

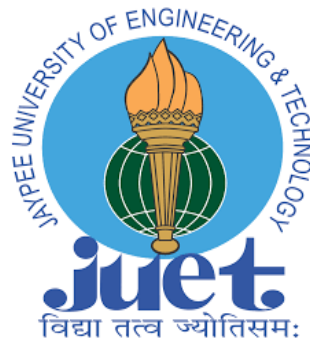
AUTOMATED IMAGE CAPTION GENERATOR

A Project Report

Submitted by:

1. Bhavya Tyagi (181B070)
2. Bhawana Mishra(181B071)
3. Chandan Kumar(181B074)

Under the guidance of Mr. Kunj Bihari Meena



Nov-2020

**Submitted in partial fulfillment of the Degree of
Bachelor of Technology**

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY GUNA (M.P.)-473226

Declaration by the Student

I hereby declare that the work reported in the B.Tech project entitled as “**Automated Image Caption Generator**”, in partial fulfillment for the award of degree of B. Tech submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation I will solely be responsible.

1. Bhavya Tyagi(181B070)
2. Bhawana Mishra(181B071)
3. Chandan Kumar(181B074)

Signature of the Student

Date:

ACKNOWLEDGEMENT

When undergoing through the making of a certain project, there exists a lot of contribution and guidance of other people who support us in achieving what lies in front. Not only do they guide our path, but also play a vital role in the establishment and understanding of what we really want to achieve.

Thus, this is to show our gratitude towards our Mentor, Mr. K.B. Meena for guiding us through our entire Report. He supported us immensely in the better understanding of the project, guiding us towards the better knowledge of the project we were working on. The weekly interaction with him brought to us more clarification on the topic and thus simplifying our task.

1. Bhavya Tyagi(181B070)
2. Bhawana Mishra(181B071)
3. Chandan Kumar(181B074)

Signature of the Student

Date:

Executive Summary

In recent years, with the rapid development of artificial intelligence, image caption has gradually attracted the attention of many researchers in the field of artificial intelligence and has become an interesting and arduous task. Image caption, automatically generating natural language descriptions according to the content observed in an image, is an important part of scene understanding, which combines the knowledge of computer vision and natural language processing. The application of image caption is extensive and significant, for example, the realization of human-computer interaction.

Common uses for Automated Image Caption Generator (AICG) are that it can act like a Caption Bot. For example, It can generate relevant captions for Blog Post. It also makes google Image Search easy.

Furthermore, the advantages and the shortcomings of this model are discussed. providing the commonly used datasets in this field.

List of Figures

Figure 1. A Functional CNN-RNN Model

Figure 2. CNN-LSTM Architecture for Image Captioning

Figure 3. Schematic of the Merge Model For Image Captioning

Figure 4. A-CNN-is-composed-of-two-basic-parts-of-feature-extraction-and-classification-Feature

Figure 5. Converting the image captions into a list of tokenized words.

Figure 6. Image Captioning Model by Team Oodles

Figure 7. Plot of the Caption Generation Deep Learning Model

CONTENTS

Title Page

Declaration by the Student

Acknowledgement

Executive Summary

List of Figures

1. Introduction	7-15
1.1 Advantages of Automated Image Caption Generator	8-8
1.2 Limitations of this model	8-8
1.3 Applications of Automated Image Caption Generator	9-10
1.4 Technology Used	10-10
1.5 Technical Description	11-15
2. What constitutes Automated Image Captioning Model?	16-23
2.1 CNN	17-18
2.2 RNN	18-18
2.3 LSTM	19-19
2.4 3 Phases of Automated Image Caption Generator	20-23
3. Train with Progressive Loading.	23-23
4. Related Works	24-27
5. Difference between Machine learning and Deep learning	28-28
6. Code Snapshots	29-30
7. Output snapshots	31-31
8. Experiments and Results.	32-33
9. Conclusion and Future Scope	33-34
References & Citations	35-35
Student Details	36-36

1. Introduction

Caption Generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem.

Deep Learning methods have demonstrated state-of-the-art results on caption generation problems. What is most impressive about these methods is a single end-to-end model can be defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

Common uses for Automated Image Caption Generator (AICG) are that it can act like a Caption Bot. For example, It can generate relevant captions for Blog Post. It also makes google Image Search easy. The development of the image description system may help the visually impaired people “see” the world in the future. Recently, it has drawn increasing attention and become one of the most important topics in computer vision.

1.1 ADVANTAGES OF AUTOMATED IMAGE CAPTION GENERATOR

Visuals and imagery continue to dominate social and professional interactions globally. With a growing scale, manual efforts are falling short on tracking, identifying, and annotating the prodigious amounts of visual data.

With the advent of artificial intelligence, multimedia businesses are able to accelerate the process of image captioning while generating significant value.

AI-powered image caption generator employs various artificial intelligence services and technologies like deep neural networks to automate image captioning processes.

1.2 LIMITATIONS OF THIS MODEL

Disadvantages of retrieval based image captioning methods are obvious. Such methods transfer well-formed human-written sentences or phrases for generating descriptions for query images.

Although the yielded outputs are usually grammatically correct and fluent, constraining image descriptions to sentences that have already existed can not adapt to new combinations of objects or novel scenes.

Under certain conditions, generated descriptions may even be irrelevant to image contents.

Retrieval based methods have large limitations to their capability to describe images.

Our model will depend on the data, so, it can not predict the words that are out of the scope Of its vocabulary.

1.3 APPLICATIONS OF AUTOMATED IMAGE CAPTION GENERATOR

The AI-powered image captioning model is an automated tool that generates concise and Meaningful captions for prodigious volumes of images efficiently. The model employs Techniques from computer vision and Natural Language Processing(NLP) to extract Comprehensive textual information about the given images.

1) Recommendations in Editing Applications

The image captioning model automates and accelerates the close captioning process for Digital content production, editing, delivery, and archival. Well-trained models replace manual efforts for generating quality captions for images as well as videos.

2) Assistance for Visually Impaired

The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals.

With AI-powered image caption generator, image descriptions can be read out to visually impaired, enabling them to get a better sense of their surroundings.

3) Media and Publishing Houses

The media and public relations industry circulate tens of thousands of visual data across borders in the form of newsletters, emails, etc.

The image captioning model accelerates subtitle creation and enables executives to focus on more important tasks.

4) Social Media Posts

For social media, artificial intelligence is moving from discussion rooms to underlying mechanisms for identifying and describing terabytes of media files.

It enables community administrators to monitor interactions and analysts to formulate business strategies.

1.4 TECHNOLOGY USED

- 1. Python 3**
- 2. Numpy(v1.19.0)**
- 3. Keras(v2.4.3)**

The model requires:

- 1. Methods from Computer Vision(to understand the content of the image)**
- 2. Language model from NLP (to turn the understanding of image into Words)**
- 3. After training the model, its learning will be tested using Deep Learning.**

1.5 TECHNICAL DESCRIPTION

Python 3



Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python is named after a TV Show called "Monty Python's Flying Circus" and not after Python-the snake.

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatible, later on many of its important features have been backported to be compatible with version 2.7.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

NumPy



NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.

Using NumPy, mathematical and logical operations on arrays can be performed.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Matplotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.

Keras



While deep neural networks are all the rage, the complexity of the major frameworks has been a barrier to their use for developers new to machine learning. There have been several proposals for improved and simplified high-level APIs for building neural network models, all of which tend to look similar from a distance but show differences on closer examination. Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines.

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

Software Used for Execution

Jupyter Notebook using Anaconda Framework in python

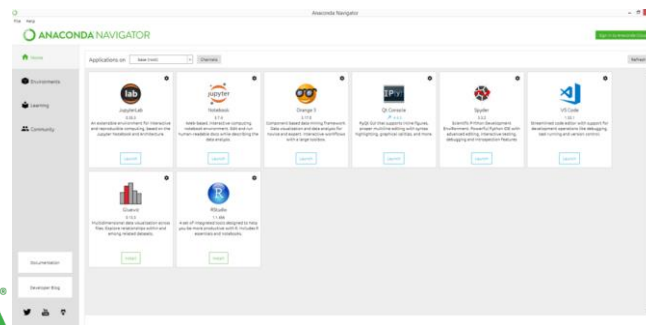


At some point, we all need to show our work. Most programming work is shared either as raw source code or as a compiled executable. The source code provides complete information, but in a way that's more "tell" than "show." The executable shows us what the software does, but even when shipped with the source code it can be difficult to grasp exactly how it works.

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at [Project Jupyter](https://projectjupyter.org/).

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Anaconda Framework



Anaconda Navigator

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](#) as well as the [conda](#) package and virtual environment manager. It also includes a GUI, **Anaconda Navigator**, as a graphical alternative to the command line interface (CLI).

Anaconda Navigator is a desktop graphical user interface(GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS And Linux.

The following applications are available by default in navigator:-

- Jupyter Lab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

2. WHAT CONSTITUTES AN AI-POWERED IMAGE CAPTIONING MODEL?

The AI-infused image caption generator is packed with deep learning neural networks; namely, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short Term Memory (LSTM), wherein-

- 1) CNNs are deployed for extracting spatial information from the images
- 2) RNNs are harnessed for generating sequential data of words
- 3) LSTM is good at remembering lengthy sequences of words

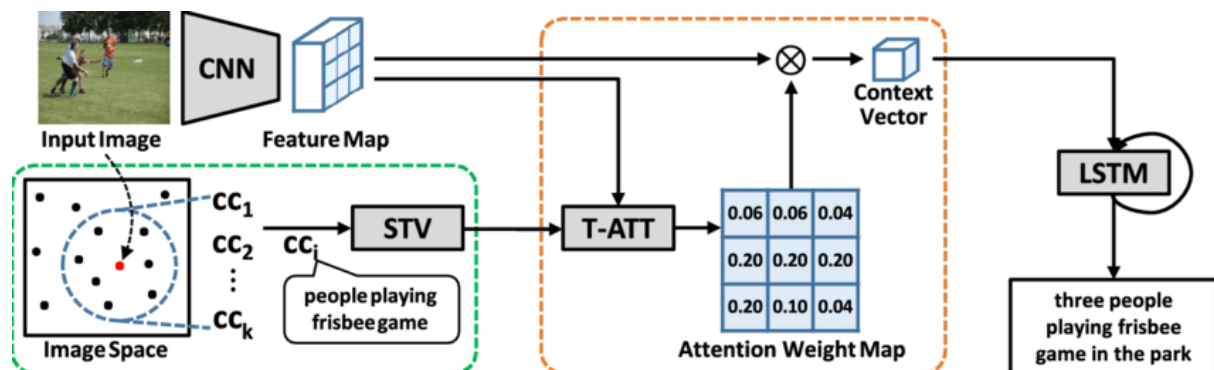


Figure 1.

A functional CNN-RNN model.

Image Source- Research Gate

2.1 CONVOLUTIONAL NEURAL NETWORKS(CNN)

Convolutional Neural Networks or CNN is a type of deep neural networks that are efficient at extracting meaningful information from visual imagery. As an experiential AI Development Company, Oodles AI decodes the underlying layers of CNN and how businesses can deploy CNN for computer vision applications.

When it comes to us, humans, evolution has gifted us with very complex yet efficient techniques to view and detect several objects. Our brain keeps on learning continuously without our notice. There are several organs and parts of our brain involved in the process like eyes, receptors and visual cortex.

In the era, with the resources and immense computational power, it would be pointless not to explore computer vision. With so many applications of computer vision services, we can take current generation technology to the next level. A great example is the upcoming Tesla's Robo-taxi which gives us a glimpse into the future.

A very popular machine learning algorithm, especially for Object Detection, is Convolutional Neural Networks or CNN. CNN consists of four hidden layers such as-

Convolutional layers

Pooling layers

fully connected layers, and

Normalization layers.

Convolutional Layers takes two input layers - a part of the image and an equally sized filter called the kernel. The output of this layer is the dot product of both inputs.

The idea of Pooling is to down-sample data. The Pooling Layer takes the input (an image) and reduces its size in terms of a number of pixels. There are two ways to perform this - Max Pooling and Min Pooling. Max Pooling picks the maximum value from the selected region, whereas Min Pooling picks up the minimum value.

Under Fully Connected Layers, as the name suggests, all the outputs from one layer are connected to the input of another layer. These layers are useful in the classification of the data.

Normalization Layers are used to stabilize the neural networks. It performs normalization on the input data.

CNN performs incredibly when it comes to analyzing a single image, but it lacks one essential quality - they only consider spatial features and visual data ignoring the temporal and time features i.e., how a frame is related to the previous frame. This is where Recurrent Neural Networks or RNN come into play. The term 'recurrent' suggests that the neural network repeats the same tasks for every sequence. RNN can also be used in Natural Language Processing.

2.2 RECURRENT NEURAL NETWORKS(RNN)

RNNs are harnessed for generating sequential data of words.

In neural image captioning systems, a recurrent neural network (RNN) is typically viewed as the primary 'generation' component. This view suggests that the image features should be 'injected' into the RNN. This is in fact the dominant view in the literature. Alternatively, the RNN can instead be viewed as only encoding the previously generated words. This view suggests that the RNN should only be used to encode linguistic features and that only the final representation should be 'merged' with the image features at a later stage.

2.3 LONG SHORT TERM MEMORY(LSTM)

We use a deep convolutional neural network to generate a vectorized representation of an image that we then feed into a Long-Short-Term Memory (LSTM) network, which then generates captions.

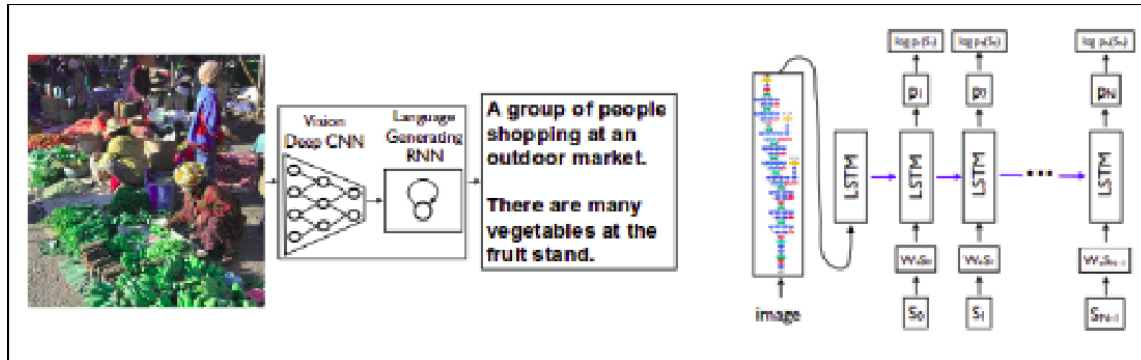


Figure 2. CNN-LSTM ARCHITECTURE FOR IMAGE CAPTIONING

We use a deep convolutional neural network to create a semantic representation of an image, which we then decode using a LSTM network. (Right) A unrolled LSTM network for our CNN-LSTM model. All LSTMs share the same parameters. The vectorized image representation is fed into the network, followed by a special start of sentence token. The hidden state produced is then used by the LSTM predict/generate the caption for the given image.

2.4 3 PHASES OF AUTOMATED IMAGE CAPTION GENERATOR

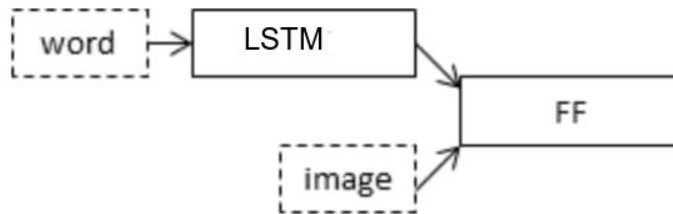


Figure 3. Schematic of the Merge Model For Image Captioning

1) Feature Extraction

The first move is made by CNNs to extract distinct features from an image based on its spatial context. CNNs create dense feature vectors, also called embedding, that is used as an input for the following RNN algorithms.

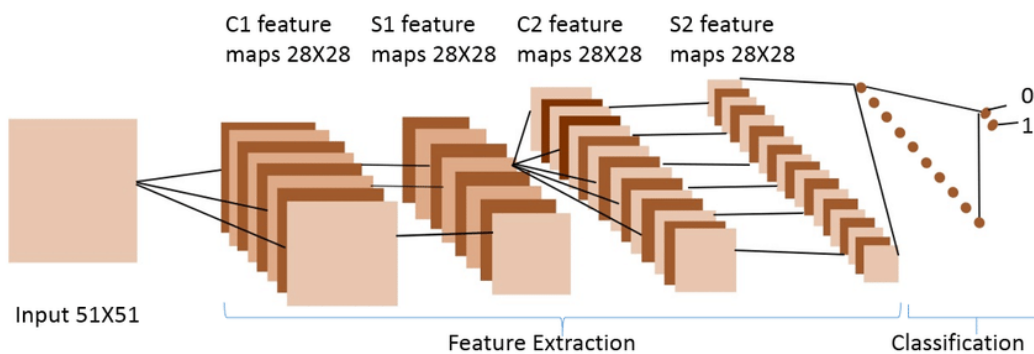


Figure 4. A-CNN-is-composed-of-two-basic-parts-of-feature-extraction-and-classification-Feature

The CNN is fed with images as inputs in different formats including png, jpg, and others. The neural networks compress large amounts of features extracted from the original image into smaller and RNN-compatible feature vector. It is the reason why CNN is also referred to as 'Encoder'.

2) Tokenization

The second phase brings RNN into the picture for ‘decoding’ the process vector inputs generated by the CNN module. For initiating the task for captions, the RNN model needs to be trained with a relevant dataset. It is essential to train the RNN model for predicting the next word in the sentence. However, training the model with strings is ineffective without definite numerical alpha values. For this purpose, it required to convert the image captions into a list of tokenized words as shown below-

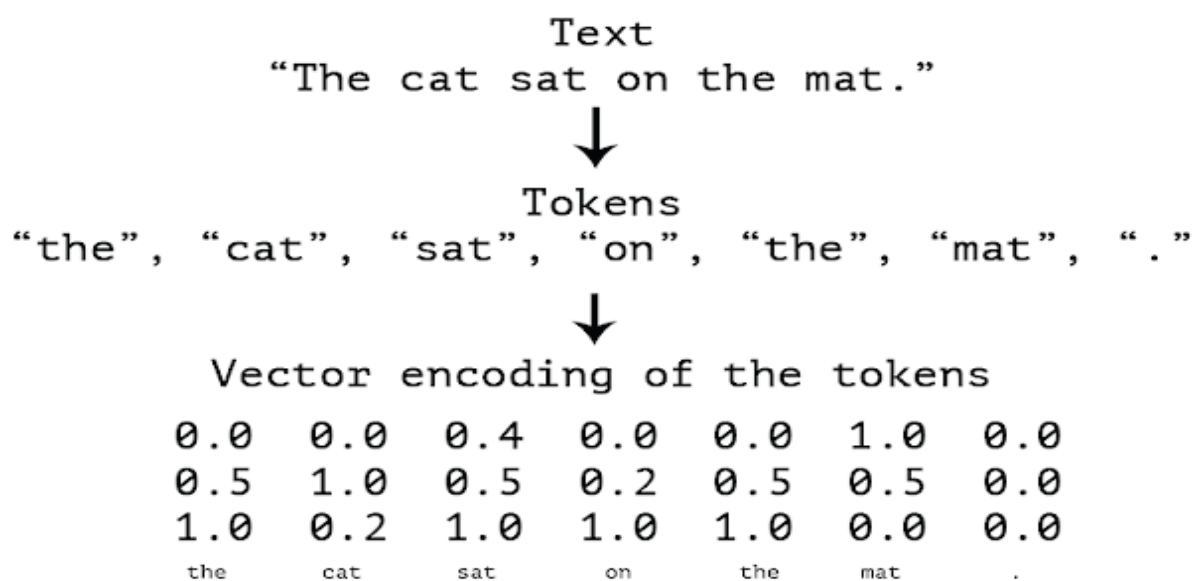


Figure 5.

Image Source- Manning

3) Text Prediction

Post tokenization, the last phase of the model is triggered using LSTM. This step requires an embedding layer for transforming each word into the desired vector and eventually pushed for decoding. With LSTM, the RNN model must be able to remember spatial information from the input feature vector and predict the next word. Now with LSTM performing its tasks, the final output is generated by calling the (get_prediction) function.

Recently, a company named **Oodles** built an image captioning model powered by deep neural networks. Here's how it process the images to generate near accurate outputs-

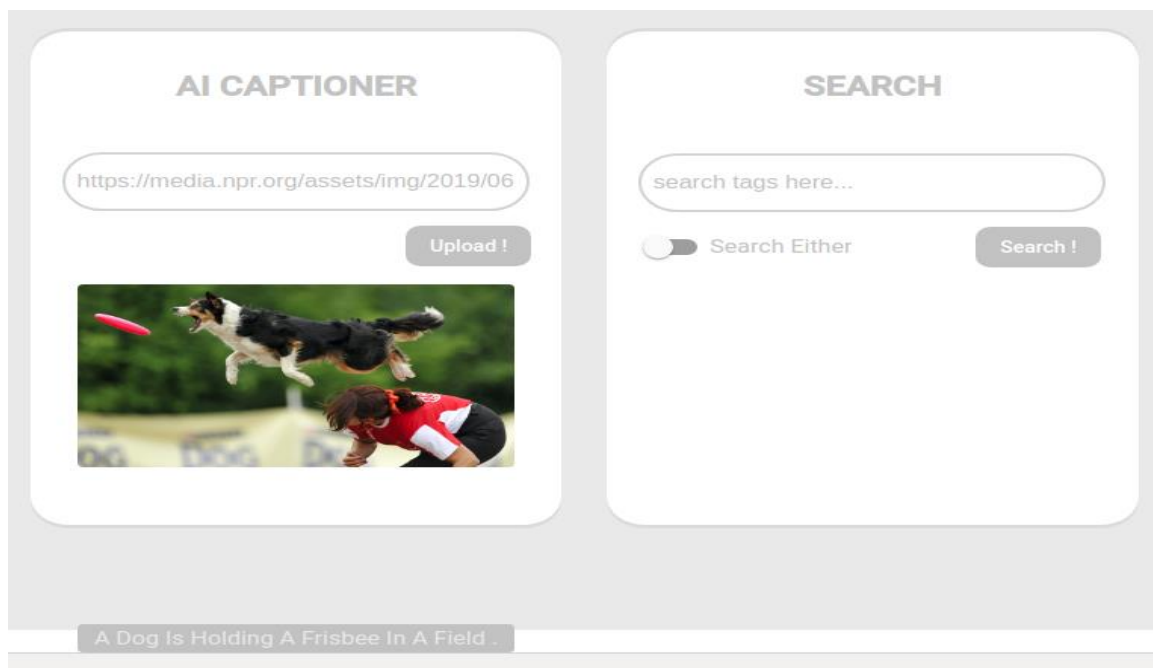


Figure 6. **Image Captioning Model by Team Oodles**
Image Source Oodles AI

In addition to image captioning, the model can be used to search for relevant images with input in the form of tags such as “cars”, “books”, etc.

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.

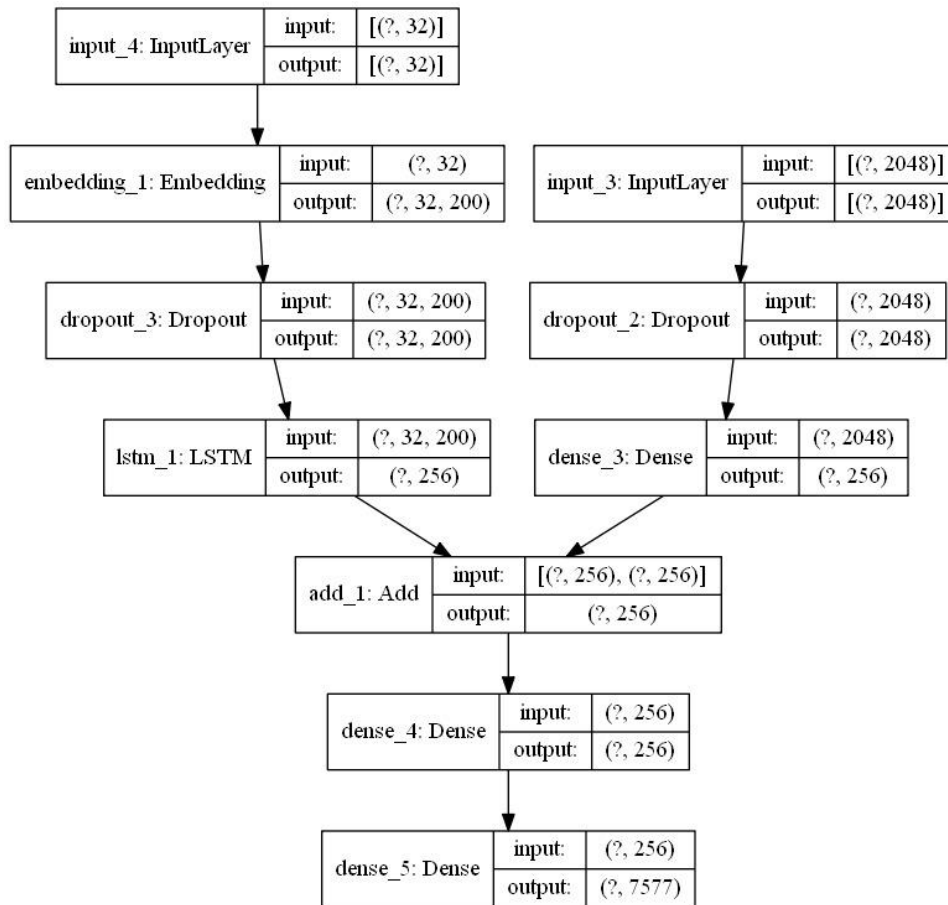


Figure 7. Plot of the Caption Generation Deep Learning Model

3. TRAIN WITH PROGRESSIVE LOADING

The training of the caption model does assume you have a lot of RAM.

The code in the previous section is not memory efficient and assumes you are running on a large EC2 instance with 32GB or 64GB of RAM. If you are running the code on a workstation of 8GB of RAM, you cannot train the model.

8GB of RAM should be more than capable.

4. RELATED WORKS

There are many number of applications wherever images and deep learning involved.

SkinVision



Lets you confirm weather a skin condition can be skin cancer or not.

Since the publication of the last systematic review of smartphone applications, a study on a smartphone application called SkinVision reported improved results for an algorithm trained on more than 130,000 images by more than 30,000 users.

Over the past years, SkinVision has made great progress towards developing an application that is significantly reliable in recognizing [dangerous skin lesions](#).

SkinVision started off with a 'rule-based' system which went through every picture and checked skin lesions for certain characteristics to determine risk. Even though this algorithm has helped them detect the risk of thousands of dangerous lesions, they are continuously looking to improve its accuracy.

They have trained the SkinVision algorithm with large quantities of images which were previously assessed by our team of dermatologists.

The algorithm learns which lesions are dangerous and which ones are not. They continuously train and improve our algorithm with new sets of images. From now on, all the

pictures submitted through the SkinVision application go through this algorithm.

It is common for doctors to ask a second opinion, and so at this moment, every photo is also reviewed by our in-house dermatologists and image recognition experts. They have set up this process to assist the algorithm to become more accurate and to make sure that their dermatologists agree with the risk indication.

The best part, however, is that we are training our algorithm to become on a par with the best dermatologists.

Picasa



Using facial Recognition to identify your friends and you in a group picture.

Google Photos



Classify your photo into Mountains, sea etc.

Deepmind



Achieved superhuman level playing Game Atari.

Facebook



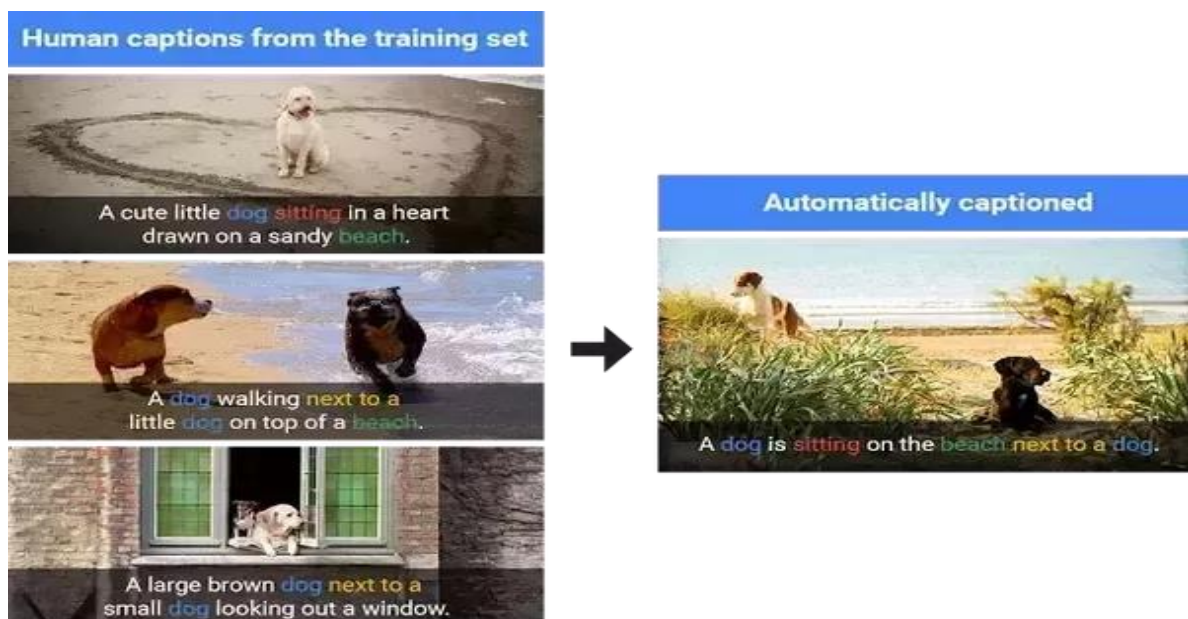
Preventing Suicide

Around the world, suicide is the second leading cause of death for 15 to 29-year olds.

Thankfully, Facebook can now help [prevent suicides through the use of AI](#). AI can signal posts of people who might be in need and/or perhaps driven by suicidal tendencies. The AI uses machine learning to flag key phrases in posts and concerned comments from friends or family members to help identify users who may be at risk. Analyzing human nuance as a whole is quite complex, but AI is able to track it the context and understand what is a suicidal pattern and what isn't. It's great to see that Facebook and other social media sites are doing their part to help with this issue.

Anyways, main implication of image captioning is automating the job of some person who interprets the image (in many different fields).

1. Probably, will be useful in cases/fields where text is most used and with the use of this, you can infer/generate text from images. As in, use the information directly from any particular image in a textual format automatically..
2. There are many NLP applications right now, which extract insights/summary from a given text data or an essay etc. The same benefits can be obtained by people who would benefit from automated insights from images.
3. A slightly (not-so) long term use case would definitely be, explaining what happens in a video, frame by frame.
4. Would serve as a huge help for visually impaired people. Lots of applications can be developed in that space.
5. Social Media. Platforms like facebook can infer directly from the image, where you are (beach, cafe etc), what you wear (color) and more importantly what you're doing also (in a way). See an example to understand it better.



5. Difference between Deep Learning and Machine Learning

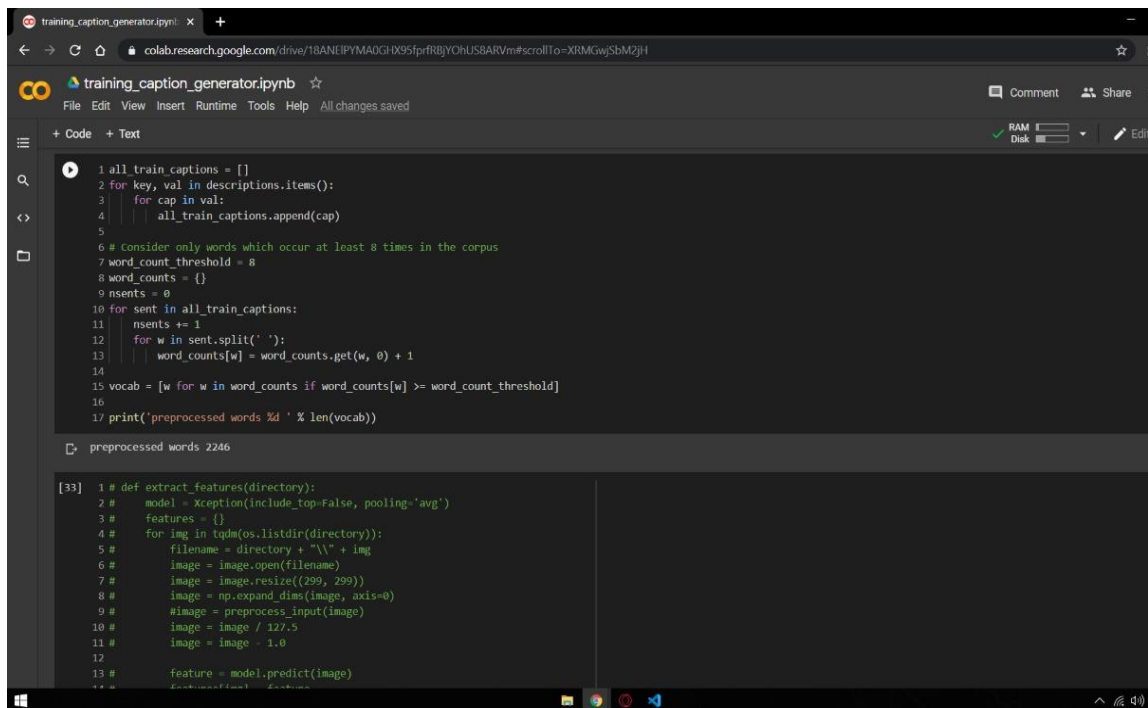


In practical terms, deep learning is just a subset of machine learning. In fact, deep learning technically *is* machine learning and functions in a similar way (hence why the terms are sometimes loosely interchanged). However, its capabilities are different.

While basic machine learning models do become progressively better at whatever their function is, they still need some guidance. If an AI algorithm returns an inaccurate prediction, then an engineer has to step in and make adjustments. With a deep learning model, an algorithm can determine on its own if a prediction is accurate or not through its own neural network.

Let's go back to the flashlight example: it could be programmed to turn on when it recognizes the audible cue of someone saying the word "*dark*". As it continues learning, it might eventually turn on with any phrase containing that word. Now if the flashlight had a deep learning model, it could figure out that it should turn on with the cues "*I can't see*" or "*the light switch won't work*," perhaps in tandem with a light sensor. A deep learning model is able to learn through its own method of computing—a technique that makes it seem like it has its own brain.

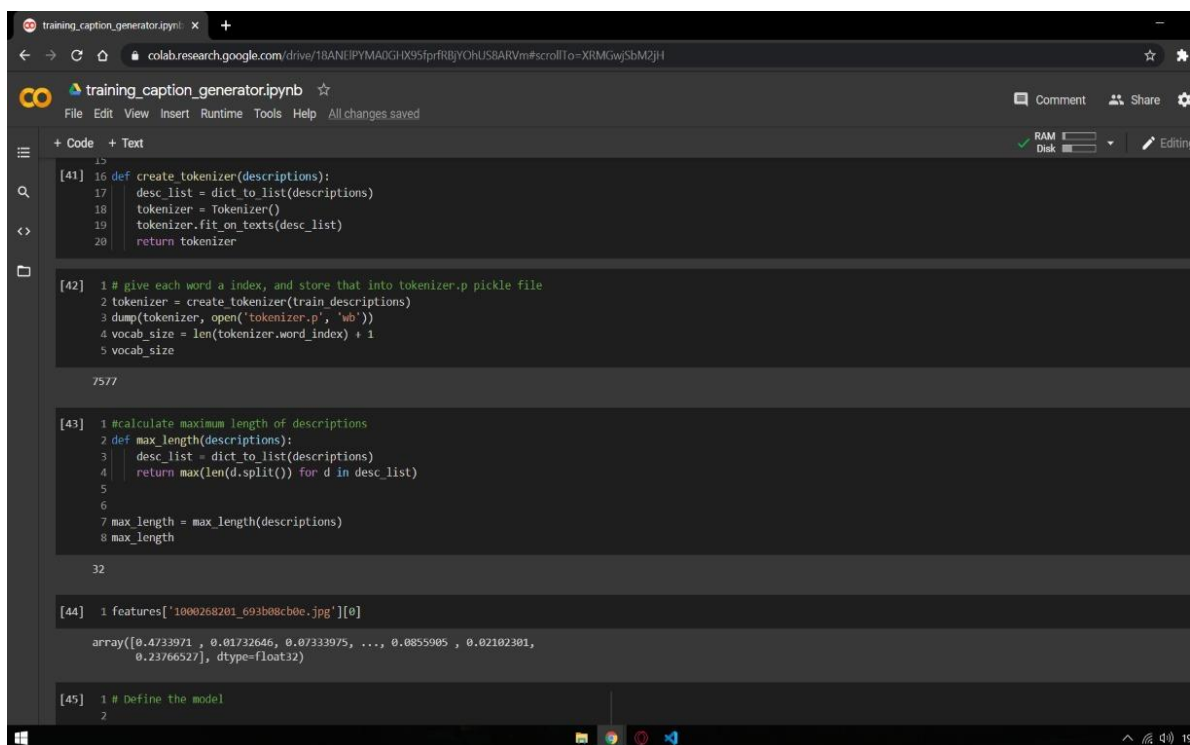
6. CODE SNAPSHOTS



```
1 all_train_captions = []
2 for key, val in descriptions.items():
3     for cap in val:
4         all_train_captions.append(cap)
5
6 # Consider only words which occur at least 8 times in the corpus
7 word_count_threshold = 8
8 word_counts = {}
9 nsents = 0
10 for sent in all_train_captions:
11     nsents += 1
12     for w in sent.split(' '):
13         word_counts[w] = word_counts.get(w, 0) + 1
14
15 vocab = [w for w in word_counts if word_counts[w] >= word_count_threshold]
16
17 print("preprocessed words %d * %d len(vocab)" % (len(all_train_captions), len(vocab)))
```

preprocessed words 2246

```
[33] 1 # def extract_features(directory):
2     model = Xception(include_top=False, pooling='avg')
3     features = []
4     for img in tqdm(os.listdir(directory)):
5         filename = directory + "/" + img
6         image = image.open(filename)
7         image = image.resize((299, 299))
8         image = np.expand_dims(image, axis=0)
9         #image = preprocess_input(image)
10        image = image / 127.5
11        image = image - 1.0
12
13        feature = model.predict(image)
```



```
[41] 16 def create_tokenizer(descriptions):
17     desc_list = dict_to_list(descriptions)
18     tokenizer = Tokenizer()
19     tokenizer.fit_on_texts(desc_list)
20     return tokenizer

[42] 1 # give each word a index, and store that into tokenizer.p pickle file
2 tokenizer = create_tokenizer(train_descriptions)
3 dump(tokenizer, open('tokenizer.p', 'wb'))
4 vocab_size = len(tokenizer.word_index) + 1
5 vocab_size

7577

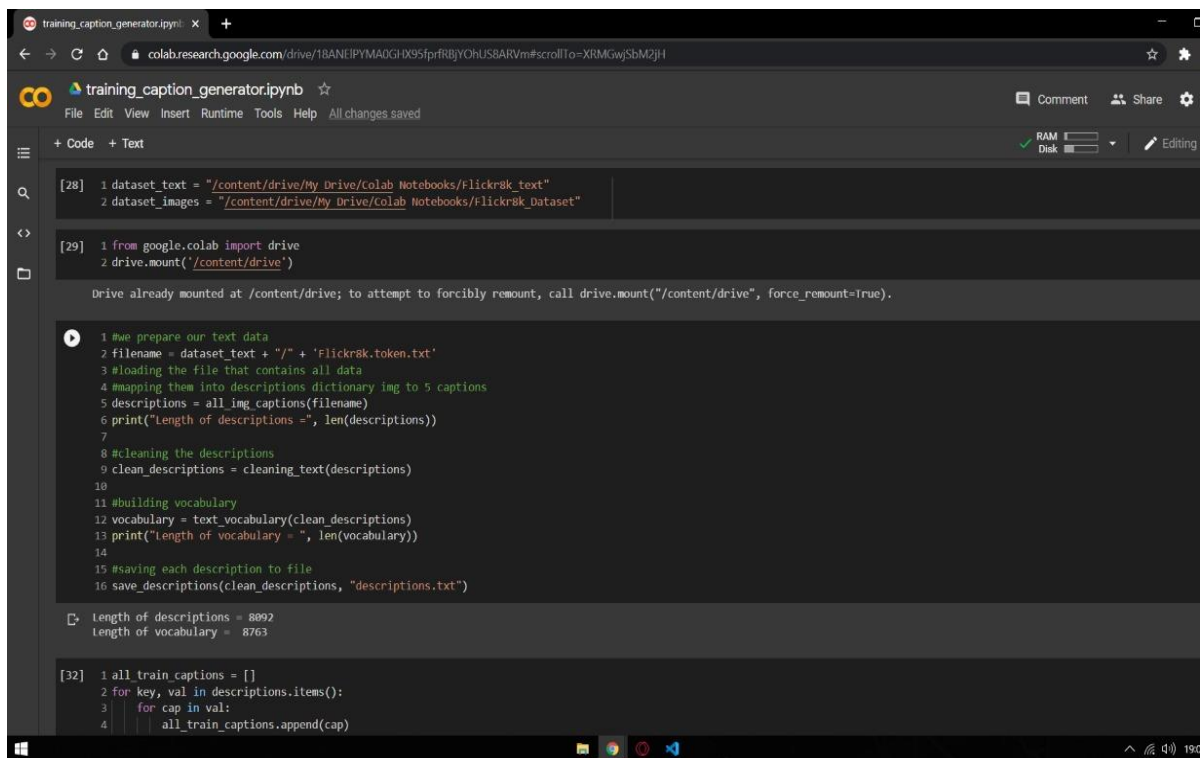
[43] 1 # calculate maximum length of descriptions
2 def max_length(descriptions):
3     desc_list = dict_to_list(descriptions)
4     return max(len(d.split()) for d in desc_list)
5
6
7 max_length = max_length(descriptions)
8 max_length

32

[44] 1 features['1000268201_693b08cb0e.jpg'][0]

array([[0.4733971, 0.01732646, 0.07333975, ..., 0.0855905, 0.02102301,
0.23766527], dtype=float32)]

[45] 1 # Define the model
2
```



```
[28] 1 dataset_text = "/content/drive/My Drive/Colab Notebooks/flickr8k_text"
      2 dataset_images = "/content/drive/My Drive/Colab Notebooks/Flickr8k_Dataset"

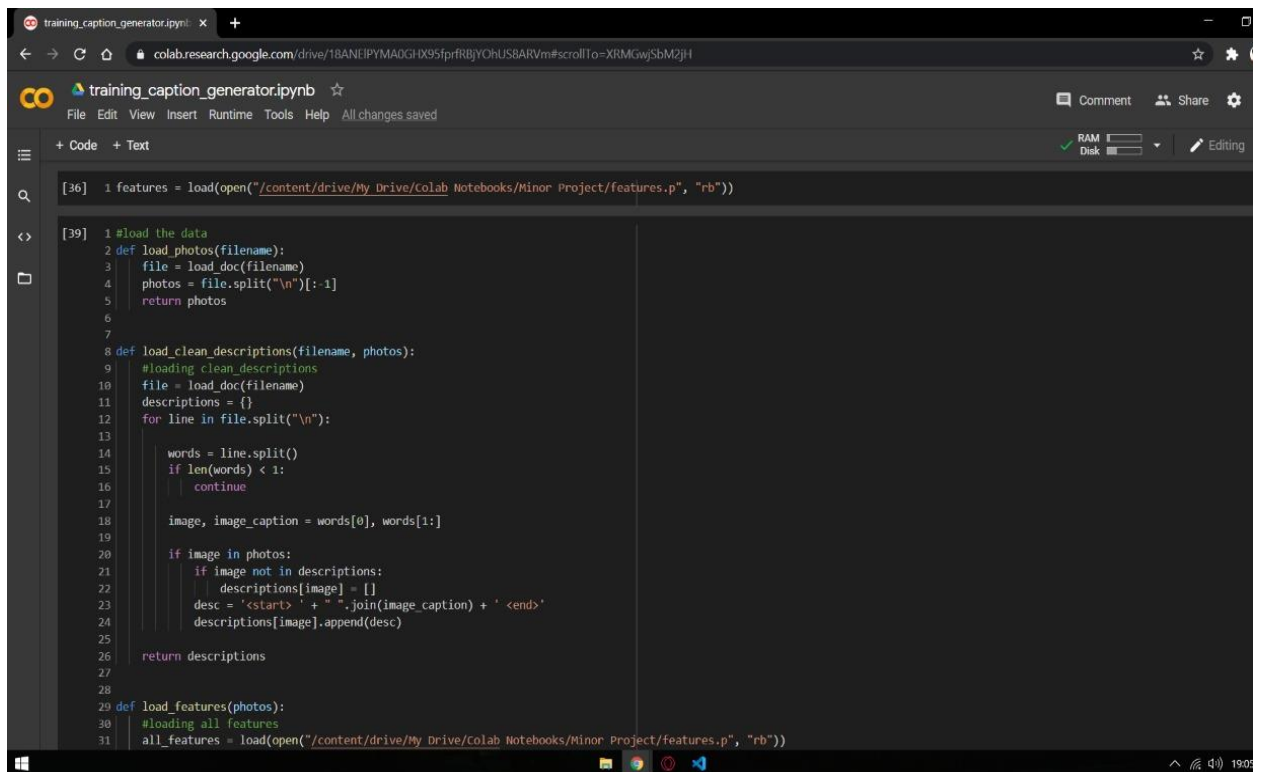
[29] 1 from google.colab import drive
      2 drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

1 #we prepare our text data
2 filename = dataset_text + "/" + 'Flickr8k.token.txt'
3 #loading the file that contains all data
4 #mapping them into descriptions dictionary img to 5 captions
5 descriptions = all_img_captions(filename)
6 print("Length of descriptions =", len(descriptions))
7
8 #cleaning the descriptions
9 clean_descriptions = cleaning_text(descriptions)
10
11 #building vocabulary
12 vocabulary = text_vocabulary(clean_descriptions)
13 print("length of vocabulary = ", len(vocabulary))
14
15 #saving each description to file
16 save_descriptions(clean_descriptions, "descriptions.txt")

Length of descriptions = 8892
Length of vocabulary = 8763

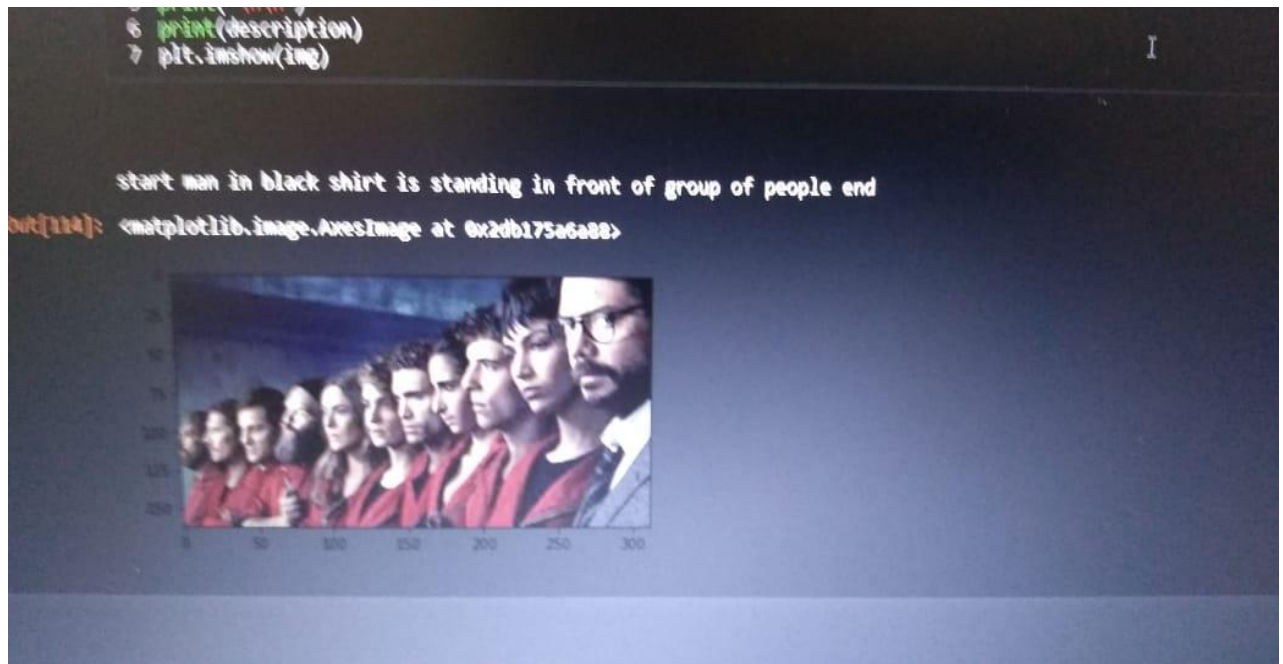
[32] 1 all_train_captions = []
      2 for key, val in descriptions.items():
      3     for cap in val:
      4         all_train_captions.append(cap)
```



```
[36] 1 features = load(open("/content/drive/My Drive/Colab Notebooks/Minor Project/features.p", "rb"))

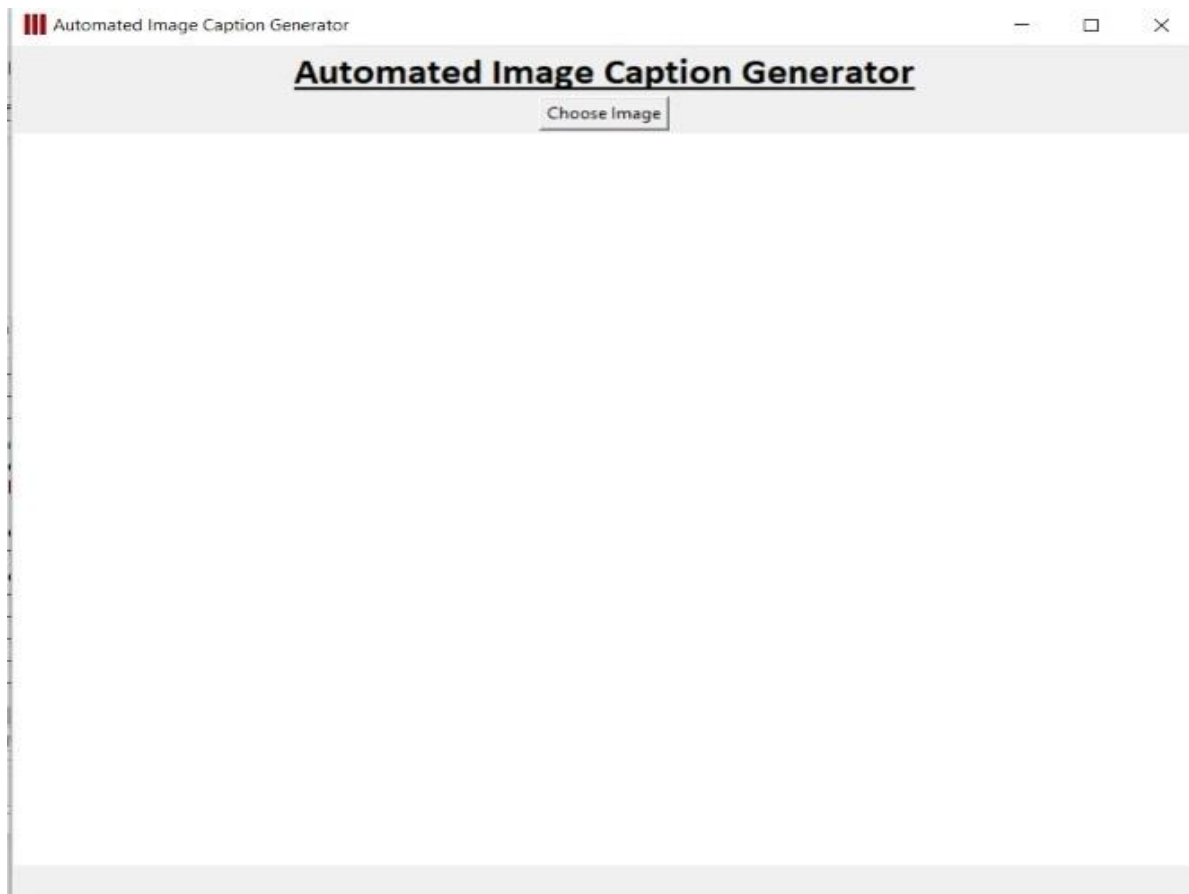
[39] 1 #load the data
      2 def load_photos(filename):
      3     file = load_doc(filename)
      4     photos = file.split("\n")[:-1]
      5     return photos
      6
      7
      8 def load_clean_descriptions(filename, photos):
      9     #loading clean descriptions
     10     file = load_doc(filename)
     11     descriptions = {}
     12     for line in file.split("\n"):
     13
     14         words = line.split()
     15         if len(words) < 1:
     16             continue
     17
     18         image, image_caption = words[0], words[1:]
     19
     20         if image in photos:
     21             if image not in descriptions:
     22                 descriptions[image] = []
     23             desc = '<start> ' + " ".join(image_caption) + ' <ends>'
     24             descriptions[image].append(desc)
     25
     26     return descriptions
     27
     28
     29 def load_features(photos):
     30     #loading all features
     31     all_features = load(open("/content/drive/My Drive/Colab Notebooks/Minor Project/features.p", "rb"))
```

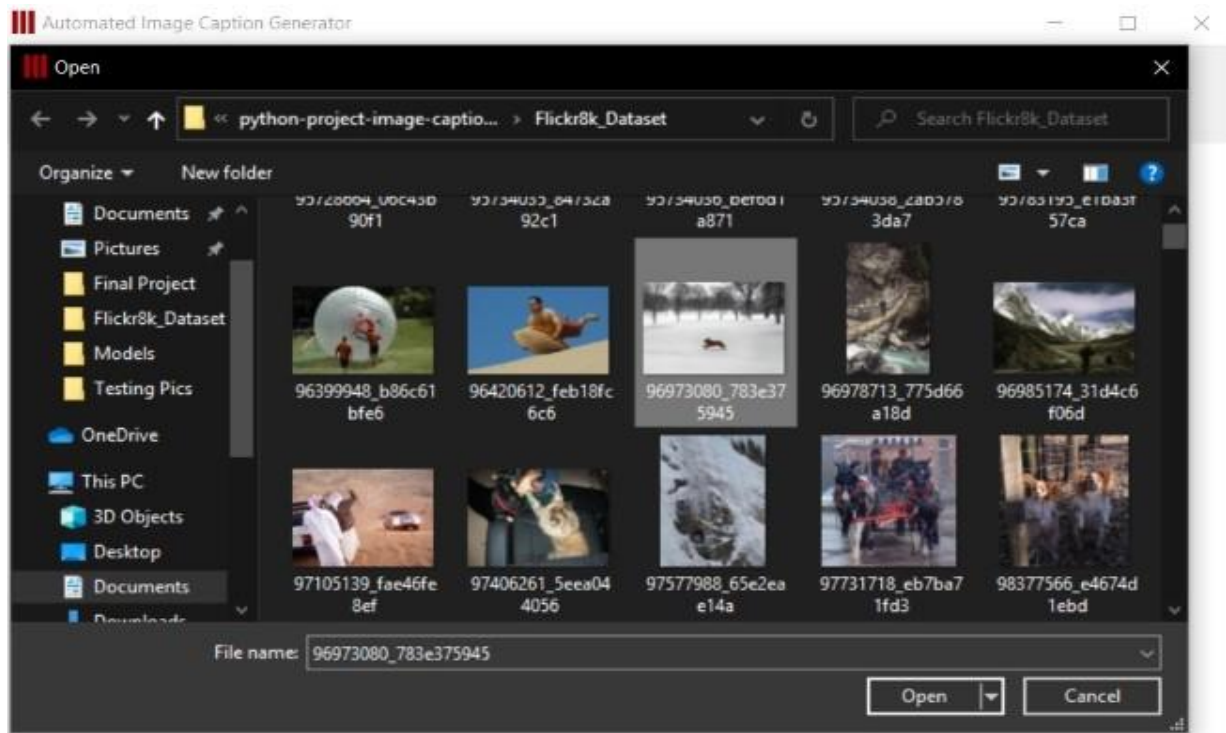

7.EXPERIMENTS AND RESULTS



Greedy: a woman in a tennis racket on the court .

UI SNAPSHOTS





Automated Image Caption Generator

Choose Image



Two dogs are running through the snow.

Automated Image Caption Generator

Choose Image



Man in red shirt and jeans is playing baseball.

Training the model with metrics Accuracy and MSE

Jupyter Training and Evaluating Xception Last Checkpoint: 11/28/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Code Notify Disabled

dropout_1 (Dropout)	(None, 32, 256)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	525312	dropout_1[0][0]
add_12 (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 256)	65792	add_12[0][0]
dense_2 (Dense)	(None, 7577)	1947289	dense_1[0][0]

=====

Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0


None

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

WARNING:tensorflow:From <ipython-input-11-3687afc1e067>:15: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.

6000/6000 [=====] - 1460s 243ms/step - loss: 4.5157 - accuracy: 0.2362 - mse: 1.1719e-04
6000/6000 [=====] - 1308s 218ms/step - loss: 3.6644 - accuracy: 0.2930 - mse: 1.1080e-04 1s - loss: 3.6646 - accuracy: 0.2930 -
6000/6000 [=====] - 1078s 180ms/step - loss: 3.3723 - accuracy: 0.3121 - mse: 1.0839e-04
6000/6000 [=====] - 1058s 176ms/step - loss: 3.1986 - accuracy: 0.3245 - mse: 1.0682e-04
6000/6000 [=====] - 1104s 184ms/step - loss: 3.0784 - accuracy: 0.3339 - mse: 1.0563e-04
6000/6000 [=====] - 1032s 172ms/step - loss: 2.9898 - accuracy: 0.3409 - mse: 1.0474e-04
6000/6000 [=====] - 1019s 170ms/step - loss: 2.9205 - accuracy: 0.3462 - mse: 1.0401e-04
6000/6000 [=====] - 967s 161ms/step - loss: 2.8656 - accuracy: 0.3510 - mse: 1.0336e-04
6000/6000 [=====] - 954s 159ms/step - loss: 2.8222 - accuracy: 0.3548 - mse: 1.0282e-04
6000/6000 [=====] - 940s 157ms/step - loss: 2.7829 - accuracy: 0.3593 - mse: 1.0229e-04


BLEU SCORE

jupyter training_caption_generator Xception - Copy Last Checkpoint: 8 hours ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [88]: 1 mod=load_model("../New folder/models/model_9.h5")
executed in 668ms, finished 23:17:07 2020-12-03

In [90]: 1 evaluate_model(mod, test_descriptions, test_features, test_tokenizer, 32)
executed in 8m 5s, finished 23:36:59 2020-12-03

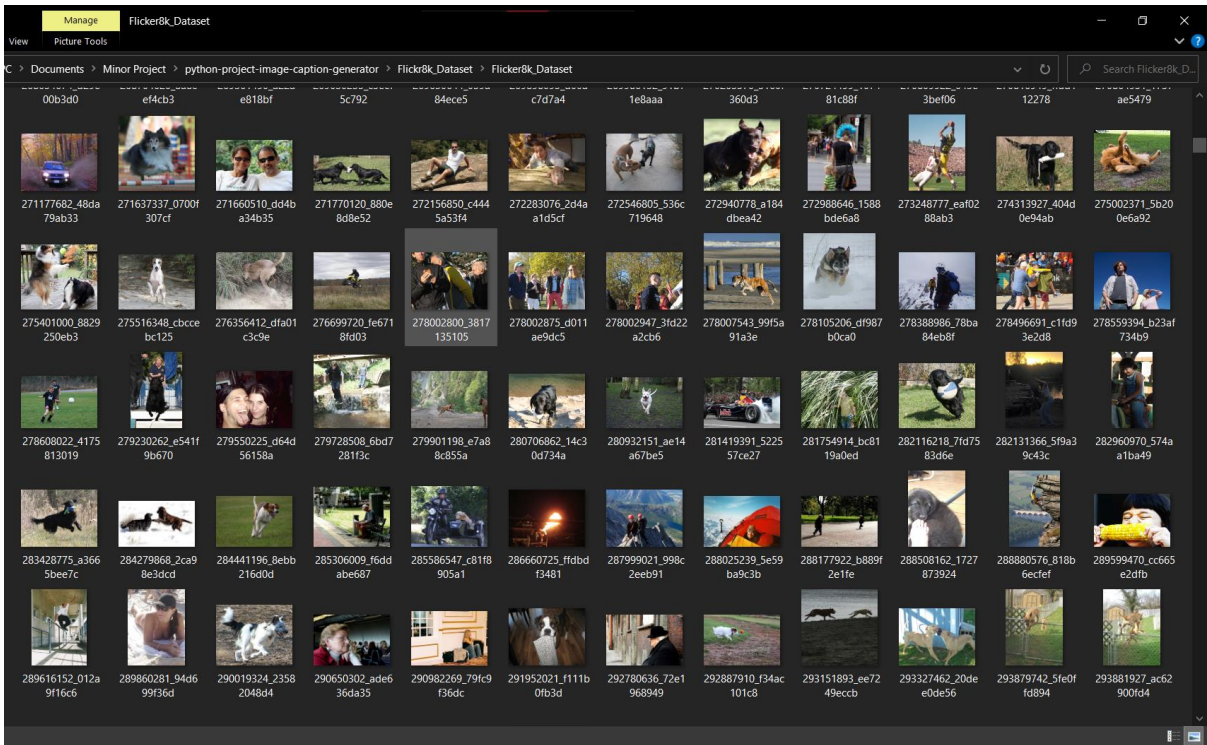
100%  1000/1000 [08:03<00:00, 2.07/s]

BLEU-1: 0.271486
BLEU-2: 0.036212
BLEU-3: 0.000000
BLEU-4: 0.000000

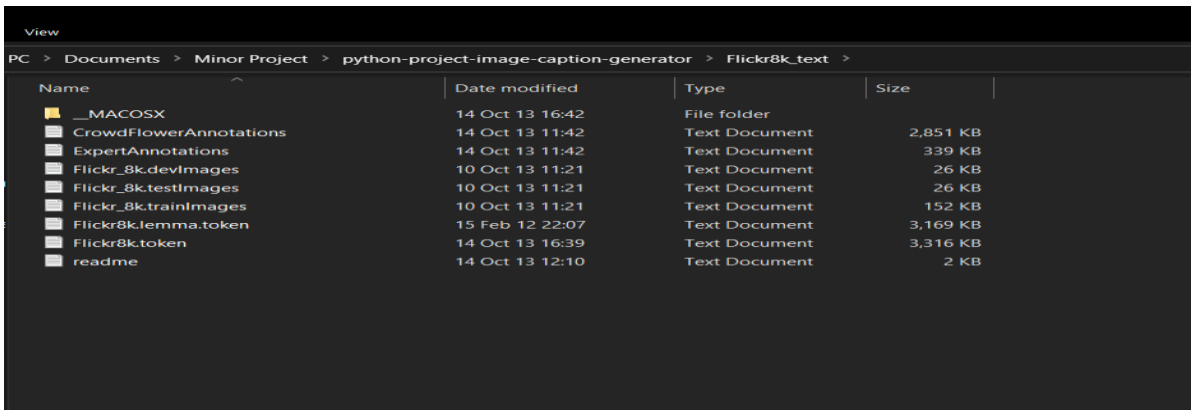
In [24]: 1 def dict_to_list(descriptions):
2 all_desc = []
3 for key in descriptions.keys():
4 [all_desc.append(d) for d in descriptions[key]]
5 return all_desc
6
7 from keras.preprocessing.text import Tokenizer
8
9 def create_tokenizer(descriptions):
10 desc_list = dict_to_list(descriptions)
11 tokenizer = Tokenizer()
12 tokenizer.fit_on_texts(desc_list)
13 return tokenizer
executed in 10ms, finished 22:42:21 2020-12-03

In [29]: 1 tokenizer = create_tokenizer(train_descriptions)
2 dump(tokenizer, open('tokenizer.p', 'wb'))
3 vocab_size = len(tokenizer.word_index) + 1
4 vocab_size
executed in 758ms, finished 22:43:55 2020-12-03

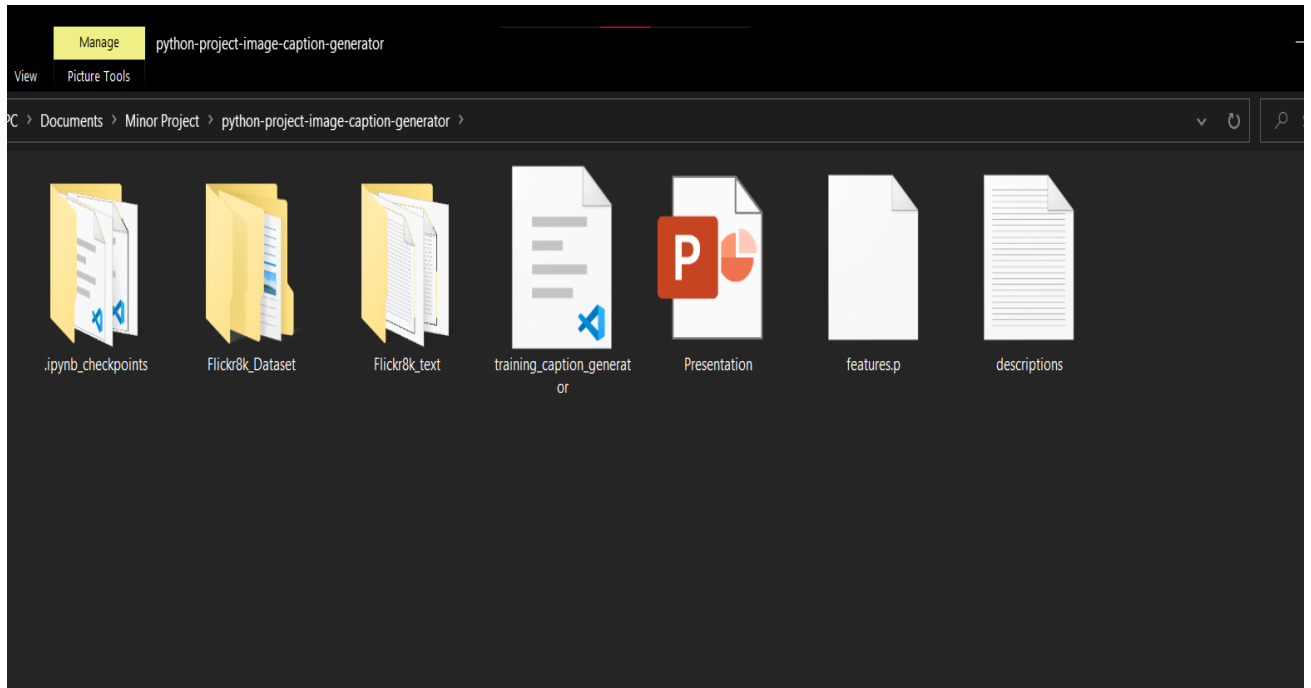
DATASET AND DIRECTORY SNAPSHOTS



Screenshot of Image Dataset



Screenshot of Text Dataset



Screenshot of Main Directory

8. Conclusion and Future Scope

Thanks a lot if you have reached here. This is our first attempt in making project report so we expect the readers to be a bit generous and ignore the minor mistakes we might have Made.

Our described model is based on a CNN that encodes an image into a compact representation, followed by an RNN that generates corresponding sentences based on the learned image features. It worked quite well when tested on several images. The captions it generated for the images were quite accurate. But the source of input image also played an important role in feature extraction and hence caption generation. Certain images are not well recognized and we found out that there is, still some scope of improvement. We Got accuracy of 35.93%.

Of course this is just a first-cut solution and a lot of modifications can be made to improve this solution like:

- Using a **larger** dataset.
- Changing the model architecture, e.g. include an **attention** module.
- Doing more **hyper parameter tuning** (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.).
- Use the cross validation set to understand **overfitting**.

9. References and Citations

[1] Brett Grossfeld(2020, 23 January). Deep learning vs machine learning: a simple way to understand the difference: <https://www.zendesk.com/blog/machine-learning-and-deep-learning/#:~:text=To%20recap%20the%20differences%20between,intelligent%20decisions%20on%20its%20own>

[2] Mike Driscoll. Jupyter Notebook: An Introduction: <https://realpython.com/jupyter-notebook-introduction/>

[3] Haoran Wang. Research Article. An Overview of Image Caption Generation Methods: <https://www.hindawi.com/journals/cin/2020/3062706/>

[4] SkinVision In Articles. How Machine Learning Technology Detects Skin Cancer: <https://www.skinvision.com/articles/how-machine-learning-detects-skin-cancer/>

[5] Kambria In Articles(2019, 5 June). How facebook Uses Artificial Intelligence: <https://kambria.io/blog/how-facebook-uses-artificial-intelligence/#:~:text=AI%20can%20signal%20posts%20of,who%20may%20be%20at%20risk>.

[6] Moses Soh. Research Article. Department of Computer Science. Stanford University. Learning CNN-LSTM Architectures for Image Caption Generation: <https://cs224d.stanford.edu/reports/msoh.pdf>

[7] Sanam Malhotra. Oodles AI(2020, 8 April). Building and Deploying an AI-powered Image Caption Generator: <https://artificialintelligence.oodles.io/blogs/ai-powered-image-captiongenerator/#:~:text=The%20advent%20of%20machine%20learning,better%20sense%20of%20their%20surroundings>.

[8] Jason Brownlee(2019, 3 September). How to develop a Deep learning photo caption generator: <https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/>

[9] Machine learning Handouts of College

[10] Dr. Vinayak D.Shinde, Mahiman P.Dave, Anuj M.Singh, Amit C.Dubey. Research Journal. International Research Journal of Engineering and Technology(IRJET)(2020, April). Image Caption Generator using Big Data and Machine Learning: <https://www.irjet.net/archives/V7/i4/IRJET-V7I41167.pdf>

10. STUDENTS DETAILS

Bhavya Tyagi

Enrollment Number- 181B070

Email-ID- 181b070@juetguna.in

Bhawana Mishra

Enrollment Number- 181B071

Email-ID- 181b071@juetguna.in

Chandan Kumar

Enrollment Number- 181B074

Email-ID- 181b074@juetguna.in