ABSTRACT

In this project, we use CNN and LSTM to identify the caption of the image. As the deep learning techniques are growing, huge datasets and computer power are helpful to 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 like CNN and RNN. Image caption generator is a process which involves natural language processing and computer vision concepts to recognize the context of an image and present it in English. In this research paper, we carefully follow some of the core concepts of image captioning and its common approaches. We discuss Keras library[7], NumPy and Kaggle notebooks for the making of this project. We also discuss about flickr_dataset and CNN used for image classification.

Table of Contents

1. INTRODUCTION

Every day, we encounter many images from various sources such as the internet, news articles, document diagrams and advertisements. These sources contain images that viewers would have to interpret themselves. Most images do not have a description, but the human can largely understand them without their detailed captions. However, machine needs to interpret some form of image captions if humans need automatic image captions from it. Image captioning is important for many reasons. Captions for every image on the internet can lead to faster and descriptively accurate images searches and indexing. Ever since researchers started working on object recognition in images, it became clear that only providing the names of the objects recognized does not make such a good impression as a full human-like description. If machines do not think, talk, and behave like humans, natural language descriptions will remain a challenge to be solved. Image captioning has various applications in various fields such as biomedicine, commerce, web searching and military etc. Social media like Instagram, Facebook etc can generate captions automatically from images.

1.1. Motivation

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

2.LITERATURE REVIEW

Image captioning has recently gathered a lot of attention specifically in the natural language domain. There is a pressing need for context based natural language description of images, however, this may seem a bit farfetched but recent developments in fields like neural networks, computer vision and natural language processing has paved a way for accurately describing images i.e., representing their visually grounded meaning. We are leveraging state-of-the-art techniques like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and appropriate datasets of images and their human perceived description to achieve the same. We demonstrate that our alignment model produces results in retrieval experiments on datasets such as Flicker.

2.1. **Image Captioning Techniques**

There are various Image Captioning Techniques some are rarely used in present, but it is necessary to take an overview of those technologies before proceeding ahead. The main categories of existing image captioning methods and they include template-based image captioning, retrieval-based image captioning, and novel caption generation. Novel caption generation-based image caption methods mostly use visual space and deep machine learning based techniques. Captions can also be generated from multimodal space. Deep learning-based image captioning methods can also be categorized on learning techniques: Supervised learning, Reinforcement learning, and Unsupervised learning. We group the reinforcement learning and unsupervised learning into Other Deep Learning. Usually, captions are generated for a whole scene in the image. However, captions can also be generated for different regions of an image (Dense captioning). Image captioning methods can use either simple Encoder-Decoder architecture or Compositional architecture[4]. There are methods that use attention mechanism, semantic concept, and different styles in image descriptions. Some methods can also generate description for unseen objects. We group them into one category as “Others". Most of the image captioning methods use LSTM as language model. However, there are several methods that use other language models such as CNN and RNN.

2.1.1. **TEMPLATE-BASED APPROACHES**

Template-based approaches have fixed templates with several blank slots to generate captions. In these approaches, different objects, attributes, actions are detected first and then the blank spaces in the templates are filled. For example, Farhadi et al. use a triplet of scene elements to fill the template slots for generating image captions. extract the phrases related to detected objects, attributes, and their relationships for this purpose. A Conditional Random Field (CRF) is adopted by Kulkarni et al. to infer the objects, attributes, and prepsitions before filling in the gaps. Template-based methods can generate grammatically correct captions. However, templates are predefined and cannot generate variable-length captions. Moreover, later, parsing based language models have been introduced in image captioning which are more powerful than fixed template-based methods. Therefore, in this paper, we do not focus on these template-based methods.

Despite template-based captioning being capable of producing sentences that are syntactically correct and descriptions that are more relevant than retrieval-based methods, there are several disadvantages of using template-based methods. Firstly, due to the general lack of visual models and the captions generated using this method are dependent on the image content identified by visual models, the complexity, structure, novelty, and creativity of the generated captions is severely limited. Secondly, following a strict template or structure for caption generation makes the generated captions seem less natural in comparison to the human-generated descriptions.

2.1.2. **RETRIEVAL-BASED APPROACHES**

Retrieval-based Image Captioning has been a well-known approach for quite a long time. A lot of initial models for Image Captioning have been developed using this technique. As the name suggests, in this technique the caption is generated by selecting or retrieving the most probable caption from a predefined collection of captions. This technique involves finding visual similarities between the query image and the training dataset.

In image captioning, the problem of image captioning can be seen as a task of ranking. With respect to a particular image, the caption that correlates with the content of the image or the caption which is successful in accurately describing the content of the image will be given a higher rank. For this purpose, the authors propose the Analysis of Kernel Canonical

Correlation method [44,45] that correlates maximum training images and their captions to align the images and the text as per their affinity. This method will be helpful in efficiently ranking the captions and therefore the caption with the highest relatability or correlation will be retrieved.

Notwithstanding the promising nature of the proposed model, there are several limitations with this method. First, the captions that are being assigned (due to the correlation or otherwise) are well-constructed sentences provided by humans. By default, this means that the assigned caption will be grammatically correct. Providing description of the images with sentences that have been predefined cannot help in generating captions for new object mixtures. The retrieved caption might not be relevant to a new change in the picture and the model may also be incapable of adapting to minute changes within the picture.

2.1.3 **Deep Learning-Based Image Captioning**

The crucial benefit of deep convolutional neural networks (CNN) is very useful. Image captioning has in recent years garnered more research focus in AI. It has many uses, since it mainly generates an automatic sentence description for an image. It allows computer systems to recognize images for mainly education purposes, sentiment analysis, an aid for the visibly impaired, etc. The model must be accurate enough to understand the various relations between various objects and express that in a correct semantic manner in natural language. Image captioning methods primarily make use of the template-based methods, that requires describing the diverse elements (objects) in addition to their relationships and attributes.

These techniques are mainly based on the encoder decoder methodology that includes two simple steps. Firstly, using CNN, Image features are deduced to encode the image into a hard and fast period embedding vector. Secondly, generating a language description usually a recurrent neural network is used as a decoder.

CNN-RNN framework-based image captioning technique have two drawbacks in training phase:

1) Each caption gets equal importance without their individual importance

2) Objects may not be correctly recognized during caption generation

Despite the existence of several categories of deep learning methods including multimodal space, encoder decoder architecture, attention based, novel object based, language models based on LSTM, we shall focus on three of the most relevant categories, Multimodal Space, Language Models and Encoder-Decoder Architecture.

2.2. IMAGE CAPTIONING METHODS BASED ON DEEP LEARNING

Here we have the multimodal learning, Encoder-decoder Framework and LSTM

2.2.1. Multimodal learning

Template based and retrieval-based image captioning methods impose restrictions on generated sentences in generation phase. Methods using deep neural networks that do not depend on existing captions about structures of sentences can produce more communicative and adjustive sentences with more affluent structures. Using multimodal neural networks is one of the few methods that rely on pure learning to create image captions. Here, using deep convolutional neural networks, image features are first removed. Then, the extracted image feature is sent to a neural language model, which maps the image feature with the common word features and performs word predication trained on the image feature and previously generated context words. A general structure of image captioning methods employing multimodal learning is presented in figure 1

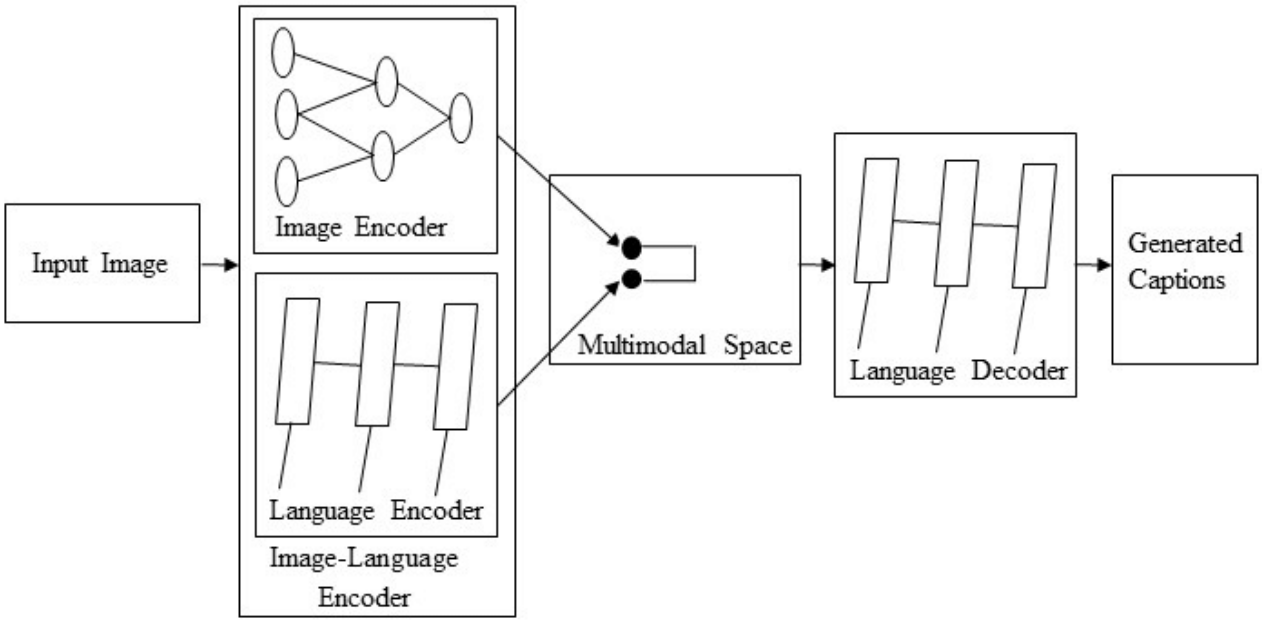


Fig -1: Multimodal space-based image captioning

A neural language model which is dependent on image inputs is suggested for generating image captions. In their method, a log-bilinear language is adapted, where an image feature is added as an extra bias to help predict the probability of generating a word along with the support of previously generated words. Feature learning is employed by back propagating gradients from the loss function through the multimodal neural network model.

This model allows the generation of the captions word by word, with each individual word being generated by conditioning on both the previously generated words and the visual features.

To generate novel captions, Mao et al. [25] proposed a multimodal Recurrent Neural Network(m-RNN). This method extracts visual features by using a deep convolutional network (CNN) and sentences by using a deep recurrent neural network (RNN) with a multimodal part as the language model. The images and sentences are both used as input in this method where the CNN and RNN both interact with each other in the multimodal layer. For the generation of the next word the probability distribution is calculated where the new word is conditioned on the input image and the previously generated words. This RNN model consists of five layers in a single time frame consisting of two word embedding layers, a recurrent layer, a multimodal layer and a SoftMax layer [25]. Various other methods utilize predefined word embedding vectors for the initialization of their language model; however, this method randomly initializes the word embedding vectors which are later learnt from the training data.

2.2.2. **Encoder-Decoder Framework**

Taking inspiration from the encoder-decoder framework[3] in neural machine translation [100] which was originally used to translate sentences and phrases from one language to another, the encoder-decoder architecture has been adopted to perform the task of image captioning by giving an image as the input and receiving the output as a sentence. The general working of this architecture includes an encoder neural network that extracts global image features which are then fed to as input to a decoder that consists of a recurrent neural network to produce a caption word by word. The general structure of this framework is shown in Fig.-2.

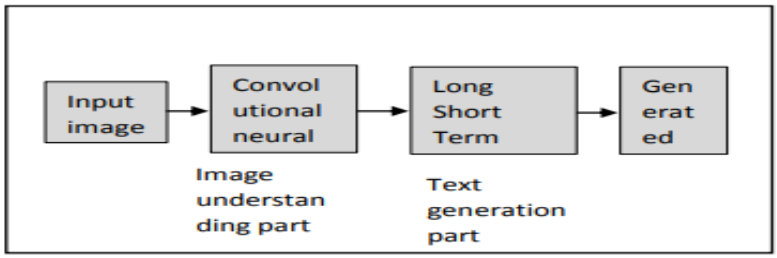


Fig-2: Encoder Decoder

An encoder-Decoder framework that effectively unifies joint image-text embedding models with multimodal neural language models. A deep Convolutional Neural Network (CNN) is used to encode the visual data whereas the textual data is encoded by employing a Long Short-Term (LSTM) Recurrent Neural Network[1]. The image features from the deep CNN are projected into the embedding space of the hidden states of the LSTM. Then by minimizing a pairwise ranking loss, the ranking of the images and descriptions is learnt. This completes the encoder part of the framework. For the decoder, a novel structure content neural language model is employed to decode image features conditioned on context word feature vectors, thus resulting in generation of novel captions word by word.

Once the model is trained, either sampling or beam search can be used to make the predictions of possible word sequences that can be used as captions.

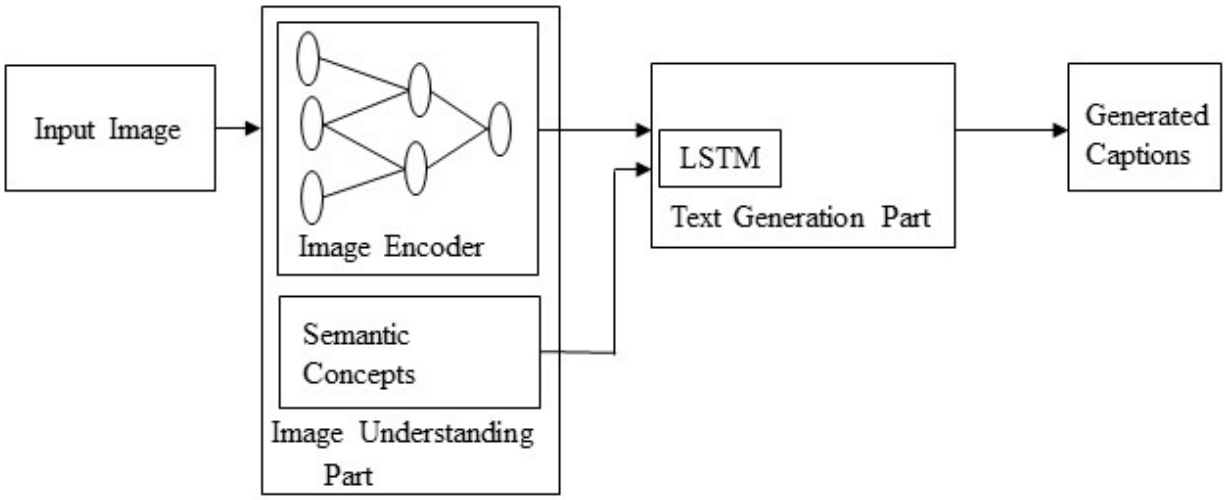


Fig-3: A block diagram of a semantic concept-based image captioning

In the previous models, the image information was fed just once, in the initial state of the LSTM thus leading to the issue of vanishing gradient thus leading to difficulty in producing

long length sentences. To solve this issue of vanishing grading proposed a guided LSTM. Global textual information is added to every gate and cell state of the LSTM. The textual information is extracted using several different methods. A multimodal embedding space can be used to extract the semantic information. Or textual information can be extracted from image captions retrieved by a crossmodal retrieval task.

The issue with unidirectional sentence generation models is while they may include past textual context, they are still limited to retain future context in case of forward direction and vice versa in case of backward direction. Thus, unidirectional models cannot produce contextually rich sentences. A bidirectional model tries to overcome this issue and utilise past and future dependence to give a prediction. Furthermore, certain object detection and classification methods have demonstrated that deep hierarchical models perform better learning in comparison than relatively shallower models. Thus propose a deep bidirectional LSTM as the decoder in the encoder decoder framework. The bidirectional model is fed sentences from both forward and backward order to make use of past and future context information.

2.2.3. **LSTM**

LSTM networks are an extension of recurrent neural networks (RNNs) mainly introduced to handle situations where RNNs fail. Talking about RNN, it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory for a short period of time (short-term memory). Nevertheless, there are drawbacks to RNNs. First, it fails to store information for a longer period. But RNNs are incapable of handling such “long-term dependencies”. Second, there is no finer control over which part of the context needs to be carried forward and how much of the past needs to be ‘forgotten’. Other issues with RNNs are exploding and vanishing gradients (explained later) which occur during the training process of a network through backtracking. Thus, Long Short-Term Memory (LSTM) was brought into the picture. It has been so designed that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, like that of Back Propagation Through Time (BPTT), which is an advantage

2.3. **Outcome of Literature Review**

Image caption generator is defined as the process of generating captions or textual descriptions for images based on the contents of the image. It is a machine learning task that involves both natural language processing (for text generation) and computer vision (for understanding image contents). Auto image captioning is a very recent and growing research problem nowadays. In above (2.2) we showed all the processes that takes place during generating captions from images. We found that CNN is used to understand image contents and find out objects in an image while RNN or LSTM is used for language generation. We found that in dense captioning, captions are generated for each region of the scene. We found that supervised learning uses labelled input and output data, while an unsupervised learning algorithm does not. We found that two most promising methods for implementing this model are Encoder Decoder, and attention mechanism and a combination of them can help in improving results.

2.4. **Problem Statement**

To develop a system for users, which can automatically generate the description of an image with the use of CNN along with LSTM. Automatically describing the content of images using natural language is a fundamental and challenging task. Although great development has been made in computer vision, tasks such as recognizing an object, action classification, image classification, attribute classification and scene recognition are possible, but it is a relatively new task to let a computer describe an image that is forwarded to it in the form of a human.
So, to make our image caption generator model, we had taken the flickr8k dataset and passes it to the ResNet50 (ResNet-50 model is a convolutional neural network (CNN) that is 50 layers deep.) and it will extract the image features in the vector of 2048 values and ResNet takes 224*224 image, and it will give 2000 different classes.

2.5. **Research Objectives**

The objective of our project is to learn the concepts of CNN and LSTM model and build a working model of Image caption generator by implementing CNN with LSTM. In this project, we will be implementing the caption generator using CNN (Convolutional Neural Networks) and LSTM (Long short term memory).

3.Methodology and Framework

3.1. System Architecture

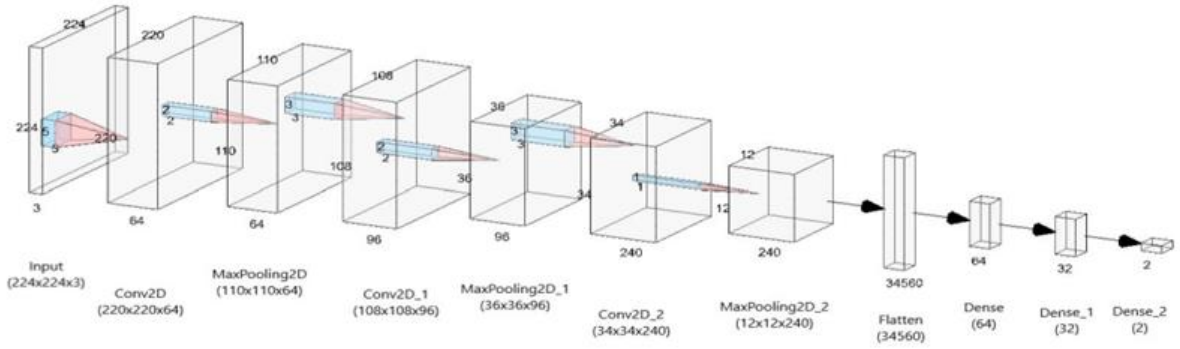


Fig-4: CNN Architecture

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

A convolutional neural network consists of an input layer, hidden layers and an output layer.

In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution.

In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

1.The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

2.The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

The most popular process is max pooling, which reports the maximum output from the neighborhood.

3.Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect.

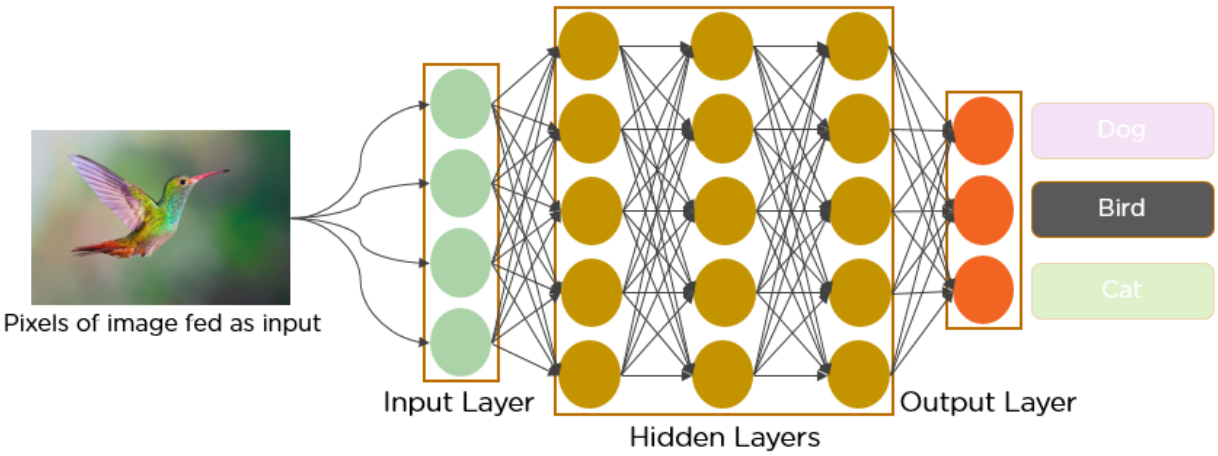
The FC layer helps to map the representation between the input and the output.

3.2. **Algorithms and techniques**

In our project, we used the Deep learning Algorithm and Encoder-Decoder language model.

3.2.1. **Deep learning Algorithm**

Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.



3.2.2. encoder Decoder language model

An Encoder-Decoder architecture was developed where an input sequence was read in entirety and encoded to a fixed-length internal representation.

A decoder network then used this internal representation to output words until the end of sequence token was reached. LSTM networks were used for both the encoder and decoder.

The final model was an ensemble of 5 deep learning models. A left-to-right beam search was used during the inference of the translations

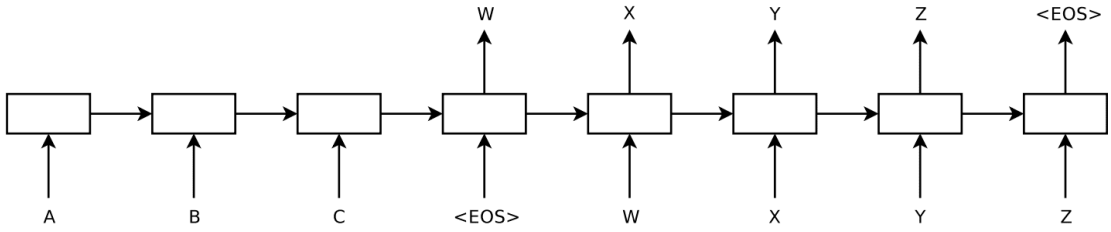


Figure-5: Encoder-Decoder model for Text Translation

3.2.3. Image caption Generating model

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

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

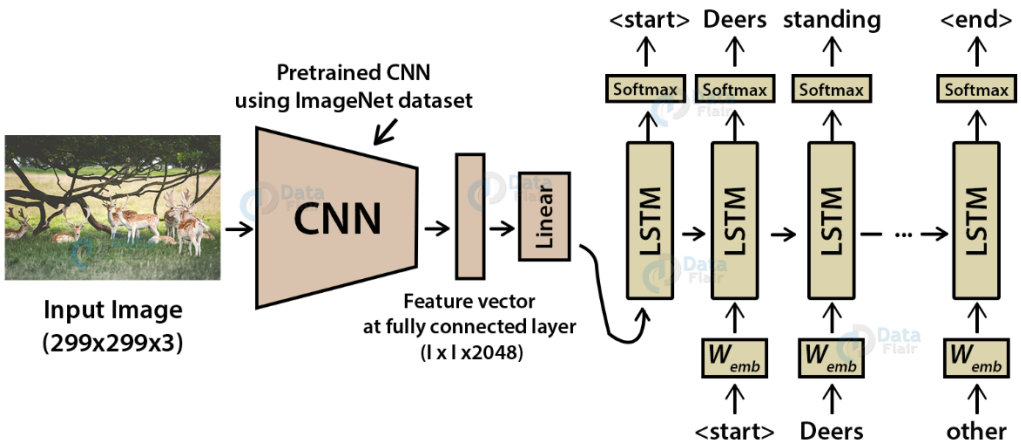


Figure-6: Model-Image caption generator

3.3. System Design

Here we will explain the system design part of our project.

3.3.1. FLICKR8K DATASET

Flickr8k dataset is a public benchmark dataset for image to sentence description. This dataset consists of 8091 images [2] with five captions for each image. Each caption provides a clear description of entities and events present in the image. Features of the dataset making it suitable for this project are:

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

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

3.3.2. Image Data Preparation

The image should be converted to suitable features so that they can be trained into a deep learning model. Feature extraction is a mandatory step to train any image in deep learning model. The features are extracted using Convolutional Neural Network (CNN) with ResNet50 model [6].

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As the Resnet-50 will extract the features from the image in the vector of 2048 values. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

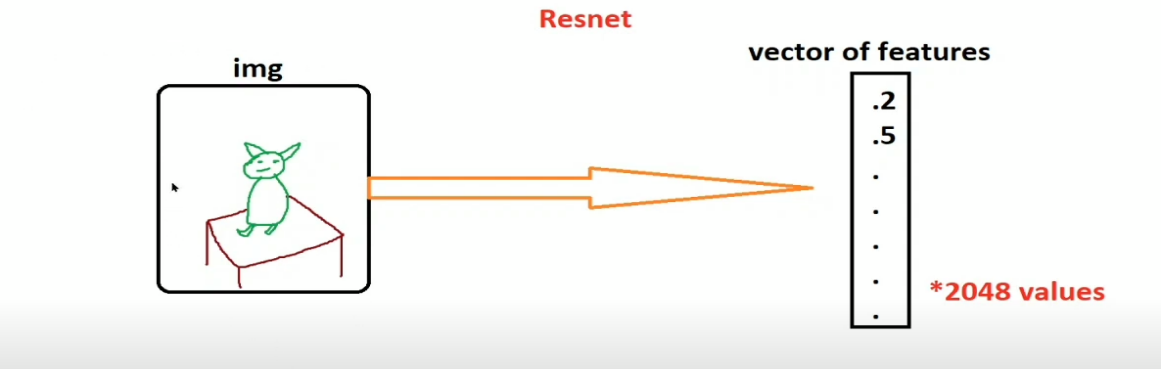


Figure-7: ResNet50 extract the features from image in the vector of 2048 values

3.3.3. **Caption Data Preparation**

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

3.3.3.1. **Data cleaning**

To make the text dataset work in machine learning or deep learning models, raw text should be converted to a usable format. The following text cleaning steps are done before using it for the project: • Removal of punctuations. • Removal of numbers. • Removal of single length words. • Conversion of uppercase to lowercase characters. Stop words are not removed from the text data as it will hinder the generation of a grammatically complete caption which is needed for this project.

4.Work Done

4.1. Pre-Requisites

This project requires good knowledge of Deep learning, Python, working on Jupyter notebooks, Keras library, Numpy, and Natural language processing. Make sure you have installed all the following necessary libraries:

- pip install tensorflow
- keras
- numpy
- open cv
- jupyterlab,

4.2. Load the data

Downloaded from dataset**:**

- Flickr8k_Dataset – Dataset folder which contains 8091 images.
- Flickr_8k_text – Dataset folder which contains text files and captions of images. 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 pre-processing.
- Testing_caption_generator.py – Python file for generating a caption of any image.
- Training_caption_generator.ipynb – In which we train and build our image caption generator.
- And we also import the paths of the Images, captions, train and test.

4.3. Image Preprocessing

- we need to import some inputs numpy, pandas, cv2, os, glob to manipulate the images..
- And then we'll display the image and the captions to it by using data set.

```
A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
A little girl is sitting in front of a large painted rainbow .
A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
There is a girl with pigtails sitting in front of a rainbow painting .
Young girl with pigtails painting outside in the grass .
```

• To import ResNet model from keras. And now it will download ResNet50.

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/re
snet50_weights_tf_dim_ordering_tf_kernels.h5
102973440/102967424 [==============================] - 1s 0us/step
```

• And now to remove the Dense layer and to keep Averagepooling layer as output layer for that we need to create a new modele and take **last = incept.model.layers[-2].output**

```
__________________________________________________________________________________________________
res5c_branch2c (Conv2D)         (None, 7, 7, 2048)   1050624     activation_48[0][0]
__________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 7, 7, 2048)   8192        res5c_branch2c[0][0]
__________________________________________________________________________________________________
add_16 (Add)                    (None, 7, 7, 2048)   0           bn5c_branch2c[0][0]
                                                                 activation_46[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 7, 7, 2048)   0           add_16[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 2048)         0           activation_49[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

4.4. Text Pre-processing

• Here we add the start and end of sequence at the ends of the captions. This will make easy to us to split the captions for the particular image code to identify.

	image_id	captions
0	2513260012_03d33305cf.jpg	<start> A black dog is running after a white d...
1	2513260012_03d33305cf.jpg	<start> Black dog chasing brown dog through sn...
2	2513260012_03d33305cf.jpg	<start> Two dogs chase each other across the s...
3	2513260012_03d33305cf.jpg	<start> Two dogs play together in the snow . <...
4	2513260012_03d33305cf.jpg	<start> Two dogs running through a low lying b...

• Captions stored in sentences

```
['<start> A black dog is running after a white dog in the snow . <end>',
 '<start> Black dog chasing brown dog through snow <end>',
 '<start> Two dogs chase each other across the snowy ground . <end>',
 '<start> Two dogs play together in the snow . <end>',
 '<start> Two dogs running through a low lying body of water . <end>']
```

• Print the padded sequences

```
[[7461    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472 4138    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472 4138  131    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472 4138  131 1865    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472 4138  131 1865 5107    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]
 [7461 5165 8078 4138 6489 4422 4174  590 2472 4138  131 1865 5107 7763
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0]]
```

• Printing the length of padded sequences

```
<start>
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
UNK
```

4.5. Create vocabulary

• And now we need to check the vocabulary, for this we use the captions_dict and we will split the words and then count them as integer. And we can see the no.of distinct words are available

```
4073
```

• And **count_words** will display this

```
{'startofseq': 7499,
 'a': 11690,
 'little': 342,
 'girl': 615,
 'covered': 55,
 'in': 3633,
 'paint': 7,
 'sits': 109,
 'front': 257,
 'of': 1167,
 'painted': 9,
 'rainbow': 11,
 'with': 1518,
 'her': 195,
 'hands': 37,
 'bowl': 7,
 '.': 6803,
 'endofseq': 7499,
 'is': 1718,
 'sitting': 252,
 'large': 193,
 'small': 230,
 'the': 3314,
 'grass': 313
```

• And we are iterating with captions_dict and after that we iterate with 5 different captions for an image, and we have encoded [] list i.e., string converted to integer captions.

And now captions dictionary looks like as captions converted to integers and key converted to image code

```
{'1012212859_01547e3f17.jpg': [[1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   2,
   11,
   12,
   13,
   14,
   15,
   16,
   17],
  [1, 2, 18, 3, 4, 19, 8, 20, 21, 2, 22, 23, 24, 25, 12, 16, 17],
  [1, 3, 23, 25, 12, 26, 27, 10, 28, 19, 9, 29, 30, 31, 17],
  [1, 18, 3, 32, 23, 2, 11, 12, 19, 8, 9, 7, 8, 31, 16, 17],
  [1, 18, 3, 23, 33, 34, 35, 7, 31, 23, 6, 36, 14, 37, 38, 16, 17]],
 '1016887272_03199f49c4.jpg': [[1, 2, 39, 21, 37, 40, 41, 2, 42, 16, 17],
  [1, 2, 43, 21, 44, 45, 46, 41, 19, 2, 46, 41, 47, 16, 17],
  [1, 2, 43, 21, 44, 41, 2, 46, 48, 37, 49, 50, 17],
  [1, 51, 52, 45, 53, 2, 46, 54, 55, 56, 49, 28, 57, 8, 58, 16, 17],
  [1,
```

4.6. Build generator function

• Formation of an array

```
array([ 7499., 11690.,   342.,     0.,     0.,     0.,     0.,     0.,
           0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
           0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
           0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
           0.,     0.,     0.,     0.])
```

4.7. **MODEL**

- Image model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 40, 128)           0
=================================================================
Total params: 262,272
Trainable params: 262,272
Non-trainable params: 0
_________________________________________________________________
```

- Language model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 40, 128)           1056512
_________________________________________________________________
lstm_1 (LSTM)                (None, 40, 256)           394240
_________________________________________________________________
time_distributed_1 (TimeDist (None, 40, 128)           32896
=================================================================
Total params: 1,483,648
Trainable params: 1,483,648
Non-trainable params: 0
_________________________________________________________________
```

- Concatenate the models

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
embedding_1_input (InputLayer)   (None, 40)            0
____________________________________________________________________________________________________
dense_1_input (InputLayer)       (None, 2048)          0
____________________________________________________________________________________________________
embedding_1 (Embedding)          (None, 40, 128)       1056512     embedding_1_input[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 128)           262272      dense_1_input[0][0]
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 40, 256)       394240      embedding_1[0][0]
____________________________________________________________________________________________________
repeat_vector_1 (RepeatVector)   (None, 40, 128)       0           dense_1[0][0]
____________________________________________________________________________________________________
time_distributed_1 (TimeDistrib (None, 40, 128)       32896       lstm_1[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 40, 256)       0           repeat_vector_1[0][0]
                                                                   time_distributed_1[0][0]
____________________________________________________________________________________________________
lstm_2 (LSTM)                    (None, 40, 128)       197120      concatenate_1[0][0]
____________________________________________________________________________________________________
lstm_3 (LSTM)                    (None, 512)           1312768     lstm_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 8254)          4234302     lstm_3[0][0]
____________________________________________________________________________________________________
activation_50 (Activation)       (None, 8254)          0           dense_3[0][0]
====================================================================================================
Total params: 7,490,110
Trainable params: 7,490,110
Non-trainable params: 0
____________________________________________________________________________________________________
```

• Model fit

```
25493/25493 [==============================] - 9s 367us/step - loss: 0.2677 - acc: 0.8996
Epoch 195/200
25493/25493 [==============================] - 10s 385us/step - loss: 0.2672 - acc: 0.9004
Epoch 196/200
25493/25493 [==============================] - 10s 374us/step - loss: 0.2622 - acc: 0.9023
Epoch 197/200
25493/25493 [==============================] - 9s 367us/step - loss: 0.2651 - acc: 0.8999
Epoch 198/200
25493/25493 [==============================] - 9s 368us/step - loss: 0.2636 - acc: 0.9023
Epoch 199/200
25493/25493 [==============================] - 10s 383us/step - loss: 0.2645 - acc: 0.9021
Epoch 200/200
25493/25493 [==============================] - 9s 366us/step - loss: 0.2605 - acc: 0.9004
```

4.8. Predictions [5]

When we give the image input, it will show us the caption for that by extracting the features of the image using resnet-50.

Input-

```
img = "../input/flickr_data/Flickr_Data/Images/1003163366_44323f5815.jpg"

test_img = get_encoding(resnet, img)
```

Output-

```
z = Image(filename=img)
display(z)

print(Argmax_Search)
```

A girl is is standing on a lake

Similarly-

Input-

```
img = "../input/flickr_data/Flickr_Data/Images/1007320043_627395c3d8.jpg"

test_img = get_encoding(resnet, img)
```

Output-

```
z = Image(filename=img)
display(z)

print(Argmax_Search)
```

Two vendor and women are play on a stroller .

• Similarly based on the image it will give us the captions by features extraction.

4.9. Individual Contribution

Bharath – I had done the image pre-processing (Reading image from directory, importing ResNet, pooling layer, splitting, make changes) and text pre-processing (decode it to ‘utf-8, adding ‘startofseq’ and ‘endofseq’ to caption) and predict the model.

Sumanth – I had crated the vocabulary (check no. of distinct words available in captions, converting captions to integers), build generator function (to check captions individually with X, y_in, y_out) and predict the model.

5.Conclusion and Future Plan

Conclusion

In this paper, we have reviewed deep learning-based image captioning methods. We have used Flickr_8k dataset which includes nearly 8000 images, and the corresponding captions are also stored in the text file. And we had done image and text pre-processing respectively. And we had seen how the Resnet50 work in the dataset which then produces encoding. And how encoder-decoder recurrent neural network architecture is used to address the image caption generation problem. And we had seen how LSTM work for decoding process. Although deeplearning -based image captioning methods have achieved a remarkable progress in recent years, a robust image captioning method that is able to generate high quality captions for nearly all images is yet to be achieved. With the advent of novel deep learning network architectures, automatic image captioning will remain an active research area for some time. The scope of image-captioning is very vast in the future as the users are increasing day by day on social media and most of them would post photos. So, this project will help them to a greater extent.

Future Plan

Future plan in Image captioning has become an important problem in recent days due to the exponential growth of images in social media and the internet. As feature extraction and similarity calculation in images are challenging in this domain, there is a tremendous scope of possible research in the future. This project can be further enhanced in future to improve the identification of classes which has a lower precision by training it with more image captioning datasets. This methodology can also be combined with previous image retrieval methods such as histogram, shapes, etc. and can be checked if the image retrieval results get better. And Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media, and several other natural language processing applications.

6.References

[1] Soheyla Amirian∗, Khaled Rasheed†, Thiab R. Taha‡, Hamid R. Arabnia, “Image Captioning with Generative Adversarial Network”, IEEE, International Conference on Computational Science and Computational Intelligence (CSCI), 2019

[2] Sheshang Degadwala, Dhairya Vyas, Haimanti Biswas, Utsho Chakraborty, Sowrav Saha, “Image Captioning Using Inception V3 Transfer Learning Model”, Proceedings of the 6th International Conference on Communication and Electronics Systems (ICCES-2021) IEEE Xplore Part Number: CFP21AWO-ART; ISBN: 978-0-7381-1405-7

[3] Vaishnavi Agrawal, Shariva Dhekane, Vibha Vyas, Neha Tuniya, “Image Caption generator using attention mechanism”, IEEE, 2021

[4] Ansar Hani, Najiba Tagougui, Monji Kherallah, “Image caption generation using a deep architecture”, International Arab Conference on Information Technology ACIT, 2019.

[5] Subhash Chand Gupta, Nidhi Raj Singh, Tulsi Sharma, Akshita Tyagi, Rana Majumdar, “Generating Image Captions using Deep Learning and Natural Language Processing”, 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) Amity University, Noida, India. Sep 3-4, 2021

[6]Nishanth Behar and Manish Shrivastava, “ResNet50-BasedEffectiveModelfor BreastCancer Classification Using Histopathology Images”, Tech Science press, 2021

[7] Pranay Mathur, Aman gill, Aayush Yadav, Anurag Mishra, Nand Kumar, “Camera2Caption: A Real-Time Image Caption Generator”, IEEE, 2017 International Conference on Computational Intelligence in Data Science (ICCIDS)