

Image Captioning Generator with CNN and LSTM

Nitin Kumar¹, Shubham Agrawal², Nitin Solanki³, Samriddh Sharma⁴

nitin.kumar.co19@nsut.ac.in¹, shubham.agrawal.co19@nsut.ac.in², nitin.solanki.co19@nsut.ac.in³, samriddh.co19@nsut.ac.in⁴

Department of Computer Engineering

Netaji Subhas University of Technology, India

Azad Hind Fauj Marg Sector - 3, Dwarka New Delhi – 110078

Abstract :

Image Captioning generates an automated short and basic statement describing the image content. Captions for a picture on the internet can help to generate detailed authentic photographs quickly for exploring and indexing. The machines need to be trained such that they can comprehend the image content and can generate captions almost accurately with limited human knowledge. This is a very tedious and interesting task and can be solved by generating simple sentences known as captions using a neural network and is associated with many issues like generating captions only for the seen images, inaccurate captions etc. In this paper, we have proposed a 2 - staged Image Captioning model prototype consisting of a combination of Deep Neural Network algorithms which is able to generate more accurate and precise captions. Convolutional (CNN) and Long Short-Term Memory (LSTM). The proposed model is able to overcome the problems that arise using Traditional CNN and RNN algorithms. We have trained and tested our model using the Flickr_8k Dataset.

In our research paper, deep learning is used for generating the image description. Image description enables the process of describing the content using an image. Detection of objects and actions in the input image help in the generation of image description. Bottom-up and top-down approaches are the two main approaches used in image description. Bottom-up approach generates contents in an input image, and then combines them into a caption. Top-down approach generates a semantic representation of an input image which is decoded using various architectures like recurrent neural networks into a caption.

Keywords : CNN, RNN-LSTM, Deep Learning, Feature extraction, Feature Vector, Filters, Captions

I. Introduction

Image captioning is just a short description created by simply looking at the image. In this job, a machine is fed an input image and, depending on the intelligence and training provided, the model generates a simple caption that very well accurately summarises the entire image's content in a human-readable format (Natural Language). When a machine needs to create a caption for unseen or untrained photos, such jobs become increasingly difficult. Before creating a text or description, a model attempts to break down a picture into items and classify these things. The goal of picture captioning is to generate natural language and simple captions that properly reflect the image content.

All items and their relationships should be accurately portrayed in this job. Traditional caption creation algorithms, which combine Convolutional and Recurrent networks, have a number of flaws, including gradient vanishing, inaccuracy in identifying objects and their relationships, and the development of captions only for viewed pictures, among others. An Automatic Image Captioning Model (AIM) is a version of the classic approach that uses sophisticated Convolutional and Long Short-Term Memory Deep Neural Network algorithms

(CNN and LSTM) to address the issues that emerge while utilising the standard method of captioning. There are two steps to the Model: The Convolutional algorithm is used in the first step, while Long Short-Term Memory is used in the second stage.

Image/picture is the first stage's input. The suggested system model additionally emphasises the most descriptive captions for the visual scenario. The suggested system model feeds an image vector to the first stage, known as the encoder, which has already been pre-processed and then delivered as input to Stage 1. Various convolutional layers are applied to the vector at this step, which retrieves relevant characteristics from the supplied vector before passing it to the next stage. The image vector is then transferred to the following step, the Decoder stage, after a number of convolutional layers/operations have been applied to it.

Stage 2 generates captions by processing the picture vector provided by Stage 1 in a linear manner. In Stage 2, the technique employs the LSTM algorithm, which is a more advanced form of the recurrent neural network (RNN) that aids in the resolution of the gradient

explosion problem. The LSTM has an advantage since it employs multiple memory gates to control the flow of data in Stage 2. It also offers the benefit of storing data for a longer period of time and with fewer dependencies. For the supplied input image, Stage 2 generates a sequentially decoded basic language statement or captions.

Models for Image Caption Generators are based on encoder-decoder architecture, which generates valid and acceptable captions using input vectors. This model combines natural language processing and computer vision in one model. It's a job of identifying and evaluating the image's context before explaining everything in natural language like English. Our approach is based on two basic models: CNN (Convolutional Neural Network) and RNN-LSTM (Recurrent Neural Network-LSTM). CNN is utilised as an encoder in the derived application to extract features from the snapshot or image, and RNN-LSTM is used as a decoder to organise the words and generate captions.

One of the key applications of this model is in self-driving automobiles, for example, it can describe the area around the car.

Second, it might help blind people in a variety of ways, such as translating scenes to captions and then to audio, using CCTV cameras to raise warnings if any dangerous behaviour is detected while describing the scene, editing suggestions, social media postings, and so on.

The main contribution demonstrated in this paper is to research, develop and demonstrate a prototype of a new Image Captioning Model that has a) high accuracy, b) Low Error rate, c) easy to deploy

The presented article is organized as follows :

Section 2 mentions the research work carried out by fellow researchers. Section 3 discusses the methodology which includes dataset requirements, libraries used, 2-stage architecture (CNN and LSTM) and their working, System architecture, and Project File structure. Section 4 discusses the result analysis and it includes Performance Parameters, Training steps, Testing steps, Testing results. Section 5 includes the 10 fold cross validation results and the flow chart showing the working of the model. Section 6 is about the conclusion of the project. Section 7 discusses the future scope and applications of this project. Section 8 mentions the references that have been used to create this research paper.

II. Literature Work

The research papers used as reference for developing the Image Captioning Model are as follows :

Image Caption using CNN & LSTM by Ali ashraf Mohamed [1]

The author of this study examined the Xception and VGG16 image feature extraction algorithms in order to create an Image caption generator. The Flickr_8k dataset was utilised, which had roughly 8000 pictures, 1000 of which were used for validation and testing and the remainder for training. For deep learning, the model was developed in Python using keras 2.0, with the tensorflow library serving as the backend for the keras framework. The model received a BLEU score of 0.55 for Xception and 0.46 for VGG16, indicating that the Xception approach outperforms VGG16 in terms of feature extraction for picture captioning.

Image Caption Generator using CNN-LSTM by Preksha Khant, Vishal Deshmukh, Aishwarya Kude, Prachi Kiraula [2]

The authors suggested an encoder-decoder architecture in which CNN served as the encoder, extracting features from pictures, while RNN-LSTM served as the decoder, arranging words and creating captions. The flickr_8k dataset was utilised, and a BLEU score of up to 0.68 was obtained. The model was implemented using Tensorflow, Keras, and Pillows. The VGG16 algorithm has been applied in conjunction with CNN to be able to extract features.

Image Caption Generation Using CNN and LSTM by UjwalaBhoga, V.Aravind, G.Sreeja, MohdArif [3]

A CNN is used in the suggested approach to extract information from a picture. It was pre-trained with the Inception-v3 model. With CNN data, LSTM was utilized to provide a visual description. Image properties were encoded using transfer learning. In order to encode the text sequence, each word was mapped to a 200-dimensional vector. Following the input layer, the data was mapped using a different layer termed the embedding layer. The caption was created in two ways: greedy search and beam search. The dataset employed in the proposed model was ImageNet. To produce satisfactory results, beam search was conducted with beams of size 3 and 5. The model performed well when it came to detecting actions, but it made errors when recognising colours. The BLEU accuracy achieved in this paper is 61%

An Advanced Image Captioning using combination of CNN and LSTM by Priyanka Raut and Rushali A Deshmukh [4]

There were four modules in the proposed System Architecture. The Input Picture is used as an input to the Image Based Module, which produces a feature vector of the input image using the CNN algorithm's Convolutional and Pooling layer. A ReLu layer is used after each Convolutional layer, followed by a Pooling layer to reduce the size of the feature vector before sending it to the next model. Only the feature vector was required, hence the last layer of CNN, fully connected Network, was left out of the model. The Feature extractors are Convolutional and Pooling layers, and the Classifier is the Fully Connected Network. The encoded characteristics vector is sent to the Language Based Module, which uses the LSTM (Long Short Term Memory) method to decode it into a natural language caption. The Captioning Model focuses on objects, colour, actions, and object relationships.

Image-Text Surgery: Efficient Concept Learning in Image Captioning by Generating Pseudo Pairs by Kun Fu, Jin Li, Junqi Jin, and Changshui Zhang [5]

Image-Text Surgery for the Captioning of the Image, a methodology suggested in the study, employed its own developed pseudo image and text pairings approach to construct a caption. Various effective learning strategies based on pseudo pairings were discussed in the study. The pseudo pair is an image and text pair created with the MSCOCO subset dataset, and it has its own syntax known as seed syntax. To prevent human interference, the pseudo pair idea was employed to avoid any human tagged data. The model was also evaluated on a subset of the MSCOCO dataset. The results reveal a considerable improvement in the approach. Their methodology is sound, and it produces better outcomes than traditional approaches.

III. Proposed Methodology

A. Dataset Requirements :

For building the image caption generator, we have taken the Flickr 8k dataset. This dataset comprises a diverse assortment of photos depicting a variety of settings and scenes. The Flickr 8k dataset contains 8091 photos, each with five captions. Each picture has its own set of proportions.

a) Type of Dataset :

The Flickr_8K dataset has been used for the image caption generator. Other large datasets, such as Flickr 30K and MSCOCO, are available, but training the network on these might take weeks, so we'll stick with the smaller Flickr 8k dataset. The benefit of a large dataset is that we can create more accurate and reliable models.

b) About Dataset :

Name of dataset : Flickr_8K dataset

Source of dataset : [Kaggle](#)

Description of dataset : Contains 8091 photographs in JPEG format.

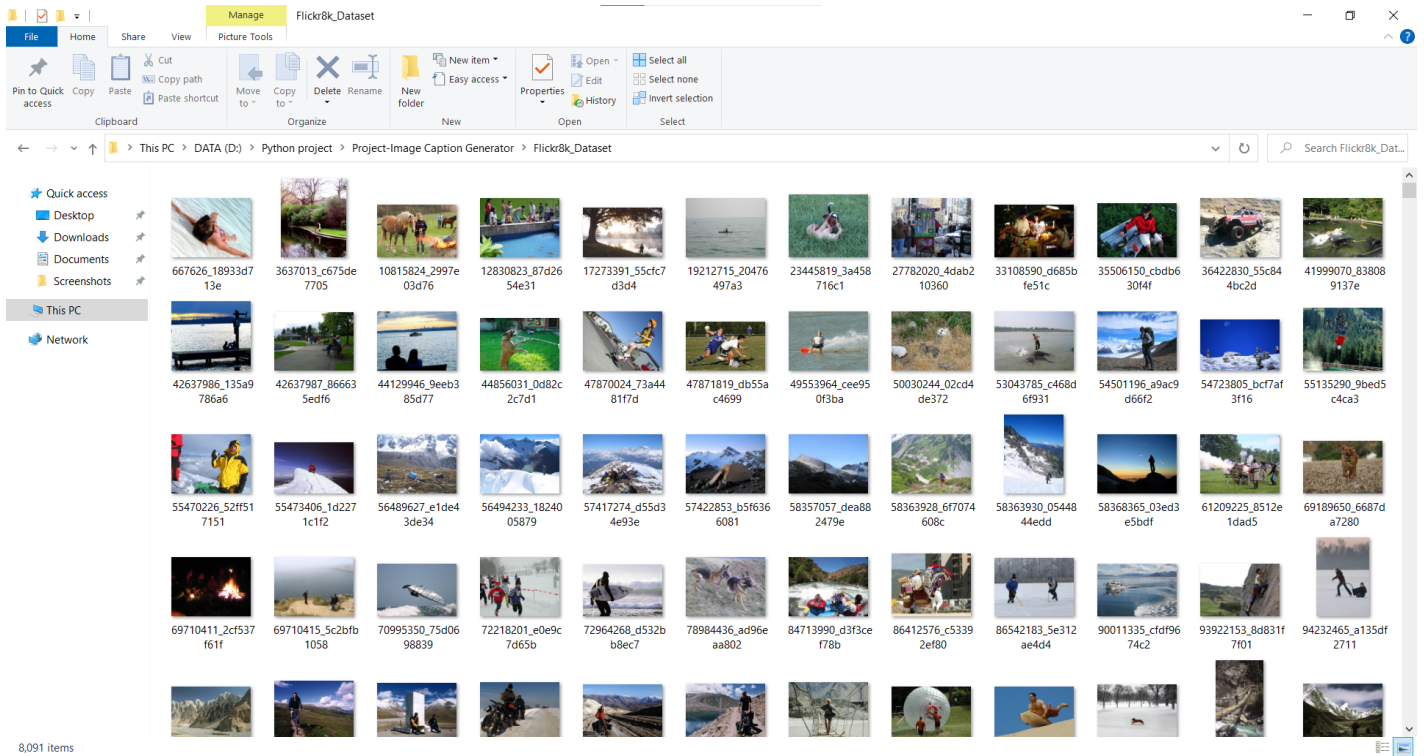
c) Download the Dataset

- [Flickr_8K dataset](#)

There are **8091** JPEG images in this collection. This folder contains 8091 photos of various sizes, shapes, and colours. This dataset is **1 GB** in size.

- [Flickr_8K Text](#)

A Flickr8k.token.txt file in the Flickr_8k text folder includes **5 captions** per picture for training the proposed model in the form of a key-value pair, where the key is the image's unique id and the value is the description for the image. This file is **2 MB** in size. This is the primary file in our dataset, and it comprises the image names and descriptions, separated by newline("\n").



B. Libraries Used :

Tensorflow : An open-source toolkit for deep learning that works with Python and other frameworks.

Keras : An open-source Python package that allows deep learning models to be evaluated.

Pillow : A Python Imaging Library (PIL) that provides functionality for opening, manipulating, and storing pictures to the Python programming language.

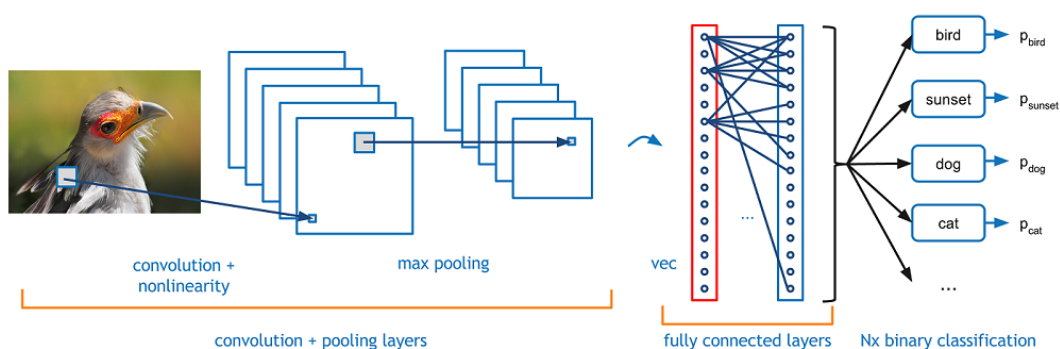
Numpy : The Numpy library is used to work with arrays.

Matplotlib : A Python library for creating static and animated graphics.

Tqdm : A Python package that allows you to create Progress Meters or Progress Bars.

C. 2 - Stage Architecture

a) CNN Architecture : (Image based Model)

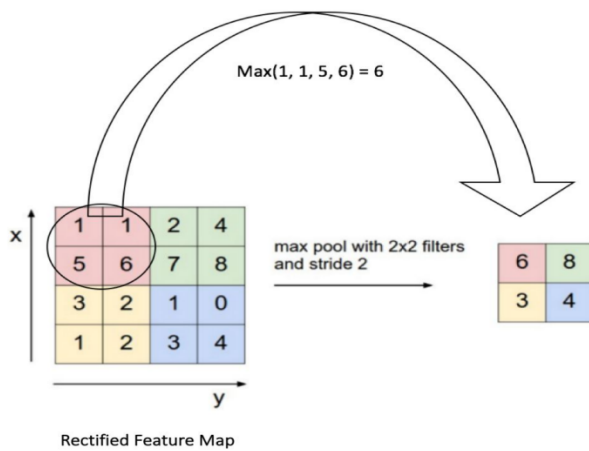


When it comes to interpreting huge photos and videos, a pure primitive neural network, in which all neurons in one layer merge with all neurons in the next layer, is inefficient. The range of limitations using an acceptable neural system for a standard size picture with numerous picture pieces called pixels and 3-tone colours (RGB i.e. red colour, green colour, blue colour) will be in the thousands, which might lead to overfitting.

CNN uses a 3D arrangement in which each adjustment of neurons breaks down a small region or "highlight" of the picture to confine effective amounts of constraints & recognition of the neural system on important pieces of the picture. Rather than all neurons skipping to the next brain layer, each group of neurons spends a substantial amount of time differentiating one aspect of the image, such as a nose, left ear, mouth, or leg. The final result is a point of scope, demonstrating how plausible each of the skills is chosen as a member of the class.

The primary reason for using CNN is that it is the only algorithm that accepts photographs as input and draws a feature map based on the input pictures, i.e. categorising each pixel based on similarity and differences.

The CNN identifies the pixels and generates a feature map, which is a matrix. A feature map is a grouping of comparable pixels into a distinct category. These matrices are crucial in determining the essence of the object in the input image.



Working of CNN

Convolutional neural networks are a type of deep neural network that can handle data in the form of a 2D matrix. Images are easily represented as a 2D matrix, and CNN is an excellent tool for working with them.

CNN is used to classify images and find out what they depict such as a plane, a bird etc

It scans photos from left to right and top to bottom in order to extract essential elements, which it then combines to categorise the images. It can handle photos that have been rotated, resized, then rotated again, as well as changes in perspective.

To put it another way, CNN takes the image's features and transforms it to a lesser dimension without losing its properties.

b) LSTM Architecture : (Language based Model)

The LSTM design is fairly basic; it comprises three primary gates that retain data for a longer amount of time and assist in solving problems that RNNs were unable to handle.

The LSTM's three main gates are as follows:

1. Forget gate — the forget gate's principal function is to filter data, that is, to remove any data that will not be needed in the future to complete a task. This gate is in charge of the LSTM's overall performance and data optimization.

2. Input gate — the LSTM begins with this gate, i.e. the input gate. This gate receives user input and passes the information on to other gates.

3. Output gate — This gate is used for displaying the appropriate result.

Working of LSTM

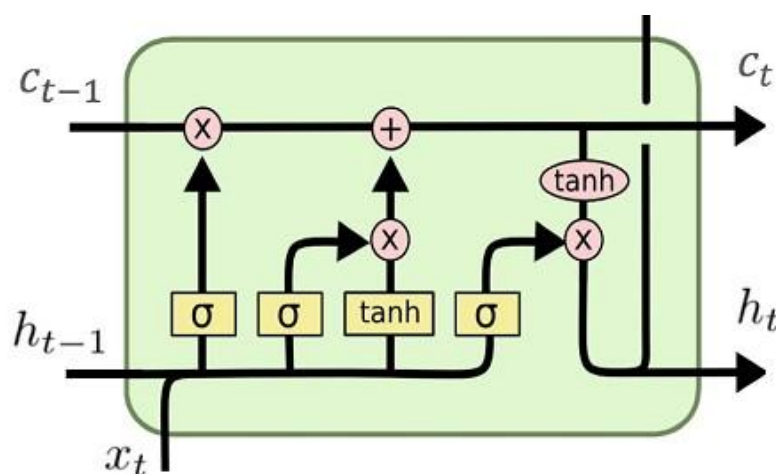
Long short term memory (LSTM) is a form of RNN (recurrent neural network) that is particularly well adapted to sequence prediction challenges.

We can guess what the following word will be based on the preceding paragraph.

It has outperformed regular RNNs in terms of overcoming the constraints of RNNs with short term memory.

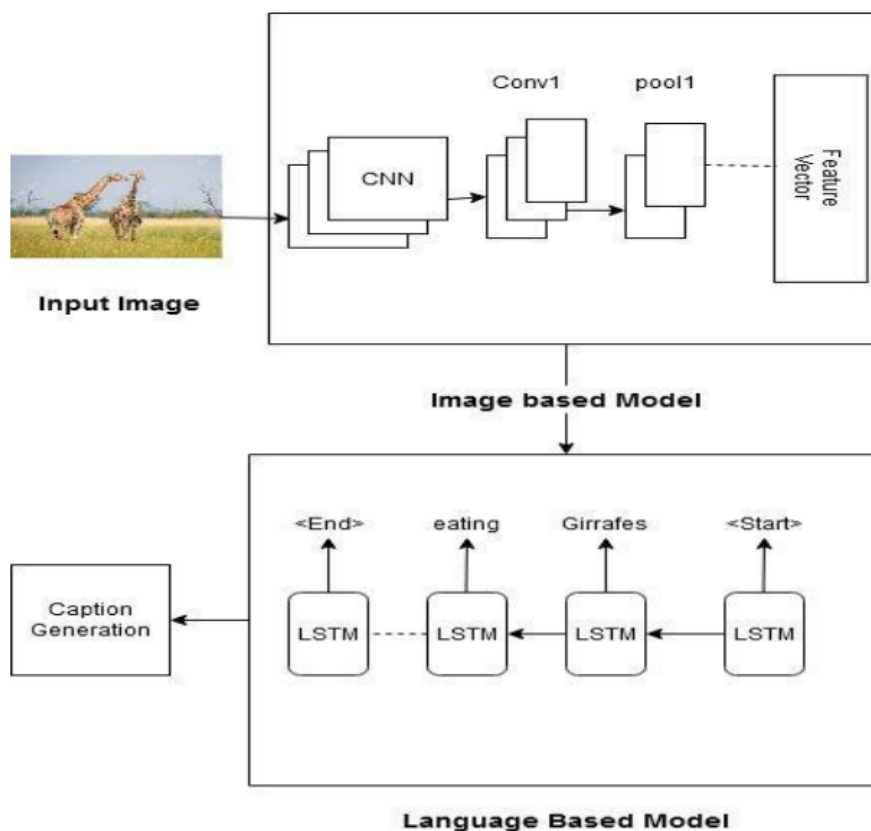
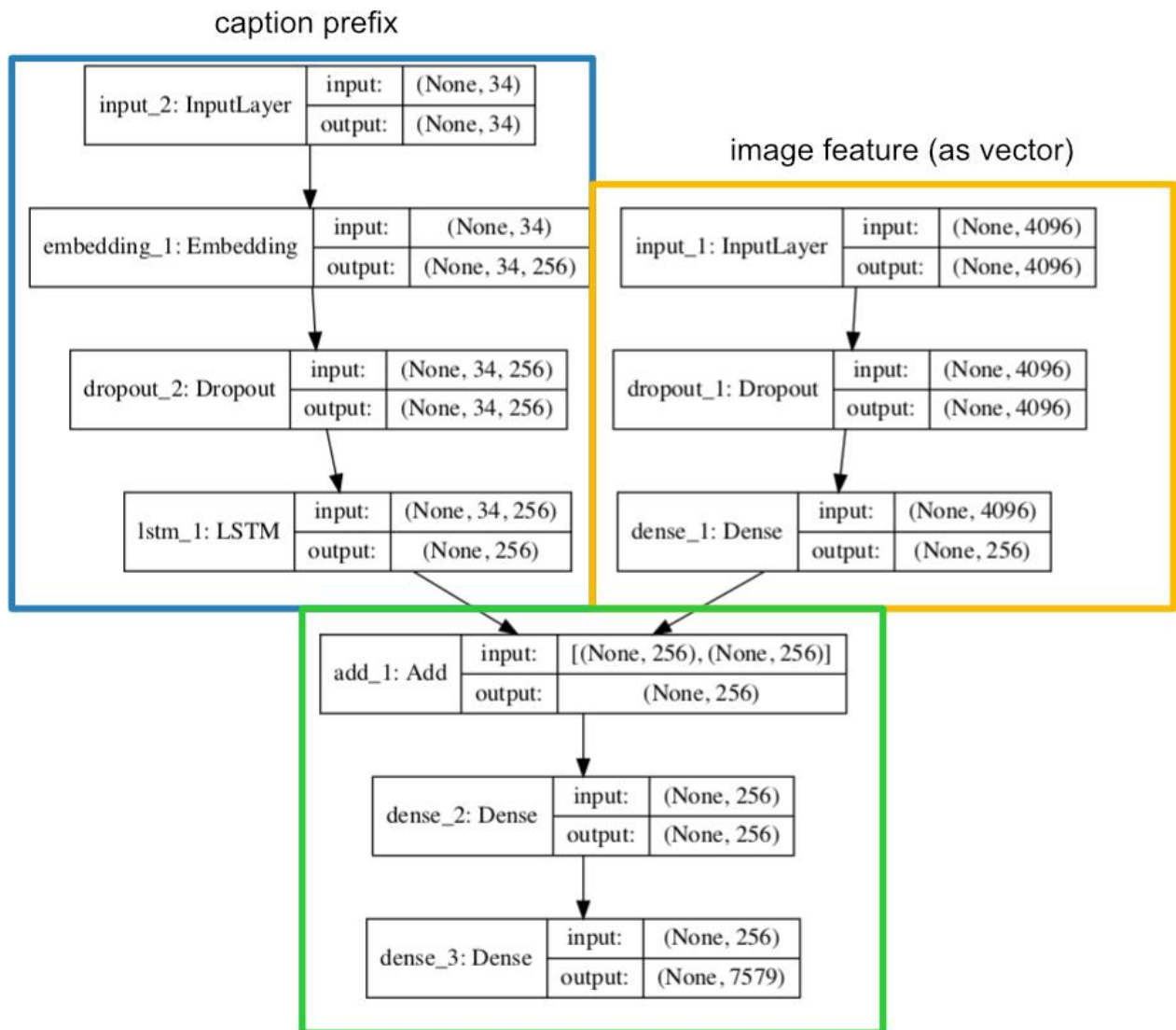
The LSTM may carry out important information throughout the processing of inputs, and it can discard non-related information using a forget gate.

As a result, we'll combine these architectures to create our picture caption generator model. It's also known as the CNN - RNN model.



LSTM
(Long-Short Term Memory)

D. System Architecture :



We shall summarise the two distinct architectures in order to construct an image caption generating model. The CNN-LSTM model is another name for it. So, to retrieve the captions for the input photographs, we'll use these two architectures.

The CNN algorithm was used to extract the key characteristics from the input image. To accomplish so, we've employed a pre-trained model called Xception, which has been used to store and analyse the data or features from the CNN model, as well as to assist in the production of a meaningful caption for the image.

E. Project File Architecture :

We have obtained the data set, which comprises of the following files, for our research purposes:

1. **Flickr 8k Datasets** — This file contains all of the images we need to train our model for. There are 8091 photos in all.
2. **Flickr 8k texts** — This folder contains text files as well as pre-written subtitles for the photos.

The following files are set up to allow us to run this system and test the CNN-LSTM model's functionality.

1. **Models** - This folder will include all the trained models which are at first trained. The model would only need to be trained once.
2. **Description.txt** – This is the file that will have the photo names and captions once they have been preprocessed.
3. **Feature.p** — This file connects the picture and their relevant captions that are taken from the 'Xception', which is a pre-trained CNN model.
4. **Tokenizers.p** — This file includes an expression we refer to as tokens, which are generalised with the index value.
5. **Models.png** — Diagrammatic illustration of the CNN-LSTM model extension.
6. **Testing captions generator.py** — This is the Python script that generates the captions for the images.
7. **Training captions generator.ipynb** — This is a Jupyter notebook, or a web-based application in brief. We utilise this to train our model and, as a result, to generate captions for our input images.

IV. Results Analysis

A. Performance parameters :

Metric : The BLEU (Bilingual Evaluation Understudy) measure is calculated by comparing candidate grammes to the reference translation's n-gram and calculating the number of matches. These matches are not based on the player's location. The better the candidate translation, the more matches there are. A perfect mismatch receives a score of 0.0, whereas a better match receives a score of 1.0. The sentence Bleu () function in NLTK is used to compare a candidate sentence to one or more reference sentences. From 0 to 1, the BLEU metric is used.

The BLEU score is what we use for an algorithm that's been used for us to assess the quality of machine-translated text. We may use BLEU to assess the quality of the caption we created. BLEU is language agnostic.

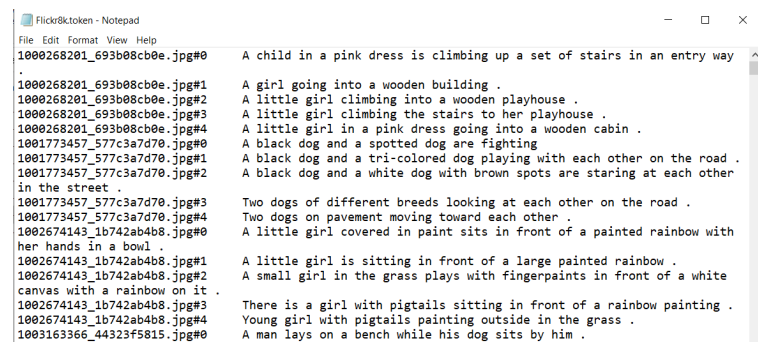
Its value lies between [0,1]. **The higher the score the better the quality of the caption.**

B. Training Steps :

STEP 1 : To begin, we import all of the required packages and libraries: **numpy, tensorflow, keras, tqdm, string, pillow, Matplotlib**, and so forth.

STEP 2 : Obtaining and cleaning data - In our Flickr 8k text folder, the primary text file that includes all image descriptions in Flickr_8k.token.

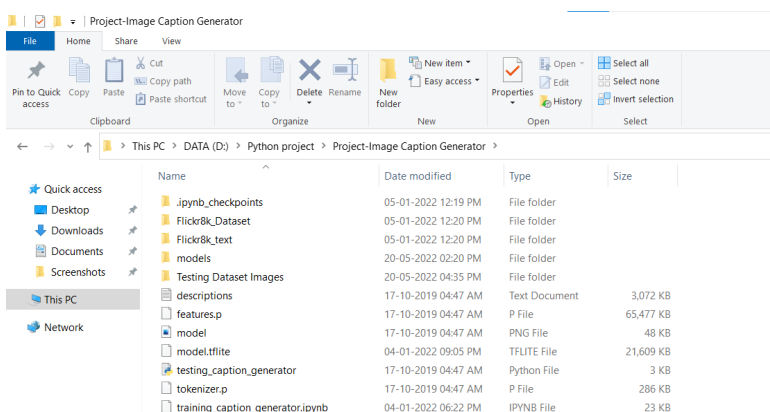
Have a look at the file –



```
File Edit Format View Help
1000268201_693b08cb0e.jpg#0 A child in a pink dress is climbing up a set of stairs in an entry way
.
1000268201_693b08cb0e.jpg#1 A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2 A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3 A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4 A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0 A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1 A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2 A black dog and a white dog with brown spots are staring at each other
in the street .
1001773457_577c3a7d70.jpg#3 Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4 Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0 A little girl covered in paint sits in front of a painted rainbow with
her hands in a bowl .
1002674143_1b742ab4b8.jpg#1 A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg#2 A small girl in the grass plays with fingerpaints in front of a white
canvas with a rainbow on it .
1002674143_1b742ab4b8.jpg#3 There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg#4 Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0 A man lays on a bench while his dog sits by him .
```

The image and description are separated by a new line ("n") in our file.

Each image contains five captions, and each caption has a #(0 to 5) number given to it.



Five Functions will be defined:

1. **load_doc (filename)** – This function loads the document file and reads the contents into a string.
2. **all_img_captions (filename)** – if all img captions(filename) – if all This programme generates a description dictionary that maps photos to a set of five captions. This is what the description dictionary will look like:

We will make minor adjustments to the Xception model for integration with our model because it was initially created for imagenet. One thing to keep in mind is that the Xception model requires an image size of 299*299*3. We'll acquire the 2048 feature vector by removing the last classification layer.

```
File Edit Format Run Options Window Help
{
'3461437556_cc5e97f3ac.jpg': ['dogs on grass',
                              'three dogs are running on the grass',
                              'three dogs one white and two brown are running together',
                              'three dogs run along grassy yard',
                              'three dogs run together in the grass'
                              ],
'3461583471_2b8b6b4d73.jpg': ['buy is grinding rail on snowboard',
                              'person is jumping ramp on snowboard',
                              'snowboarder goes down ramp',
                              'snowboarder going over ramp',
                              'snowboarder performs jump on the clean white snow'
                              ],
'997722733_0cb5439472.jpg': ['man in pink shirt climbs rock face',
                              'man is rock climbing high in the air',
                              'person in red shirt climbing up rock face covered in as',
                              'rock climber in red shirt',
                              'rock climber practices on rock climbing wall'
                              ]
}
```

3. **cleaning_text (descriptions)** – if cleaning text(descriptions) – if cleaning This function takes all of the descriptions and cleans them up. When working with textual material, this is a crucial step. We'll remove punctuation, convert all text to lowercase, and remove terms that include numbers in our instance. As a result, a caption like "A guy in a three-wheeled wheelchair" will become "man in a three-wheeled wheelchair."

4. **text_vocabulary (descriptions)** – This is a basic function that will separate all the unique words from all the descriptions and generate the vocabulary.

5. **save_descriptions (filename, descriptions)** – This function will generate a list of all the preprocessed descriptions and save them to a file. We'll save all of the captions in a descriptions.txt file.

STEP 3 : Extracting the feature vector from all images

We will take a pre-trained model that has previously been built on a huge dataset and extract the features from it to utilize in our tasks. We're utilising the Xception model, which was trained on an imagenet dataset with 1000 discrete classifications to sort through. This model may be immediately imported from the keras.applications folder. Make sure you're connected to the internet because the weights are downloaded automatically.

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

The function extract_features() will extract features for all photos, and image identifiers will be mapped to their corresponding feature arrays. The features dictionary will then be dumped into a "features.p" pickle file.

```
In [45]: #2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p", "wb"))
```

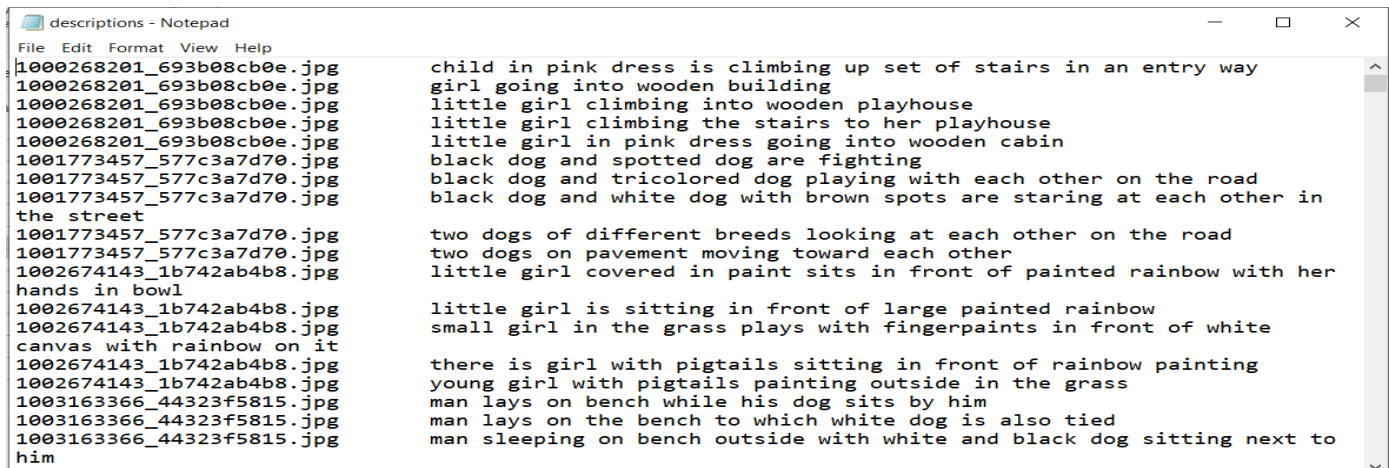
100% 8091/8091 [06:29<00:00, 20.78it/s]

Depending on your system, this procedure might take a long time. Because I'm training with an Nvidia 1050 GPU, this assignment took me around 7 minutes to complete. If you're using a CPU, though, this procedure might take up to two hours.

STEP 4 : Loading the dataset for the model's training:

More functions are required to load the training dataset:

1. **load_photos (filename)** – Returns a list of image names after loading the text file in a string.



2. load_clean_descriptions (filename, images) – This function creates a dictionary with captions for each of the photographs in the list. Each caption's start and finish identifiers are also included. This is required for our LSTM model to recognise the beginning and end of the caption.

3. load_features (photos) – This function returns the dictionary of picture names and associated feature vectors that we obtained from the Xception model before.

STEP 5 : Tokenizing the vocabulary :

The tokenizer function in the Keras library will be used to generate tokens from our vocabulary and store them to a "tokenizer.p" pickle file. There are 7577 words in our lexicon.

The maximum length of the descriptions is calculated. This is critical for determining model structural parameters.

The maximum length of a description is 32 characters.

STEP 6 : Create Data Generator :

We will use a generator approach to generate batches since the quantity of data for 8091 photos is too large to

fit into memory. The input and output sequences will be generated by the generator.

STEP 7 : Defining the CNN - RNN model :

We'll use the Keras Model from Functional API to define the model's structure. It will be divided into three sections:

1. Feature Extractor – The picture feature extracted has a size of 2048 nodes, which we will compress to 256 nodes using a dense layer.

2. The textual input - Will be handled by an embedding layer, which will be followed by the LSTM layer.

3. Decoder – We will process the dense layer to produce the final prediction by combining the output from the above two layers. The number of nodes in the final layer will be equal to the size of our vocabulary.

STEP 8 : Training the model :

We'll use the 8091 training photos to train the model by generating the input and output sequences in batches and fitting them to the model with the model.fit generator() function. The model is also saved to our models folder.

STEP 9 : Testing the model :

Now that the model has been trained, we'll create a new file called testing_caption_generator.py that will load the model and make predictions.

C. Testing Steps :

1. Open Anaconda Prompt.

2. Go to the path, where your all project files or dataset (both testing and training) present.

3. Now you need to run python file testing_caption_generator.py

4. Before running you need to give a path to the input image.

5. Type the following command in the anaconda prompt.

```
python testing_caption_generator.py -i
"Path to the input image"
```

6. After typing the command, hit enter ↵.

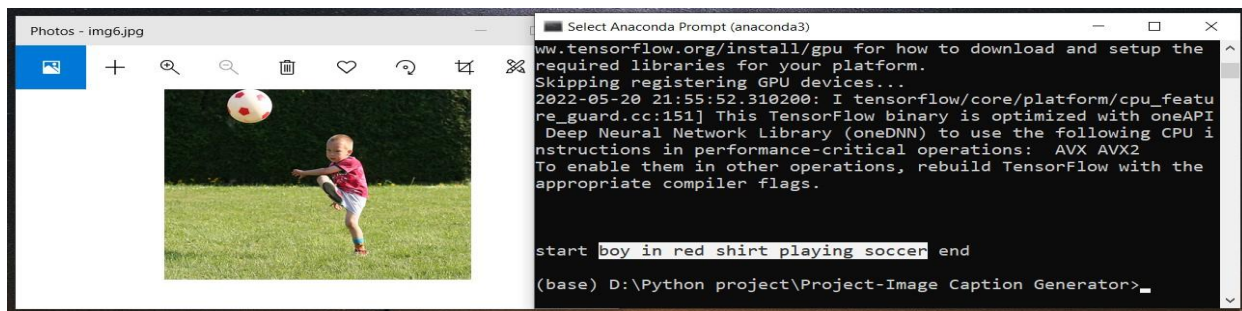
7. After a few seconds, you will see a required caption between **START** and caption **END**.

D. Testing Result :

GOOD CAPTIONS

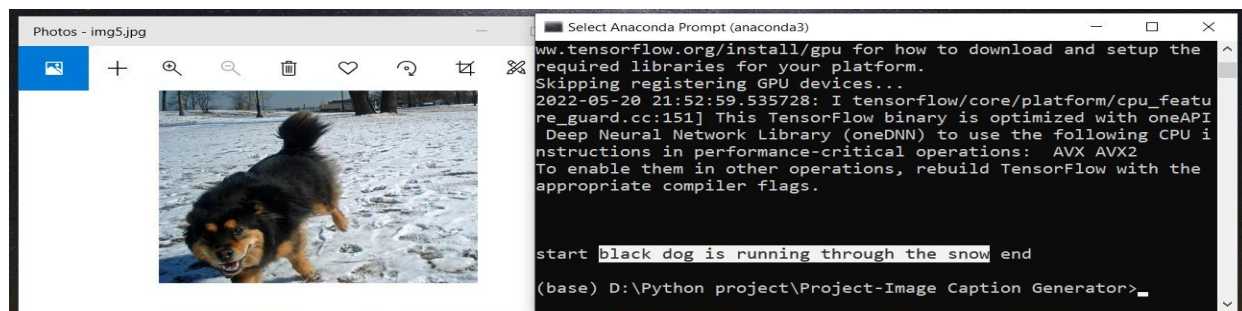
True : Boy in red shirt playing soccer.

Prediction : Boy in red shirt playing soccer.



True : Black dog is running through the snow.

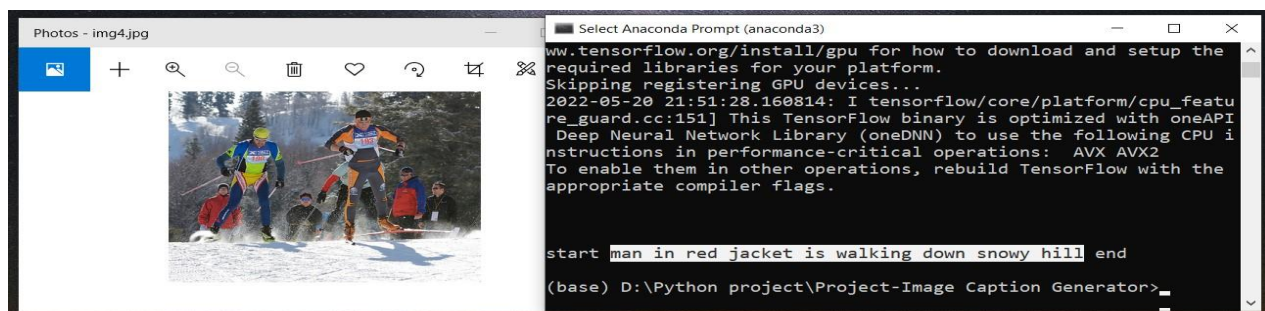
Prediction : Black dog is running through the snow.



BAD CAPTIONS

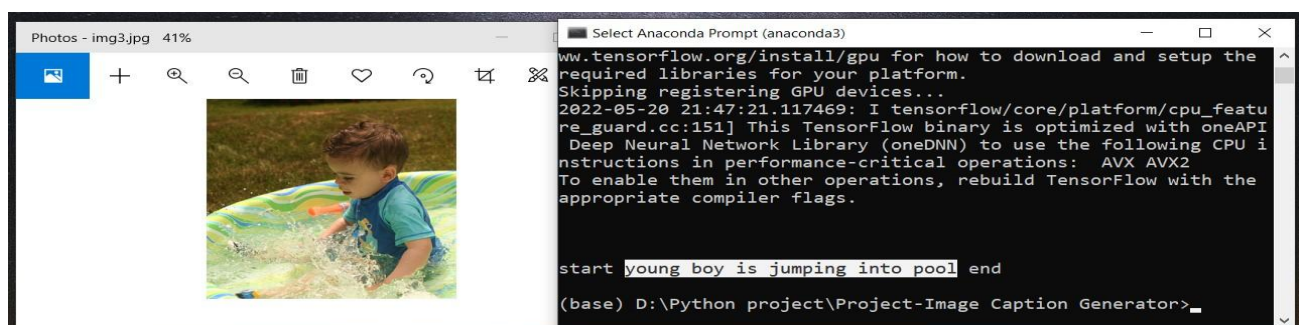
True : Man in a red jacket is **skiing** over a snow hill.

Prediction: Man in red jacket is **walking** down snowy hill.



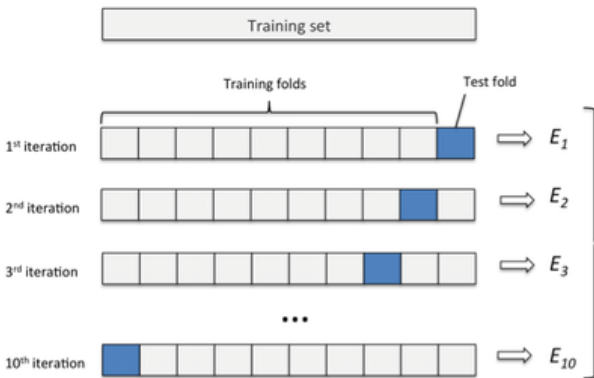
True : a young boy is **playing** into the pool.

Prediction : young boy is **jumping** into pool.



V. Observation

Fold Cross Validation: This approach uses a single data set that is randomly divided into ten sections. Nine of those components are used for training, while one tenth is set aside for testing. We repeat this technique ten times, with one tenth set aside for testing each time.



Cross Validation Observed Values

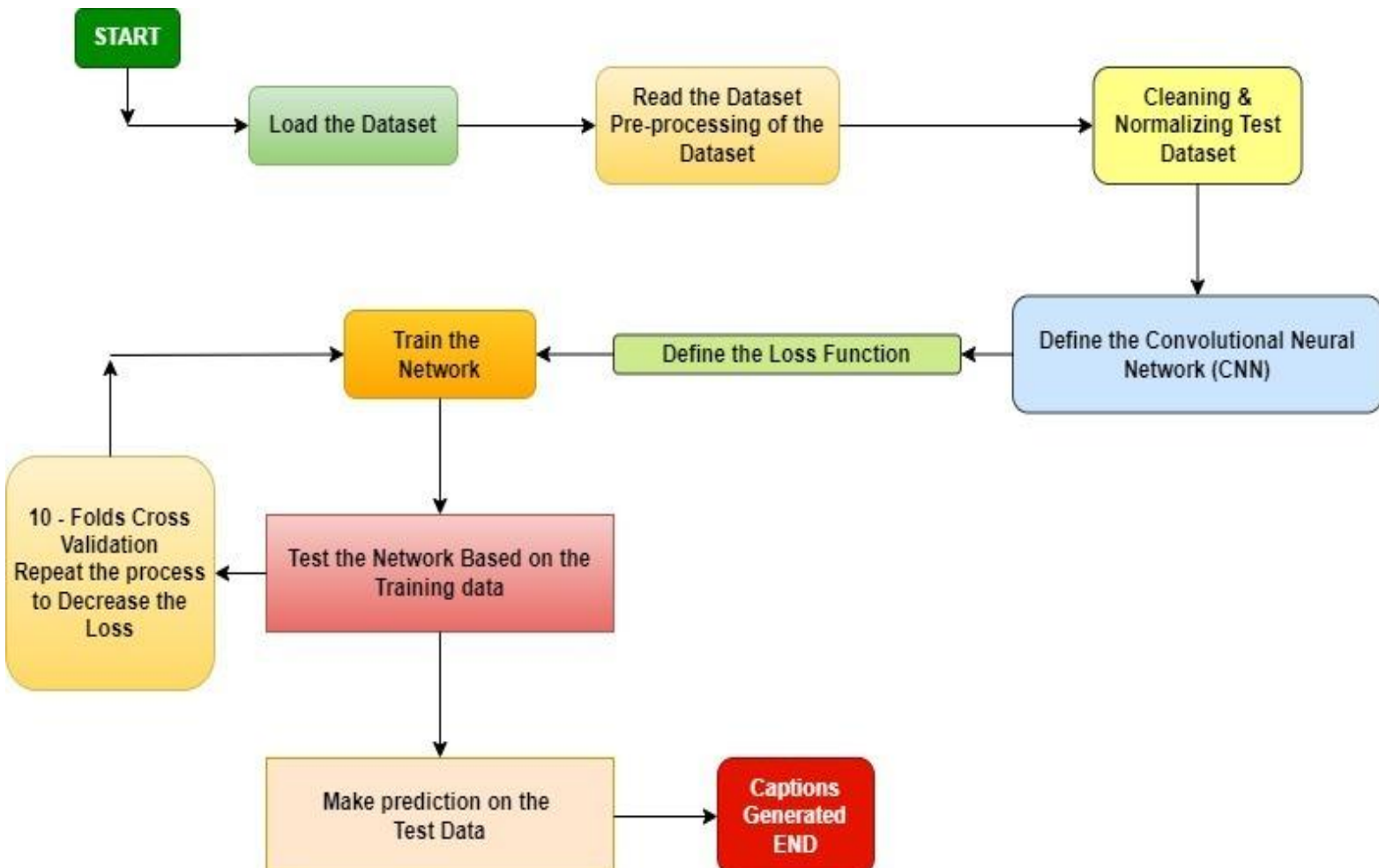
```

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

None
('You must install pydot (`pip install pydot`) and install graphviz (see instru
_to_dot to work.')
6000/6000 [=====] - 874s 145ms/step - loss: 4.4936
C:\Users\nitin\anaconda3\lib\site-packages\keras\engine\functional.py:1410: Cu
t_config. When loading, the custom mask layer must be passed to the custom_obj
layer_config = serialize_layer_fn(layer)
6000/6000 [=====] - 944s 157ms/step - loss: 3.6392
6000/6000 [=====] - 981s 164ms/step - loss: 3.3536
6000/6000 [=====] - 1017s 170ms/step - loss: 3.1812
6000/6000 [=====] - 997s 166ms/step - loss: 3.0629
6000/6000 [=====] - 933s 156ms/step - loss: 2.9749
6000/6000 [=====] - 1104s 184ms/step - loss: 2.9079
6000/6000 [=====] - 1234s 206ms/step - loss: 2.8526
6000/6000 [=====] - 1188s 198ms/step - loss: 2.8055
6000/6000 [=====] - 1267s 211ms/step - loss: 2.7702

```

FLOWCHART



VI. Conclusion

For the captioning of photos, we used a deep learning technique in this paper. The CNN-LSTM model was created with the goal of automatically producing captions for the input images. This concept may be used in a wide range of situations. We looked at the CNN model, RNN models, and LSTM models, and we verified that the model is creating captions for the input images. In order for the model to be as accurate as possible, BeLU scores must be high.

We can improve our result by a lot of modifications for example :

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

VII. Future Work

- Image captions are used in self-driving automobiles to explain the environment surrounding the vehicle.
- Second, it might be a useful tool for blind people, as it can help them in every manner possible by translating scenes to captions and subsequently to audio.
- Thirdly, while describing the scenario, CCTV cameras might be used to trigger warnings if any harmful behaviour is detected, modifying suggestions, social media postings.

Integration with android application and camera. What I do using this model mentioned the steps below :

Steps :

- First of all, train an ML Model on a large dataset (so that it can predict better) and Create a 'tflite' model using TensorflowLite.
- After Creating tflite, Build or integrate this 'model.tflite' in android studio.
- **Front end implementation** – `ImageView`, `TextView`, `select(From Internal)`, `capture(Camera)` and `predict` button in Android Studio.
- **Back end Implementation** – Show output `caption` in `textview` as well as convert this

'text to speech' for blind people or blurred vision people.

- Finally Improve Front end of App for Better Visuals (Colorful and splash screen & Headlines - Caption Bot for Assisted Vision)
- App is ready to 'predict'.

VIII. References

[1] Abhaya Agarwal and Alon Lavie. (2008) Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In Proceedings of the Third Workshop on Statistical Machine Translation. Association for Computational Linguistics

[2] Satanjeev Banerjee and Alon Lavie (2005) Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization

[3] Yuntao Wang^{a,b}, Albert S. Chen^a, Guangtao Fu^a, Slobodan Djordjević^b, Chi Zhang^a, Dragan A. Savić^b, (2018), "An integrated framework for high-resolution urban flood modeling considering multiple information sources and urban features".

[4] Manish Raypurkar, Abhishek Supe, Pratik Bhumkar, Pravin Borse, Dr. Shabnam Sayyad (2021): Deep learning-based Image Caption Generator

[5] Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra and Nand Kumar Bansode (2017): Camera Caption: A Real-Time Image Caption Generator

[6] Swarajya Lakshmi V Papineni, Snigdha Yarlagadda, Haritha Akkineni, A. Mallikarjuna Reddy. Big Data Analytics Applying the Fusion Approach of Multicriteria Decision Making with Deep Learning Algorithms International Journal of Engineering Trends and Technology

[7] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. (2016). "Spice: Semantic propositional image caption evaluation". In the European Conference on Computer Vision. Springer, 382–398.

[8] Parth Shah, Vishvajit Bakrola, Supriya Pati (2017). Image Captioning using Deep Neural Architectures in IEEE International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS).

[9] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan (2015): Show and Tell: A Neural Image Caption Generator

[10] Jianhui Chen, Wenqiang Dong, Minchen Li (2015): Image Caption Generator based on Deep Neural Networks

[11] Ali Ashraf Mohamed (2020): Image Caption using CNN & LSTM by Ali ashraf Mohamed

[12] Preksha Khant, Vishal Deshmukh, Aishwarya Kude, Prachi Kiraula (2021): Image Caption Generator using CNN-LSTM

[13] Ujwala Bhoga, V.Aravind, G.Sreeja, MohdArif (2021): Image Caption Generation Using CNN and LSTM

[14] Priyanka Raut and Rushali A Deshmukh (2021): An Advanced Image Captioning using combination of CNN and LSTM

[15] Kun Fu, Jin Li, Junqi Jin, and Changshui Zhang (2021): Image-Text Surgery: Efficient Concept Learning in Image Captioning by Generating Pseudo Pairs