PROJECT REPORT

ON

Image Caption Generator

AT

ज्ञानेन प्रकाशते जगत्

INDUS UNIVERSITY

In partial fulfilment of the requirement for the degree of Bachelor of Technology inComputer Engineering

PREPARED BY

DHRUVI KAKADIA (IU1941220021)

HETANSHI DARBAR(1941220008)

UNDER THE GUIDANCE OF

Prof. Sejal Thakkar

Assistant Professor,
Dept. of Computer Engineering,
Indus University, Ahmedabad

Index

ABSTRACT

Image caption generator is a process of recognizing the context of an image and annotating it with relevant captions using deep learning, and computer vision. It includes the labeling of an image with English keywords with the help of datasets provided during model training. Imagenet dataset is used to train the CNN model called Xception. Xception is responsible for image feature extraction. These extracted features will be fed to the LSTM model which in turn generates the image caption. Even Caption generation is becoming a growing business in the world, and many data annotation firms are earning billions from this. In this guide, we are going to build one such annotation tool which is capable of generating very relevant captions for the image with the help of datasets. Basic knowledge of two techniques of Deep learning including LSTM(a type of Recurrent Neural Network) and Convolutional Neural Networks(CNN) is required for the same.

LIST OF FIGURES

ABBREVIATIONS

Following are the abbreviations that are used in this document:

- **CNN- Convolutional Neural Network**
- **LSTM- Long Short Term Memory**
- **NLP- Natural Language Processing**
- **CPU- Control Processing Unit**
- **PIL- Python Image Library**
- **RNN- Recurrent Neural Network**

CHAPTER 1
INTRODUCTION

- **PROJECT SUMMARY**
- **PROJECT SCOPE**
- **PROJECT OBJECTIVES**
- **TECHNOLOGY OVERVIEW**

1.1 PROJECT SUMMARY

n image and your brain can easily tell what the image is about, but can a computer tell what the image is representing? Computer vision researchers worked on this a lot and they considered it impossible until now! With the advancement in Deep learning techniques, availability of huge datasets and computer power, we can build models that can generate captions for an image.

This is what we are going to implement in this Python based project where we will use deep learning techniques of Convolutional Neural Networks and a type of Recurrent Neural Network (LSTM) together.

1.2 PROJECT SCOPE

The system will be created with state-of-the-art technology hence it will be viable or can be used in the future without requiring much remodelling in future. The system will be cost- effective economical and hassle-free to handle and will be easy to maintain. Since the system will be having good and efficient technology it is possible to say it would be easily adaptable to the changing environment, secondly, since it is software-based only the charges of the hardware are required to store the database, hence the only cost of hardware is incurred.

1.3 PROJECT OBJECTIVES

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

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

1.4 TECHNOLOGY & LITERATURE OVERVIEW

To create and understand the whole system we require to get the basic knowledge of these technologies:

1.4.1 PYTHON

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

1.4.2 MICROSOFT WORD 2021

Microsoft Word or MS Word (often called Word) is a graphical word processing program that users can type with. It is made by the computer company Microsoft. Its purpose is to allow users to type and save documents.

CHAPTER 2
Requirements

- **DataSet**
- **Python Libraries**

2.1 The Dataset of Project

For the image caption generator, we will be using the Flickr_8K dataset. There are also other big datasets like Flickr_30K and MSCOCO dataset but it can take weeks just to train the network so we will be using a small Flickr8k dataset. The advantage of a huge dataset is that we can build better models.

Thanks to Jason Brownlee for providing a direct link to download the dataset (Size: 1GB).

- Flicker8k_Dataset
- Flickr_8k_text

The Flickr_8k_text folder contains file Flickr8k.token which is the main file of our dataset that contains image name and their respective captions separated by newline("\n").

2.2 The Libraries used in Project

Python have large number of libraries from which few have been and listed below with its brief details:

2.2.1 TENSORFLOW

TensorFlow is an open-source library for fast numerical computing. It was created and is maintained by Google and was released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least of which is RankBrain in Google search and the fun DeepDream project. It can run on single CPU systems and GPUs, as well as mobile devices and large-scale distributed systems of hundreds of machines.

2.2.2 KERAS

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. It cannot handle low-level computations, so it makes use of the Backend library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

2.2.3 PILLOW

Python Pillow module is built on top of PIL (Python Image Library). It is the essential modules for image processing in Python. But it is not supported by Python 3. But, we can use this module with the Python 3.x versions as PIL. It supports the variability of images such as jpeg, png, bmp, gif, ppm, and tiff. We can do anything on the digital images using the pillow module. In the upcoming section, we will learn various operations on the images such as filtering images, Creating thumbnail, merging images, cropping images, blur an image, resizing an image, creating a water mark and many other operations.

2.2.4 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

2.2.5 TQDM

tqdm is a Python library which is used for creating a progress bar. It is typically used to display the progress of a lengthy operation and it provides a visual cue that processing is underway.

A tqdm progress bar gives information such as The Number and Percentage of iterations completed out of the total number of iterations, Total Elapsed Time Estimated Time to Complete the loop, and The speed of the loop in iterations per second (or it/s).

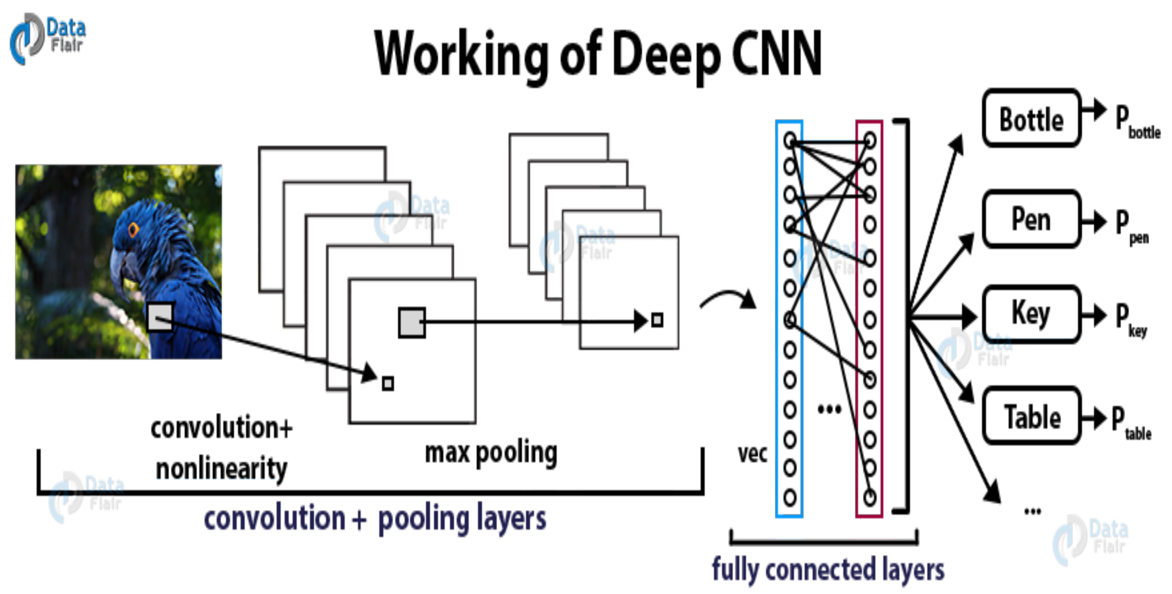
CHAPTER 3
Model Formation

- ⊙ **CNN**
- ⊙ **LSTM**
- ⊙ **Image Caption Generator Model**

3.1 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.

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

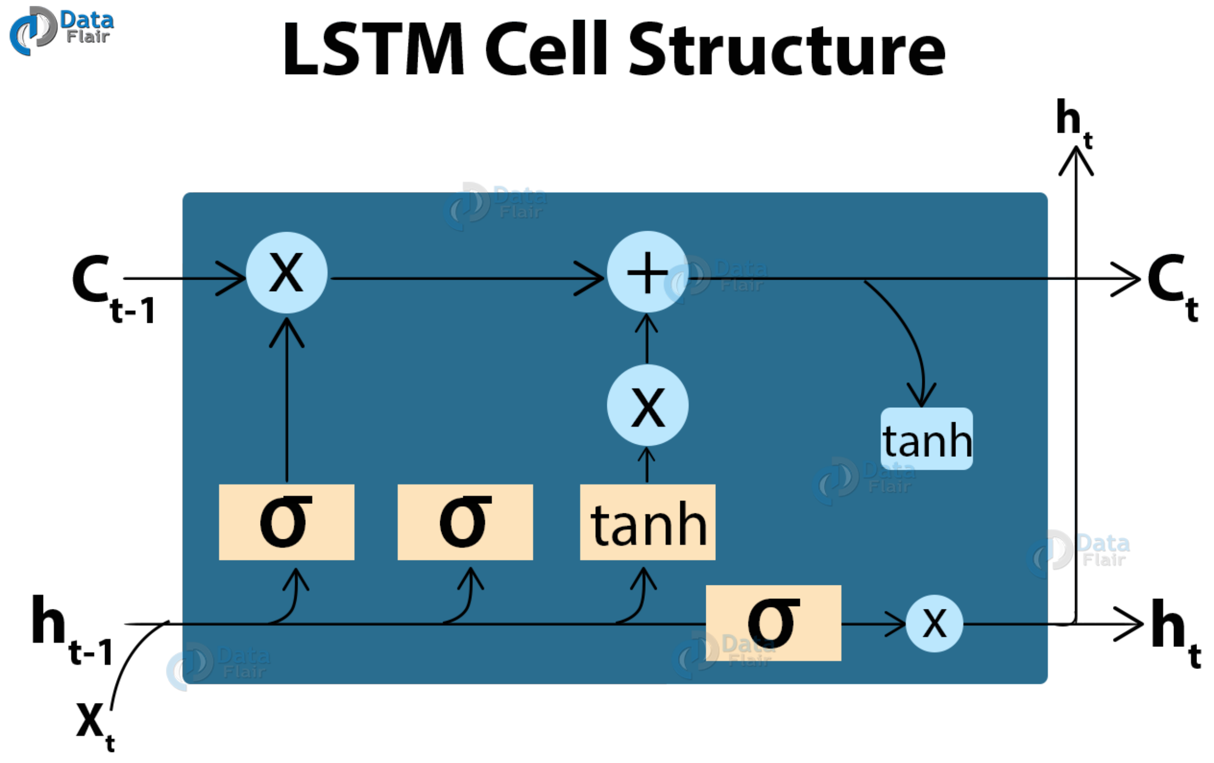


It scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.

3.2 LSTM

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

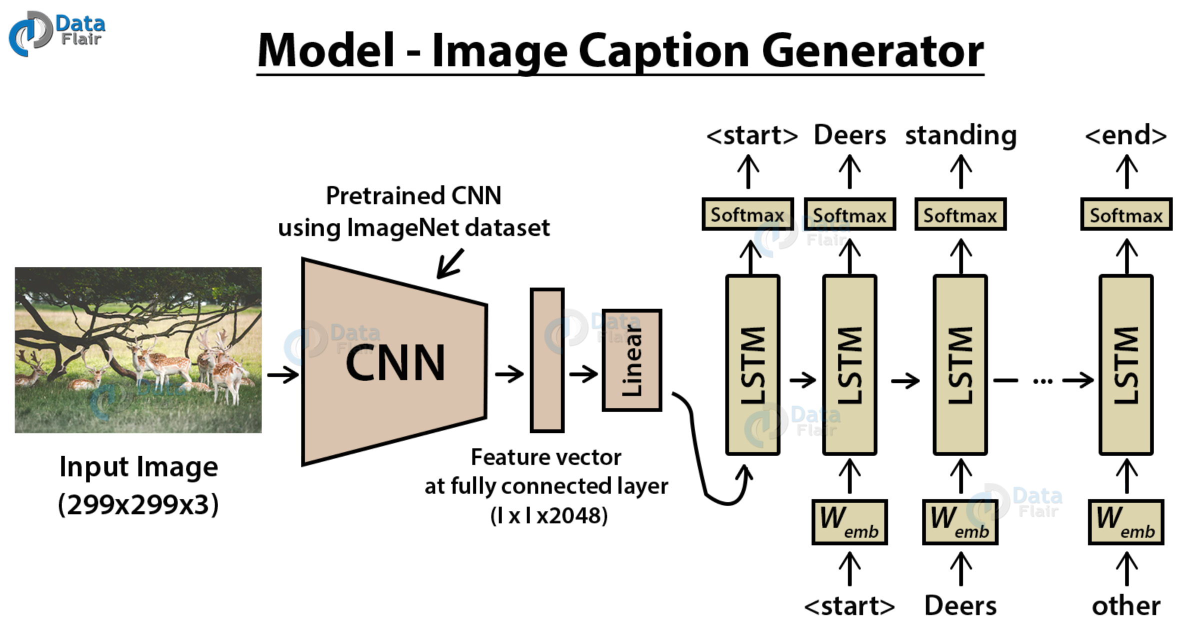
This is what an LSTM cell looks like –



3.3 Image Caption Generator

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

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



Model - Image Caption Generator
Pretrained CNN using ImageNet dataset
CNN
Linear
Input Image (299x299x3)
Feature vector at fully connected layer (I x I x2048)
<start>
Deers
standing
<end>
Softmax
LSTM
W emb
<start>
Deers
other

CHAPTER 4
File Structure

- Downloaded
- Create while Making

4.1 Downloaded from Internet

The below files are downloaded from intersect for purpose as dataset:

- **Flicker8k_Dataset** – Dataset folder which contains 8091 images.
- **Flickr_8k_text** – Dataset folder which contains text files and captions of images

4.2 Created while Making

The below files will be created by us while making the project.

- **Models** – It will contain our trained models.
- **Descriptions.txt** – This text file contains all image names and their captions after preprocessing.
- **Features.p** – Pickle object that contains an image and their feature vector extracted from the Xception pre-trained CNN model.
- **Tokenizer.p** – Contains tokens mapped with an index value.
- **Model.png** – Visual representation of dimensions of our project.
- **Testing_caption_generator.py** – Python file for generating a caption of any image.
- **Training_caption_generator.ipynb** – Jupyter notebook in which we train and build our image caption generator.

CHAPTER 5 Building of Project

- **Import Packages**
- **Data cleaning**
- **Extracting the feature vector**
- **Loading dataset**
- **Tokenizing the vocabulary**
- **Data generator**
- **CNN-RNN model**
- **Training**
- **Testing**

5.1 Import Packages

First, we import all the necessary packages:-

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

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

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

5.2 Data Cleaning

The main text file which contains all image captions is Flickr8k.token in our Flickr_8k_text folder.

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

We will define 5 functions:

- **load_doc**(filename) – For loading the document file and reading the contents inside the file into a string.
- **all_img_captions**(filename) – This function will create a descriptions dictionary that maps images with a list of 5 captions.
- **cleaning_text(descriptions) –** This function takes all descriptions and performs data cleaning. This is an important step when we work with textual data, according to our goal, we decide what type of cleaning we want to perform on the text. In our case, we will be removing punctuations, converting all text to lowercase and removing words that contain numbers. So, a caption like "A man riding on a three-wheeled wheelchair" will be transformed into "man riding on three wheeled wheelchair"
- **text_vocabulary(descriptions) –** This is a simple function that will separate all the unique words and create the vocabulary from all the descriptions.
- **save_descriptions(descriptions, filename) –** This function will create a list of all the descriptions that have been preprocessed and store them into a file. We will create a descriptions.txt file to store all the captions.

5.3 Extracting the feature vector from all images

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

model = Xception(include_top=False, pooling='avg')

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

5.4 Loading dataset for Training the model

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

- **load_photos(filename) –** This will load the text file in a string and will return the list of image names.
- **load_clean_descriptions(filename, photos) –** This function will create a dictionary that contains captions for each photo from the list of photos. We also append the <start> and <end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.
- **load_features(photos) –** This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Xception model.

5.5 Tokenizing the vocabulary

Computers don't understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value. Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a "tokenizer.p" pickle file. Our vocabulary contains 7577 words. We calculate the maximum length of the descriptions. This is important for deciding the model structure parameters. Max_length of description is 32.

5.6 Data Generator

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

The generator will yield the input and output sequence

For example:

The input to our model is [x1, x2] and the output will be y, where x1 is the 2048 feature vector of that image, x2 is the input text sequence and y is the output text sequence that the model has to predict.

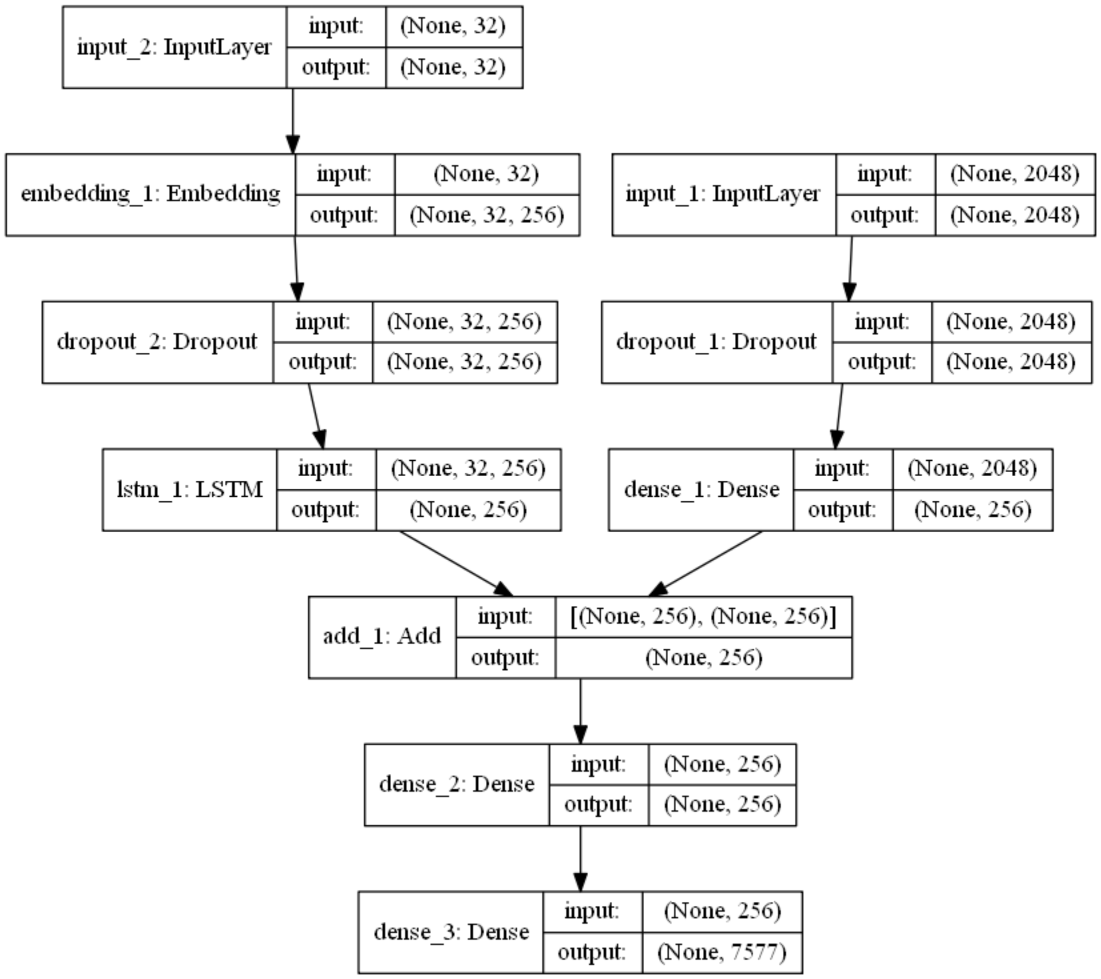
x1(feature vector)	x2(Text sequence)	y(word to predict)
feature	start,	two
feature	start, two	dogs
feature	start, two, dogs	drink
feature	start, two, dogs, drink	water
feature	start, two, dogs, drink, water	end

5.7 Defining CNN-RNN model

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

- **Feature Extractor –** The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.
- **Sequence Processor –** An embedding layer will handle the textual input, followed by the LSTM layer.
- **Decoder –** By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.

Visual representation of the final model is given below –


input_2: InputLayer | input: (None, 32) | output: (None, 32)
embedding_1: Embedding | input: (None, 32) | output: (None, 32, 256)
input_1: InputLayer | input: (None, 2048) | output: (None, 2048)
dropout_2: Dropout | input: (None, 32, 256) | output: (None, 32, 256)
dropout_1: Dropout | input: (None, 2048) | output: (None, 2048)
lstm_1: LSTM | input: (None, 32, 256) | output: (None, 256)
dense_1: Dense | input: (None, 2048) | output: (None, 256)
add_1: Add | input: [(None, 256), (None, 256)] | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 256)
dense_3: Dense | input: (None, 256) | output: (None, 7577)

5.8 Training the model

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

5.9 Testing the model

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

CHAPTER 6
Conclusion

6.1 Conclusion

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