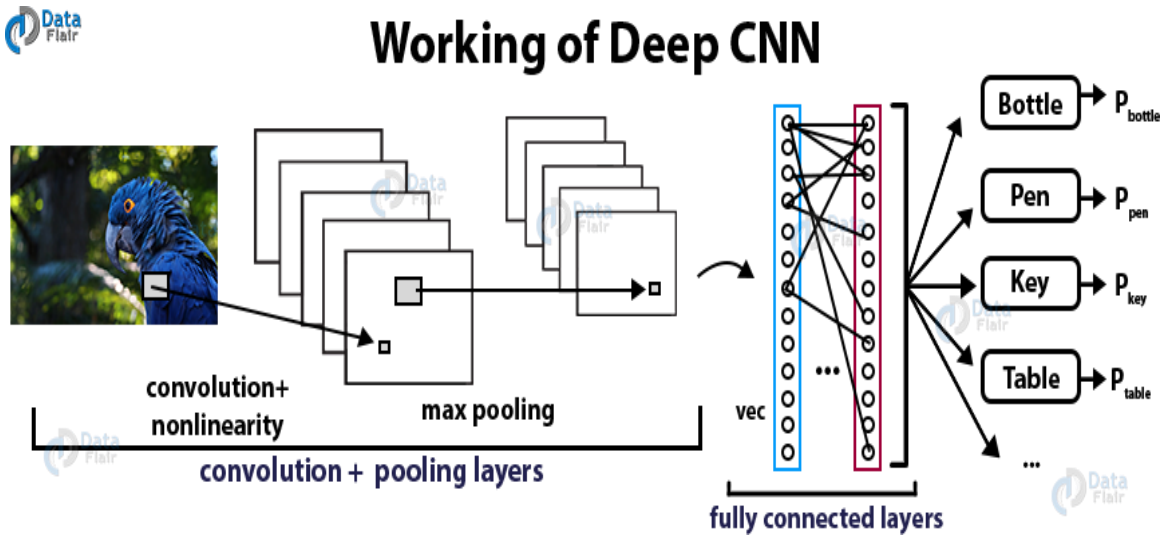
3.3 System Requirements

To be used efficiently, all computer software needs certain hardware components or other software resourced to be present on a computer. These prerequisites are known as computer system requirements and are often used as guidelines as opposed to absolute rules. Most software defines two sets of system requirements. Minimum and recommended. With the increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to exist computer systems than technological advancements.

4 OVERVIEW

4.1 WHAT IS CNN?

Convolutional Neural networks are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images. It scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.



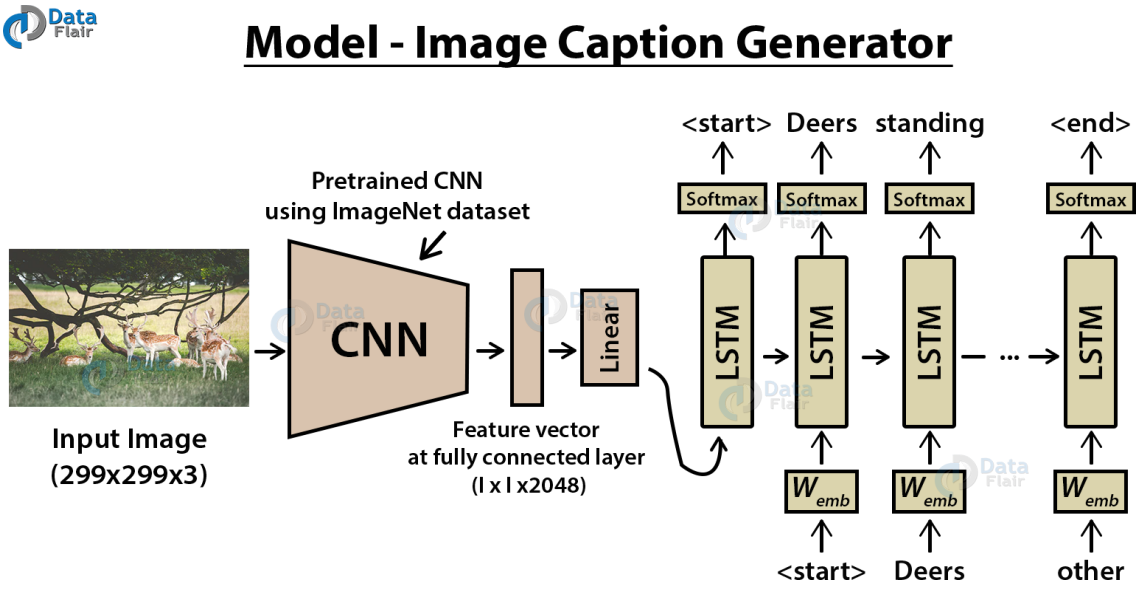
4.2 WHAT IS LSTM?

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

4.3 IMAGE CAPTION GENERATOR MODEL

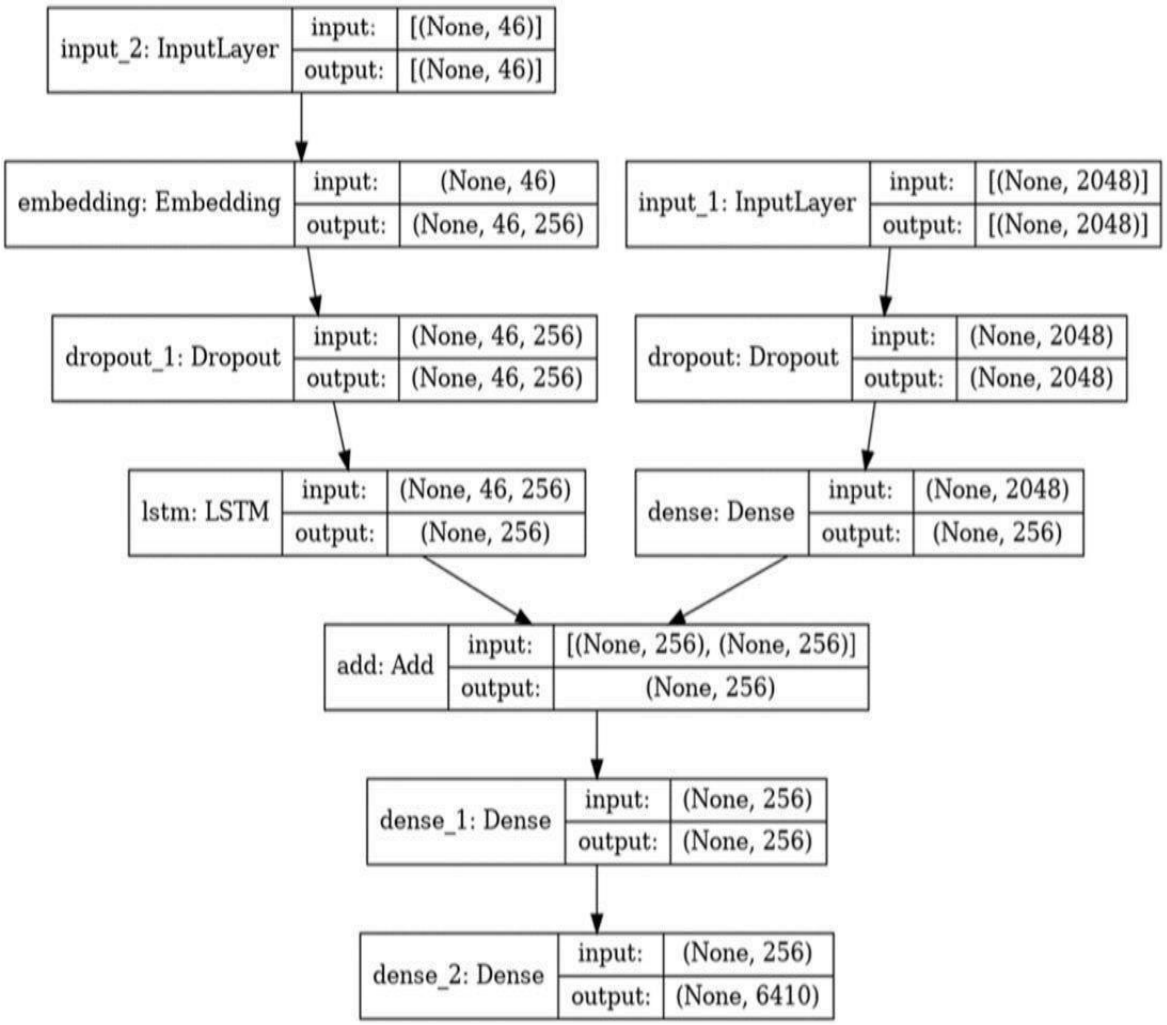
So, to make our image caption generator model, we will be merging these architectures. It is also called a CNN-RNN model.

- CNN is used for extracting features from the image. We will use the pretrained model Xception.
- LSTM will use the information from CNN to help generate a description of the image.



4.4 METHODOLOGY

- Import Libraries
- Upload flickr Dataset. (Data Preprocessing) ● Apply CNN to identify the objects in the image.
- Preprocess and tokenize the captions.
- Use LSTM to predict the next word of the sentence.
- Make a Data Generator
- View Images with caption.



WORKFLOW DIAGRAM

5 DATASET

We have obtained our dataset from kaggle

Dataset link: Flickr 8k Dataset (kaggle.com)

Target: To generate captions for the images according to user preferences.

6 RESULTS

The result of this program is going to be a user being allowed to generate a cap tion for a visual image using Deep Learning, NLP, and Computer Vision.

Representation Of What Image Captioning Is

7 IMPLEMENTATION

Importing the dependencies

[ ]: ln

```
from keras.preprocessing.image import load_img from
keras.preprocessing.image import img_to_array from
keras.applications.vgg16 import preprocess_input from
keras.applications.vgg16 import VGG16 from keras.models
import  Model ,load_model from
```

tensorflow.keras.preprocessing.text import Tokenizer from

keras.preprocessing.sequence import pad_sequences from

keras.models import Model from keras.utils import

to_categorical, plot_model

from keras.layers import Input , Dense , LSTM , Embedding , Dropout , add ,BatchNormalization from PIL

import Image import

matplotlib.pyplot as plt

import os import tqdm import

numpy as np

Defining the model

---

ln [
]:

```
model=VGG16()
model=Model(inputs=model.inputs,outputs=model.layers[-2].output)   model.summary()
```

OUTPUT :

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0

block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0

```
│ fc1 (Dense)              │ (None, 4096)           │       102,764,544 │
├──────────────────────────┼────────────────────────┼───────────────────┤
│ fc2 (Dense)              │ (None, 4096)           │        16,781,312 │
└──────────────────────────┴────────────────────────┴───────────────────┘
```

Total params: 134,260,544 (512.16 MB)
Trainable params: 134,260,544 (512.16 MB)
Non-trainable params: 0 (0.00 B)

Loading images

```
CRNT_DIR="D:/"
dir=os.path.join(CRNT_DIR,"ImgData/")
```

```
def ExtractImgData(dir):
  features={}

  for image_name in tqdm.tqdm(os.listdir(dir)):
imgpath=dir + image_name
    img=load_img(imgpath,target_size=(224,224))
img=img_to_array(img)

    img=img.reshape((1, img.shape[0], img.shape[1], img.shape[2]))
img=preprocess_input(img)

    feature=model.predict(img,verbose=0)

    img_id = image_name.split('.')[0]

    features[img_id] = feature

  return features
```

```
ImageData=ExtractImgData(dir)
```

Saving features

```
[ ] import pickle

pickle.dump(ImageData, open('Imagefeatures.pkl', 'wb'))
```

Caption data

```
ln[ ]:  with open('captions/captions.txt','r') as txt:
 next(txt)
 imgcaptions=txt.read()


ln[ ]:
txtMapping={} for line in
imgcaptions.split('\n'):
  tokens=line.split(',')
  imgId,caption=tokens[0],tokens[1:]
  imgId=imgId.split('.')[0]
  caption="".join(caption)

 if imgId not in txtMapping:
txtMapping[imgId] = []
  txtMapping[imgId].append(caption)

In [ ]: def
clean_captions(mapping):
   for key,captions in mapping.items():
for i in range(len(captions)):
caption=captions[i]

        caption=caption.lower()          caption=caption.replace('[^A-
Za-z]','')          caption=caption.replace('\s+',' ')
        caption= 'startseq '+" ".join([word for word in caption.split() if len(word)>1])+'
endseq'
        captions[i]=caption  In
[ ]:
txtMapping['1000268201_693b08cb0e']
```

OUTPUT :

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']

In [ ]:
clean_captions(txtMapping)

In [ ]:
```

```
txtMapping['1000268201_693b08cb0e']
```

OUTPUT :

```
['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
 'startseq girl going into wooden building endseq',
 'startseq little girl climbing into wooden playhouse endseq',
 'startseq little girl climbing the stairs to her playhouse endseq',
 'startseq little girl in pink dress going into wooden cabin endseq']
```

In [ ]:

```
captions=[] for key in
txtMapping:     for caption in
txtMapping[key]:

      captions.append(caption)
```

In [ ]:

```
captions[:5]
```

OUTPUT :

```
['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
 'startseq girl going into wooden building endseq',
 'startseq little girl climbing into wooden playhouse endseq',
 'startseq little girl climbing the stairs to her playhouse endseq',
 'startseq little girl in pink dress going into wooden cabin endseq']
```

In [ ]:

```
tokenizer=Tokenizer() tokenizer.fit_on_texts(captions)
vacob=len(tokenizer.word_index)+1
```

In [ ]:

```
maxlen=max(len(caption.split()) for caption in captions) maxlen
```

OUTPUT :

35

In [ ]:

```
pickle.dump(open('tokenizer.pkl','wb'))
```

In [ ]:

```
image_ids = list(txtMapping.keys())
split = int(len(image_ids) * 0.90) train
= image_ids[:split]
test = image_ids[split:]
```

```
In [ ]: def
DataGenrator(data,mapping,imgFeatures,tokenizer,maxlen,vacob,batchsize):
x1,x2,y=list(),list(),list()      n=0     while 1:        for keys in data:
        n+=1
        captions=mapping[keys]

        for caption in captions:

            seq=tokenizer.texts_to_sequences([caption])[0]
for i in range(1,len(seq)):
inseq,outseq=seq[:i],seq[i]

                inseq=pad_sequences([inseq],maxlen=maxlen)[0]
outseq = to_categorical([outseq],num_classes=vacob)[0]

                x1.append(imgFeatures[keys][0])
x2.append(inseq)
                y.append(outseq)

        if n == batchsize:
            x1,x2,y=np.array(x1),np.array(x2),np.array(y)
yield (x1,x2),y              x1,x2,y=list(),list(),list()
n=0
```

Model Creation

```
#Img Feature Layer input1=Input(shape=(4096,))
feature1=Dropout(0.4)(input1)
feature2=Dense(256,activation='relu')(feature1)
#Seq Feature Layer
input2=Input(shape=(maxlen,))
seq1=Embedding(vacob,256,mask_zero=True)(input2)
seq2=Dropout(0.4)(seq1) seq3=LSTM(256)(seq2)

#Decoder decoder1=add([feature2,seq3])
decoder2=Dense(256,activation='relu')(decoder1)
outputs=Dense(vacob,activation='softmax')(decoder2)
```

```
model=Model(inputs=[input1,input2],outputs=outputs)
model.compile(loss='categorical_crossentropy',optimizer='adam')

# plot_model(model,show_shapes=True)

model.summary()
```

Model: "functional"

```
┌─────────────────────┬───────────────────┬────────────┬──────────────────┐
│ Layer (type)        │ Output Shape      │    Param # │ Connected to     │
├─────────────────────┼───────────────────┼────────────┼──────────────────┤
│ input_layer_1       │ (None, 35)        │          0 │ -                │
│ (InputLayer)        │                   │            │                  │
├─────────────────────┼───────────────────┼────────────┼──────────────────┤
│ input_layer         │ (None, 4096)      │          0 │ -                │
│ (InputLayer)        │                   │            │                  │
├─────────────────────┼───────────────────┼────────────┼──────────────────┤
│ embedding           │ (None, 35, 256)   │  2,172,160 │ input_layer_1[0]… │
│ (Embedding)         │                   │            │                  │
├─────────────────────┼───────────────────┼────────────┼──────────────────┤
│ dropout (Dropout)   │ (None, 4096)      │          0 │ input_layer[0][0] │
├─────────────────────┼───────────────────┼────────────┼──────────────────┤
```

dropout_1 (Dropout)	(None, 35, 256)	0	embedding[0][0]
not_equal (NotEqual)	(None, 35)	0	input_layer_1[0]…
dense (Dense)	(None, 256)	1,048,832	dropout[0][0]
lstm (LSTM)	(None, 256)	525,312	dropout_1[0][0], not_equal[0][0]
add (Add)	(None, 256)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	65,792	add[0][0]
dense_2 (Dense)	(None, 8485)	2,180,645	dense_1[0][0]

Total params: 5,992,741 (22.86 MB)

Trainable params: 5,992,741 (22.86 MB)

Non-trainable params: 0 (0.00 B)

Loading image features

ln[ ] :

```
import pickle imgFeatures=pickle.load(open('Imagefeatures.pkl','rb'))
```

In [ ]:

```
imgFeatures['1001773457_577c3a7d70']
```

OUTPUT :

```
array([[0.       , 0.       , 0.49414796, ..., 0.       , 0.       ,
       0.       ]], dtype=float32)
```

In [ ]:

```
epochs=15 batchsize=64
steps=len(train)//batchsize
for i in range(epochs):

Traingenerator=DataGenrator(train,txtMapping,imgFeatures,tokenizer,maxlen,vacob,ba
tchsize)

   print("Epochs ",i," out of ",epochs)
   history=model.fit(Traingenerator,epochs=1,steps_per_epoch=steps,verbose=1)
```

OUTPUT :

Epochs 0 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **355s** 3s/step - loss: 5.0057

Epochs 1 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **271s** 2s/step - loss: 4.0792

Epochs 2 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **309s** 3s/step - loss: 3.6608

Epochs 3 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **396s** 3s/step - loss: 3.4086

Epochs 4 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **285s** 3s/step - loss: 3.2199

Epochs 5 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **449s** 4s/step - loss: 3.0754

Epochs 6 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **327s** 3s/step - loss: 2.9605

Epochs 7 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **266s** 2s/step - loss: 2.8580

Epochs 8 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **1102s** 10s/step - loss: 2.7775

Epochs 9 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **333s** 3s/step - loss: 2.7089

Epochs 10 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **377s** 3s/step - loss: 2.6417

Epochs 11 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **292s** 3s/step - loss: 2.5812

Epochs 12 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **351s** 3s/step - loss: 2.5280

Epochs 13 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **427s** 4s/step - loss: 2.4823

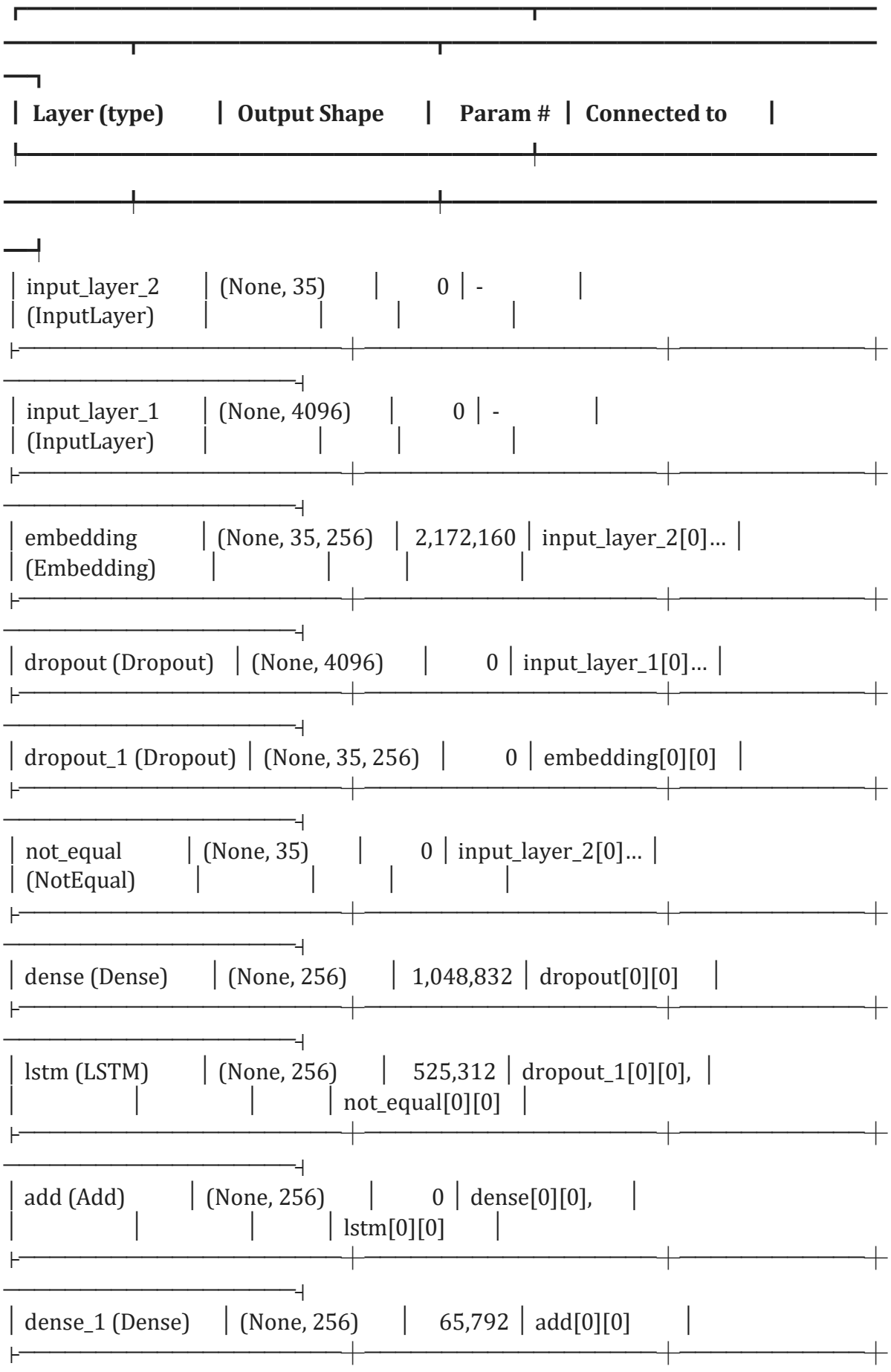
Epochs 14 out of 15

113/113 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **343s** 3s/step - loss: 2.4372

In [28]:
model.summary()

OUTPUT :

Model: "functional_1"



Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 35)	0	-
input_layer_1 (InputLayer)	(None, 4096)	0	-
embedding (Embedding)	(None, 35, 256)	2,172,160	input_layer_2[0]...
dropout (Dropout)	(None, 4096)	0	input_layer_1[0]...
dropout_1 (Dropout)	(None, 35, 256)	0	embedding[0][0]
not_equal (NotEqual)	(None, 35)	0	input_layer_2[0]...
dense (Dense)	(None, 256)	1,048,832	dropout[0][0]
lstm (LSTM)	(None, 256)	525,312	dropout_1[0][0], not_equal[0][0]
add (Add)	(None, 256)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	65,792	add[0][0]

```
──────────────────────────┤
│ dense_2 (Dense)     │ (None, 8485)      │ 2,180,645 │ dense_1[0][0]     │
└─────────────────────┴──────────────────────┴──────────┴──
──────────────────────┘
```

Total params: 17,978,225 (68.58 MB)
Trainable params: 5,992,741 (22.86 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 11,985,484 (45.72 MB)

```
In [ ]:
model.save('./Models/my_model.keras')

In [ ]:
model_json = model.to_json() with
open("./Models/model.json", "w") as json_file:
   json_file.write(model_json)

In [ ]:
model=load_model('./Models/my_model.keras')
Generating Captions

In [ ]: def
idx_to_word(integer):
   for word, index in tokenizer.word_index.items():
if index == integer:
         return word
   return None

In [ ]:
linkcode def
GenarateCaptions(image,maxlen):
   inputtxt='startseq'
pred=[]     for i in
range(maxlen):

      inputSeq=tokenizer.texts_to_sequences([inputtxt])[0]
inputSeq=pad_sequences([inputSeq],maxlen=maxlen)
predCaption=model.predict([image,inputSeq],verbose=0)
predCaption=np.argmax(predCaption)
word=idx_to_word(predCaption)        if word is None:
         break
inputtxt+=' '+word
if word == 'endseq':
break

   return inputtxt
```

Defining Model

```
ln[ ]:
 inputmodel=VGG16()
 inputmodel=Model(inputs=inputmodel.inputs,outputs=inputmodel.layers[-2].output)

 def ExtractInputImg(imgpath):
 feature=[]

    inImg=load_img(imgpath,target_size=(224,224))
 inImg=img_to_array(inImg)
    inImg=inImg.reshape((1, inImg.shape[0], inImg.shape[1], inImg.shape[2]))
 inImg=preprocess_input(inImg)

    feature.append(inputmodel.predict(inImg,verbose=0))

    return feature

 In [ ]: def
 CleanCaption(caption):

    caption=[word for word in caption.split() if word.lower() not in ['startseq','endseq']]
 caption=' '.join(caption)

    c=[]     for w in
 caption.split():         if w
 not in c:
          c.append(w)
 caption=" ".join(c)
    return caption

 In [ ]:
 linkcode
 from nltk.translate.bleu_score import corpus_bleu actual,
 predicted = list(), list()

 for key in tqdm.tqdm(test):

    captions = txtMapping[key]
 y_pred=GenarateCaptions(imgFeatures[key],maxlen)      actual_captions
 = [caption.split() for caption in captions]
 actual.append(actual_captions)      predicted.append(y_pred)
 print("BLEU-1: %f" % corpus_bleu(actual, predicted, weights=(1.0, 0,
```

```
0, 0))) print("BLEU-2: %f" % corpus_bleu(actual, predicted,
weights=(0.5, 0.5, 0, 0)))
```

Input Image

---

ln[ ]:

```
InputImg='./ImgData/3489774350_a94e6c7bfc.jpg'
```

---

ln[ ]:

```
Image.open(InputImg)
```

```
inputFeature=ExtractInputImg(InputImg)
```

```
caption=GenarateCaptions(inputFeature,maxlen)
```

```
caption=CleanCaption(caption)
```

```
image=Image.open(InputImg)  plt.imshow(image)

plt.title(caption) plt.axis(False) plt.show()

print(caption)
```

8 SUMMARY AND CONCLUSION

8.1 Summary

We combined all components of the image caption generation problem in this overview, addressed the model framework proposed in recent years to handle the description task, concentrated on the algorithmic essence of various attention methods, and summarized how the attention mechanism is implemented. The huge datasets and evaluation criteria that are regularly

utilized in practice are summarized. Despite the fact that image captioning can be used for image retrieval [92], video caption [93, 94], and video movement [95], and a wide range of image caption systems are currently available, experimental results suggest that this task still requires higher performance systems and improvement.

8.2 Conclusion

The CNN-LSTM model was created to automatically generate captions for the input images. This concept can be used in a wide range of situations. We learned about the CNN model, and LSTM models, and how to overcome previous limitations in the field of graphical image captioning by building a CNN-LSTM model capable of scanning and extracting information from any input image and transforming it into a single line sentence in natural language English. The algorithm attention and how the attention mechanism is used were the main topics of discussion. I was able to successfully create a model that is a major improvement above the earlier image caption generator.