International Journal of Research in Science and Technology http://www.ijrst.com

(IJRST) 2020, Vol. No. 10, Issue No. IV, Oct-Dec **e-ISSN: 2249-0604, p-ISSN: 2454-180X**

Employability of Neural Network Tools and Techniques for Enhancing Image Caption Generation[1]

Harshit Dua

Galgotias University, Uttar Pradesh, India

DOI: 10.37648/ijrst.v10i04.003

Received: 04th September, 2020; Accepted: 01st October, 2020; Published: 14th October, 2020

ABSTRACT

Nowadays, there is massive research in generating automatic image caption; this technique is very challenging and uses Natural language processing. For instance, it could assist incapacitated people with improving the matter of images on the web. Likewise, it could give more precise and minimized images/recordings in situations, such as picture sharing in interpersonal organization or video surveillance system. The structure comprises a convolutional neural organization (CNN) traced by a repetitive neural organization (RNN). The strategy can produce picture sayings that are generally semantically unmistakable and linguistically right by taking in information from picture and subtitle matches. Individuals, for the most part, depict a scene utilizing characteristic languages which are concise and reduced. However, computer vision frameworks define the set by taking a picture which is a two-measurement presentation. The plan is to picture and engrave similar places and projects from the image to the sentences.

***Keywords:** Computer Vision, Image Captions, Neural Nets, Mappings*

1. INTRODUCTION

Consequently, depicting the substance of pictures utilizing characteristic dialects is an essential and testing task. It has an incredible expected effect. For instance, it could assist outwardly hindered individuals with bettering the substance of pictures on the web. Likewise, it could give more precise and minimized images/recordings in situations, such as picture partaking in interpersonal organization or video reconnaissance frameworks. This undertaking achieves this errand utilizing profound neural organizations. By taking in information from picture and inscription matches, the strategy can produce picture subtitles that are generally semantically expressive and syntactically right. People commonly portray a scene utilizing standard dialects which are succinct and minimized. Notwithstanding, machine vision frameworks depict the scene by taking a picture that is a two-measurement cluster. From this viewpoint, Vinyl et al. (Vinyals et al.) model the picture inscribing issue as a language interpretation issue in their Neural Image Caption (NIC) generator framework. The thought is

Planning the picture and inscriptions to a similar space and planning from the image to the sentences.

Donahue et al. (Donahue et al.) proposed a broader Long-term Recurrent Convolutional Network (LRCN) strategy. The LRCN strategy not just models the one-to-many (words) picture inscribing yet in addition models many-to-one activity age and many-to-numerous video portrayal. They also give openly

[1] *How to cite the article:* Dua H., Employability of Neural Network Tools and Techniques for Enhancing Image Caption Generation, IJRST, Oct-Dec 2020, Vol 10, Issue 4, 20-29, DOI: http://doi.org/10.37648/ijrst.v10i04.003

accessible execution dependent on the Caffe structure (Jia et al., 2014), which further lifts the exploration of picture subtitling. This work depends on the LRCN technique. Albeit every one of the mappings is learned in a start to finish structure, we have confidence in the advantages of a superior comprehension of the framework by breaking down various segments independently. Fig. 1 shows the pipeline. The model has three parts. The central part is a CNN which is utilized to comprehend the substance of the picture. Picture understanding answers the regular inquiries in PC vision, for example, "What are the items?", "Where are the articles?" and "How are the articles intelligent?". For instance, CNN needs to perceive the "dog", "chair", and their broad areas in the picture. The next part is an RNN which use to create a sentence given the visual component. For instance, the RNN needs to produce a sequence of possibilities of words given two words "teddy bear, table". The third segment is utilized to create a sentence by investigating the blend of the options. This part is less concentrated in the reference paper (Donahue et al.). This venture targets understanding the effect of

Various parts of the LRCN technique (Donahue et al.). We have the accompanying commitments:

_ comprehend the LRCN technique at the execution level.

_ break down the impact of the CNN part by supplanting three CNN models (two from the creator's and one from our execution).

_ break down the impact of the RNN part by supplanting two RNN models. (one from the creator's and one from our execution).

_ break down the impact of the sentence age strategy by contrasting two techniques (one from the creator's and one from our execution).

2. RELATED WORK

We present a combined output generator that defines and depicts items, traits, and connections in a picture in a characteristic language structure. Thus, to make our picture inscription generator model, we will be blending these structures. It is likewise called a CNN-RNN model.

- CNN uses feature extraction from the images.
- LSTM will utilize the data from CNN to help create data of the picture.

CNN-Convolutional Neural organizations are particular intense neural organizations that can interact with input shapes like a 2D framework. Pictures are effortlessly addressed as a 2D framework, and CNN is valuable in working with images. CNN is fundamentally utilized for picture orders and recognizing if a picture is a bird, a plane or Superman, and so forth. It checks images from left to right and through and through to pull out significant highlights from the picture and join them to characterize images. It can deal with the photos that have been interpreted, turned, scaled and changes in context.

- LSTM

LSTM represents Long short memory; they are a kind of RNN (intermittent neural network) appropriate for grouping forecast issues. Given the preceding text, we can anticipate what the following word will be. It has substantiated itself power from the usual RNN by defeating the restrictions of RNN, which had a short memory. LSTM can do significant data by preparing data sources, and with a dropped access, it disposes of non-pertinent data.

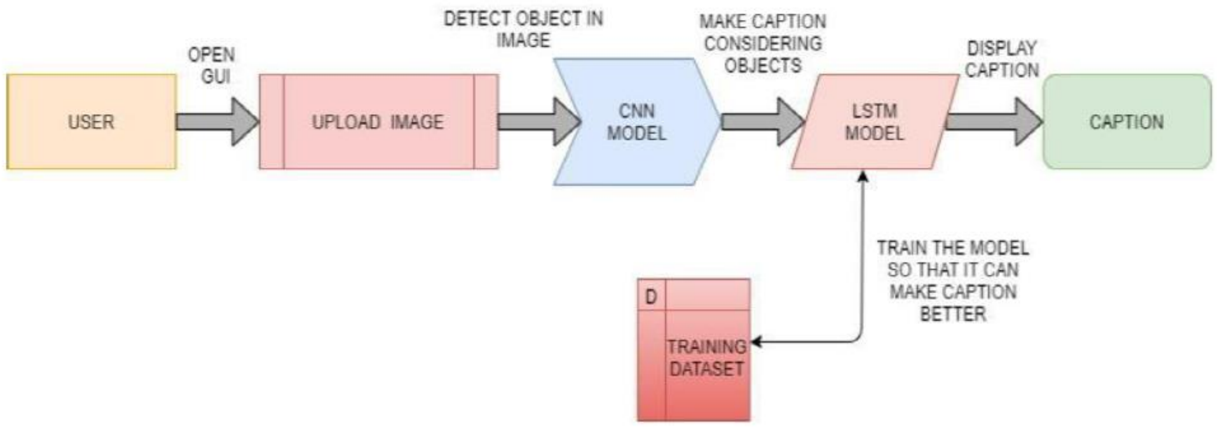


Fig 1: Flow of LSTM

3. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (ConvNets or CNNs) are a sort of Artificial Neural Network that has demonstrated excellent picture acknowledgement and grouping. Article identification, self-driving vehicles, picture inscribing, and different errands have all utilized them. Yann Lecun found the first convnet in 1990, and the model's design was named the LeNet engineering. The outline beneath portrays a fundamental convnet.

Convolution, Non-Linearity (ReLU), Pooling or Sub Sampling and Classification are the four principal tasks that can utilize to clarify the whole engineering of a convnet (Fully Connected Layer). Since these activities are the essential structure squares of each Convolutional Neural Network, seeing how they work is critical to acquiring an exhaustive comprehension of ConvNets. Beneath, we'll turn out every one of these activities in detail. Each picture can be addressed as a grid of pixel esteems. An image from a standard computerized camera will have three channels: red, green, and blue. Envision three 2d-lattices stacked on top of one another (one for each tone), each with pixel range from 0 to 255.

4. MODEL BUILDING RELATED WORK

Subtitle age is a kind of human-made brainpower issue in which a literary portrayal for a photo is required. Attempt to give an inscription for the picture beneath.

Subtitle age is a sort of human-made consciousness issue in which a literary portrayal for a photo is required. Take a stab at subtitling the picture underneath.

"A canine on the grass for certain pink blossoms," a human may compose as an inscription. Different depictions may exist, yet they all mean the same thing. It's so essential for us, as people, to take a gander at an image and depict it in a language that bodes well. Be that as it may, how does a PC program think of a suitable subtitle for a picture?

Now, profound learning enters the image, with two models: CNN and RNN, being utilized to inscribe the pictures. The articles in the image are distinguished utilizing CNN, and the subtitles are created utilizing RNN. Both of these models' functioning models recently appeared. To prepare our model, we used the MS-COCO dataset. The dataset contains around 82,000 pictures, each within any event five distinctive inscription explanations.

Move learning is an AI method wherein a model made for one undertaking is utilized as the reason for a model on an alternate errand. To pre-process and order the pictures, we use InceptionV3 (which is pre-prepared on Imagenet). The yield is reserved to plate in the wake of pre-processing, and the inscriptions are tokenized, with the jargon size restricted to 5000 words to save memory and the leftover words supplanted with UNK tokens. The highlights are extricated from InceptionV3's lower convolutional layer, giving us a shape vector (8, 8, 2048).

The CNN Encoder (which comprises a solitary completely associated layer) is then used to deal with this vector. The RNN (for this situation, GRU) at that point takes a gander at the picture and predicts the following word. The model is then prepared by extricating the highlights put away in the. Any documents and afterwards going those highlights through the encoder. The decoder gets the encoder yield, covered upstate (introduced to 0), and decoder input (which is the beginning token). The decoder returns the forecasts just as the secret condition of the decoder.

The secret condition of the decoder is then passed once more into the model, and the misfortune is determined utilizing the forecasts. To decide the following contribution to the decoder, use instructor compelling.

The objective word is passed as the following contribution to the decoder in a procedure known as instructor compelling. The model should then be back-spread after the angles have been determined and applied to the enhancer.

5. ASSESMENT

Execution of the whole program happens in 5 significant advances. The performance of the five essential modules is as per the following:

A. Information Cleaning and Pre-processing:

1. For an agreeable and quick work insight, we use Google Collaboratory: an apparatus that gives free GPU/TPU preparing power over our nearby machines, which can require a few hours to do the undertaking that a GPU will require few moments to do.

2. Our program begins with stacking both the content record and the picture document into independent factors; the test record is put away in a string.

3. This string is utilized and controlled in such a manner to make a word reference that maps each picture with a rundown of 5 portrayals.

4. The fundamental assignment of information cleaning includes eliminating accentuation marks, changing the entire content over to lowercase, eliminating stop words and eliminating words that contain numbers.

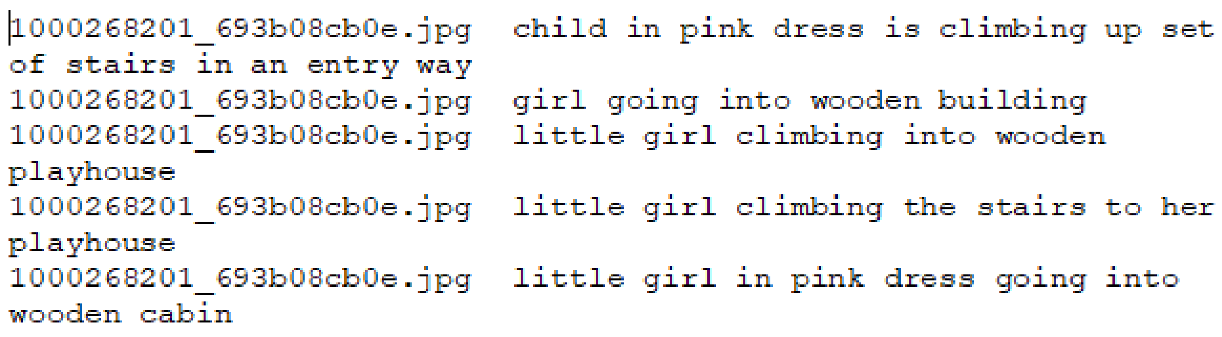
5. Further, a jargon of all novel words from every one of the portrayals is made, which will be utilized to create subtitles to test pictures.

6. Another part of Pre-processing the information includes tokenizing our jargon with novel file esteem. This is because a PC will not comprehend everyday English words. Subsequently, they should be addressed utilizing numbers. The tokens are then put away in a predicament record, such as a character stream; however, it is essential to remaking it into the first item type.

7. The over two pre-processing errands can be accomplished physically or by utilizing the Keras. Pre-processing module for simplicity of composing the code.

8. We continue to annex the <start> and <end> identifier for each inscription since these will go about as markers for our LSTM to comprehend where a subtitle is beginning and where its completion.

9. We will continue with computing the number of words in our jargon and tracking down the most extreme length of the portrayal, which will be utilized later.



```
1000268201_693b08cb0e.jpg  child in pink dress is climbing up set
of stairs in an entry way
1000268201_693b08cb0e.jpg  girl going into wooden building
1000268201_693b08cb0e.jpg  little girl climbing into wooden
playhouse
1000268201_693b08cb0e.jpg  little girl climbing the stairs to her
playhouse
1000268201_693b08cb0e.jpg  little girl in pink dress going into
wooden cabin
```

Figure 2: Generation of Text file after data cleaning

B. Feature vectors Extraction :

1. An element vector(or highlight) is a mathematical worth in the framework structure, containing data about an article's important qualities, e.g. power worth of every pixel of the picture for our situation. These vectors, we'll eventually store in a sticky situation record.

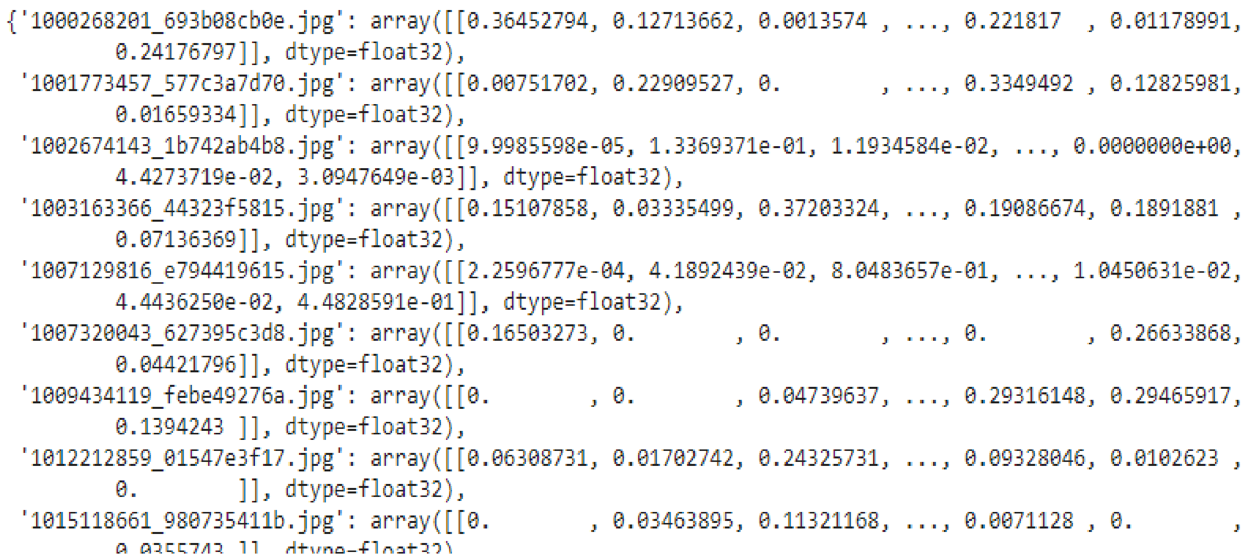
2. In our model, we'll utilize Transfer Learning, which just methods, using a pre-prepared model(for our situation, the Xception model) to extricate highlights from it.

3. The Xception model is a Convolutional Neural Network that is 71 layers profound. It is prepared on the well-known Imagenet dataset, which has many pictures and more than 1000 unique classes to group from.

4. Python utilises this model in our code very simple with Keras.applications.exception module. To use it in our code, we'll drop the grouping layer from it and consequently acquire the 2048 component vector.

5. Henceforth, you will download loads for each picture and plan picture names with their particular element cluster afterwards.

6. This interaction can require a couple of hours, relying upon your processor.



```
{'1000268201_693b08cb0e.jpg': array([[0.36452794, 0.12713662, 0.0013574 , ..., 0.221817  , 0.01178991,
        0.24176797]], dtype=float32),
 '1001773457_577c3a7d70.jpg': array([[0.00751702, 0.22909527, 0.        , ..., 0.3349492 , 0.12825981,
        0.01659334]], dtype=float32),
 '1002674143_1b742ab4b8.jpg': array([[9.9985598e-05, 1.3369371e-01, 1.1934584e-02, ..., 0.0000000e+00,
        4.4273719e-02, 3.0947649e-03]], dtype=float32),
 '1003163366_44323f5815.jpg': array([[0.15107858, 0.03335499, 0.37203324, ..., 0.19086674, 0.1891881 ,
        0.07136369]], dtype=float32),
 '1007129816_e794419615.jpg': array([[2.2596777e-04, 4.1892439e-02, 8.0483657e-01, ..., 1.0450631e-02,
        4.4436250e-02, 4.4828591e-01]], dtype=float32),
 '1007320043_627395c3d8.jpg': array([[0.16503273, 0.        , 0.        , ..., 0.        , 0.26633868,
        0.04421796]], dtype=float32),
 '1009434119_febe49276a.jpg': array([[0.        , 0.        , 0.04739637, ..., 0.29316148, 0.29465917,
        0.1394243 ]], dtype=float32),
 '1012212859_01547e3f17.jpg': array([[0.06308731, 0.01702742, 0.24325731, ..., 0.09328046, 0.0102623 ,
        0.        ]], dtype=float32),
 '1015118661_980735411b.jpg': array([[0.        , 0.03463895, 0.11321168, ..., 0.0071128 , 0.        ,
        0.0355743 ]], dtype=float32)
```

Figure 3: Glimpse of extracted features with corresponding image names, that we'll store in a pickle file

C. Layering the CNN-RNN model

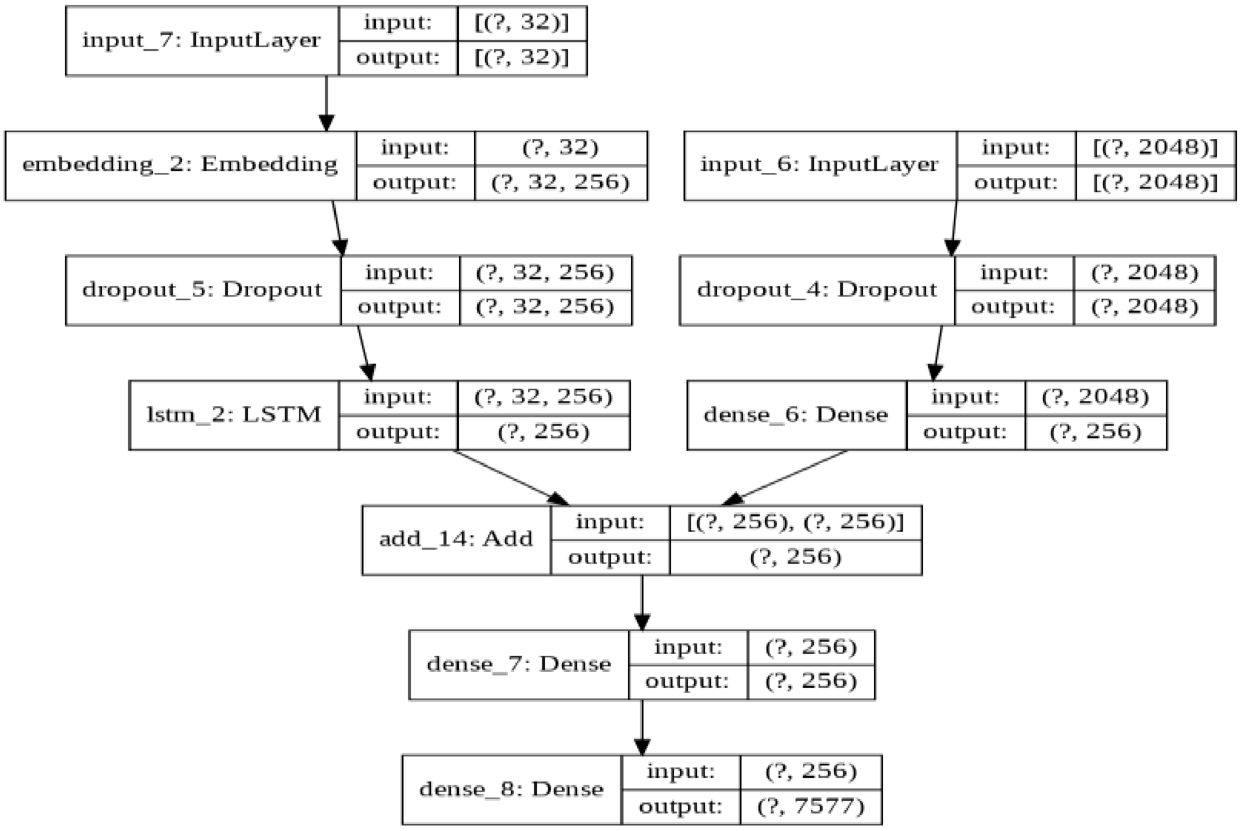

input_7: InputLayer | input: [(?, 32)] | output: [(?, 32)]
embedding_2: Embedding | input: (?, 32) | output: (?, 32, 256)
input_6: InputLayer | input: [(?, 2048)] | output: [(?, 2048)]
dropout_5: Dropout | input: (?, 32, 256) | output: (?, 32, 256)
dropout_4: Dropout | input: (?, 2048) | output: (?, 2048)
lstm_2: LSTM | input: (?, 32, 256) | output: (?, 256)
dense_6: Dense | input: (?, 2048) | output: (?, 256)
add_14: Add | input: [(?, 256), (?, 256)] | output: (?, 256)
dense_7: Dense | input: (?, 256) | output: (?, 256)
dense_8: Dense | input: (?, 256) | output: (?, 7577)

Figure 4: Neural Network Structure

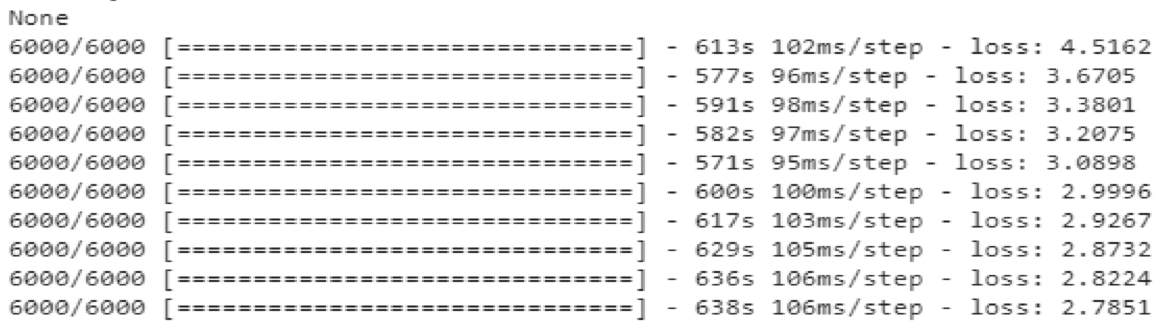
To stack the model, we'll utilize the Keras Model from Functional API. The design will comprise three sections:

1. Highlight Extractor: It will be utilized to decrease the measurements from 2048 to 256. We'll use a Dropout Layer. One of these will be included the CNN and the LSTM each. We have pre-prepared the photographs with the Xception model (without the yield layer) and will utilize the extricated highlights anticipated by this model as info.

2. Succession Processor: This Embedding layer will deal with the literary info trailed by the LSTM layer.

3. Decoder: We will consolidate the yield from the over two layers and utilize a Dense layer to make the last forecasts. Both the component extractor and grouping processor yield a fixed-length vector. These are consolidated and handled by a Dense layer. The number of hubs in the last layer will be equivalent to the size of our jargon. [6,7].

D. Training the model:



```
None
6000/6000 [==============================] - 613s 102ms/step - loss: 4.5162
6000/6000 [==============================] - 577s 96ms/step - loss: 3.6705
6000/6000 [==============================] - 591s 98ms/step - loss: 3.3801
6000/6000 [==============================] - 582s 97ms/step - loss: 3.2075
6000/6000 [==============================] - 571s 95ms/step - loss: 3.0898
6000/6000 [==============================] - 600s 100ms/step - loss: 2.9996
6000/6000 [==============================] - 617s 103ms/step - loss: 2.9267
6000/6000 [==============================] - 629s 105ms/step - loss: 2.8732
6000/6000 [==============================] - 636s 106ms/step - loss: 2.8224
6000/6000 [==============================] - 638s 106ms/step - loss: 2.7851
```

Figure 5: Training Model

1. We'll prepare our model on 6000 pictures, each having 2048 long component vectors.

2. Since it is beyond the realm of imagination to expect to hold this information in the memory simultaneously, we'll utilize a Data Generator. This will assist us with making groups of the information and will improve the speed.

3. Alongside this, we'll characterize the number of epochs(i.e. emphasis of the preparation dataset) the model needs to finish during its preparation. This number must be chosen so that our model is neither under fitted nor overfitted.

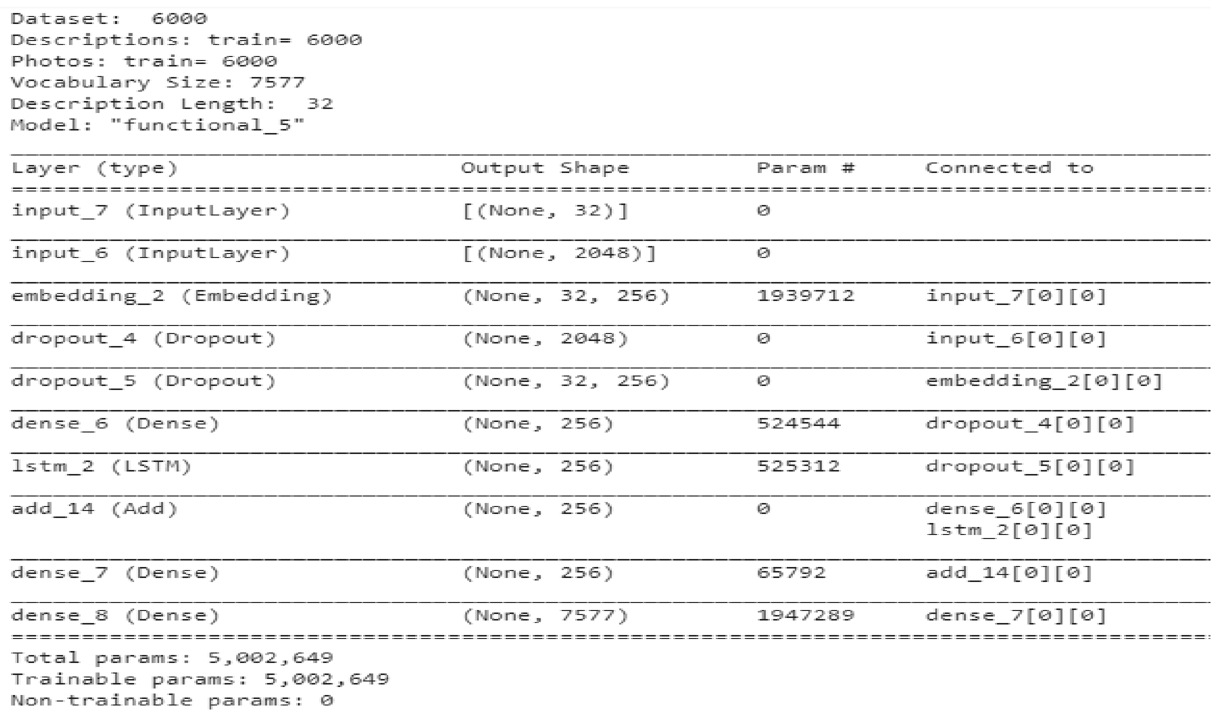
4. will utilize model.fit_generator() strategy. Also, this entire interaction will take some time contingent upon the processor.

5. The most extreme length of depictions determined before will be utilized as boundary esteem here. It will likewise take as information the clean and tokenized information.

6. We'll likewise make an arrangement maker, which will assume the part of anticipating the following word dependent on the past word and highlight vectors of the picture.

7. While preparing our model, we can utilize the advancement dataset(It's given the remainder of the documents) to screen the presence of the model, to choose when to save the model adaptation to the record

8. We will continue to save a few models, which will utilize the last one for testing in the future.



```
Dataset:  6000
Descriptions: train= 6000
Photos: train= 6000
Vocabulary Size: 7577
Description Length:  32
Model: "functional_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_7 (InputLayer)            [(None, 32)]         0
__________________________________________________________________________________________________
input_6 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 32, 256)      1939712     input_7[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 2048)         0           input_6[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 32, 256)      0           embedding_2[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 256)          524544      dropout_4[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 256)          525312      dropout_5[0][0]
__________________________________________________________________________________________________
add_14 (Add)                    (None, 256)          0           dense_6[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 256)          65792       add_14[0][0]
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 7577)         1947289     dense_7[0][0]
==================================================================================================
Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0
```

Figure 6 : Model details

E. Testing the model:

1. A different Python note pad can be made or can utilize the equivalent to perform testing. In any case, we'll load the prepared model that we had saved in the past advance and create expectations.

2. The succession generator will become an integral factor at this stage, other than the tokenizer document we made.

3. will perform the critical advance of highlight extraction for the specific picture under perception.

4. The way one of the pictures from the excess 2000 test pictures is passed to the capacity physically. You can likewise emphasize through the informational test index and store the expectation for each image in a word reference or a rundown.

5. The real working behind picture age includes utilizing the beginning arrangement and the end succession and calling the model recursively to create significant sentences.

6. TEST RESULTS

The underneath picture shows how Mask R-CNN deals with a view, and articles are independently distinguished in the type of the cover of pixels.

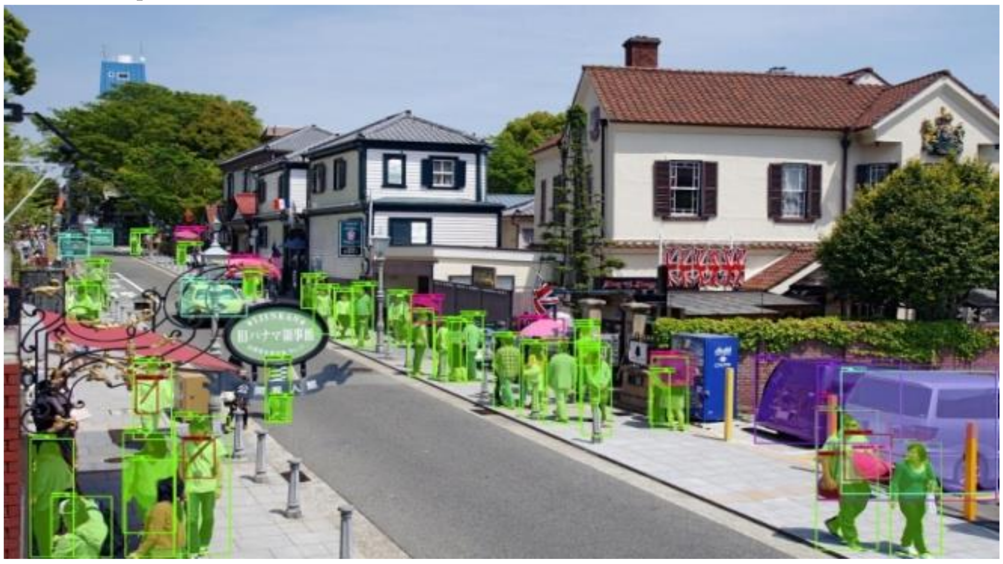
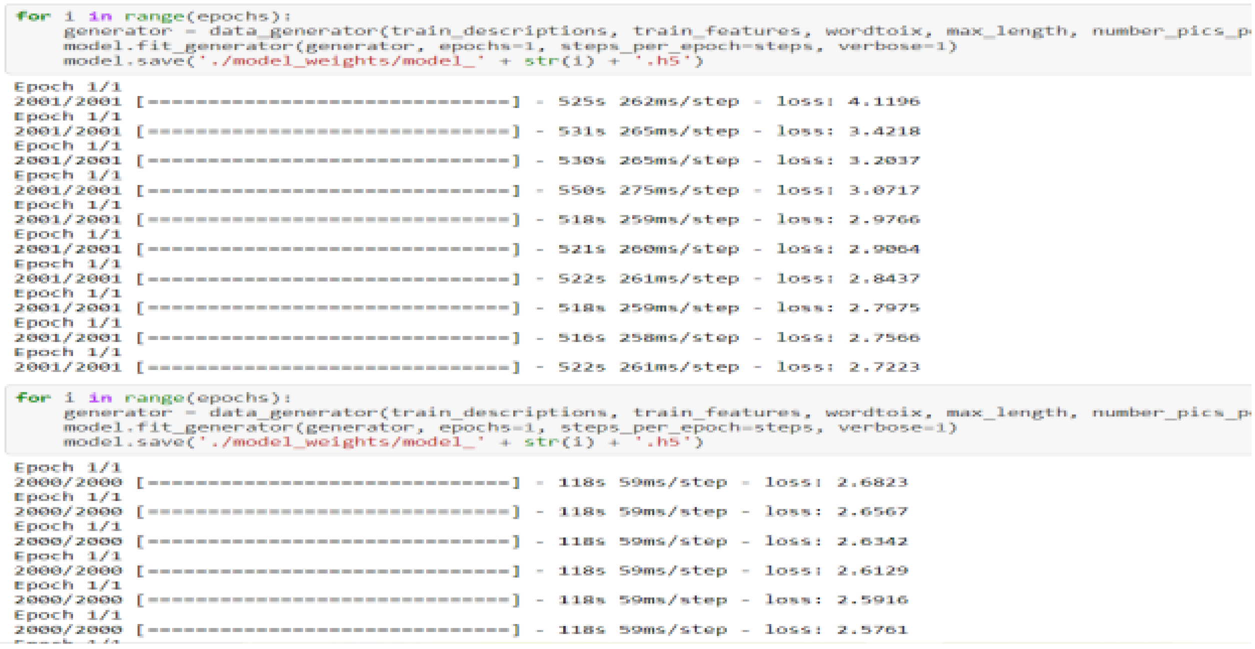


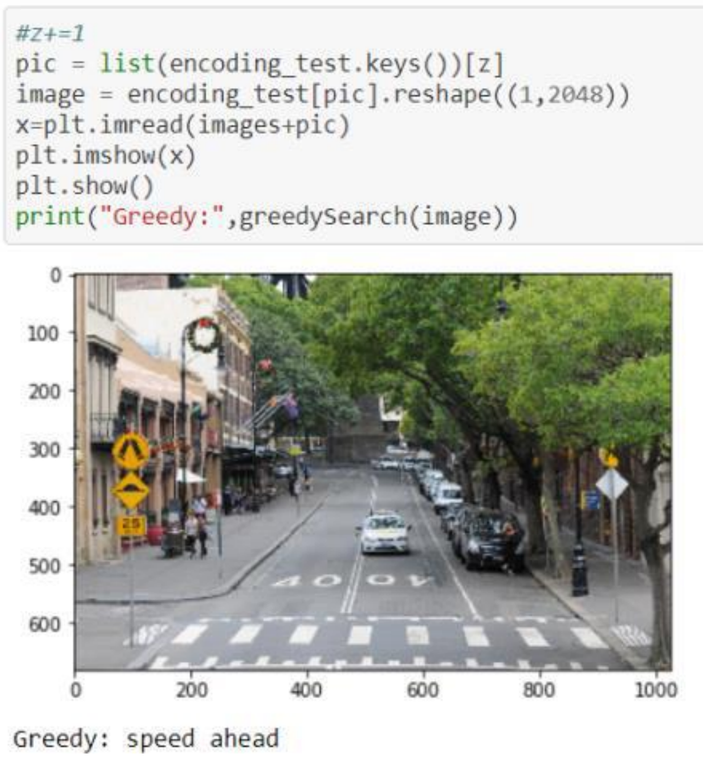
Fig. 7- Example Mask R-CNN

Output snap of mode generator with epochs shown below



```
for i in range(epochs):
    generator = data_generator(train_descriptions, train_features, wordtoix, max_length, number_pics_p
    model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
    model.save('./model_weights/model_' + str(i) + '.h5')
Epoch 1/1
2001/2001 [==============================] - 525s 262ms/step - loss: 4.1196
Epoch 1/1
2001/2001 [==============================] - 531s 265ms/step - loss: 3.4218
Epoch 1/1
2001/2001 [==============================] - 530s 265ms/step - loss: 3.2037
Epoch 1/1
2001/2001 [==============================] - 550s 275ms/step - loss: 3.0717
Epoch 1/1
2001/2001 [==============================] - 518s 259ms/step - loss: 2.9766
Epoch 1/1
2001/2001 [==============================] - 521s 260ms/step - loss: 2.9064
Epoch 1/1
2001/2001 [==============================] - 522s 261ms/step - loss: 2.8437
Epoch 1/1
2001/2001 [==============================] - 518s 259ms/step - loss: 2.7975
Epoch 1/1
2001/2001 [==============================] - 516s 258ms/step - loss: 2.7566
Epoch 1/1
2001/2001 [==============================] - 522s 261ms/step - loss: 2.7223
for i in range(epochs):
    generator = data_generator(train_descriptions, train_features, wordtoix, max_length, number_pics_p
    model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
    model.save('./model_weights/model_' + str(i) + '.h5')
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.6823
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.6567
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.6342
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.6129
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.5916
Epoch 1/1
2000/2000 [==============================] - 118s 59ms/step - loss: 2.5761
```

Fig. 8. The output of greedysearch function in image captioning is shown below



```
#z+=1
pic = list(encoding_test.keys())[z]
image = encoding_test[pic].reshape((1,2048))
x=plt.imread(images+pic)
plt.imshow(x)
plt.show()
print("Greedy:",greedySearch(image))
```

```
Greedy: speed ahead
```

7. CONCLUSION

We made a model to the inscription the pictures in our venture. Additionally extended our model by changing our subtitles over to discourse. We desire to keep dealing with voice union later on. We directed examination to grasp our models and afterwards executed every one independently entirely. We figured out how profound learning methods work and how to assemble models utilizing them. The dataset contained 82000 pictures with five inscriptions for each image. For better precision, our model can be prepared for the more excellent dataset. While running the model and working with our datasets, we experienced various challenges. Notwithstanding, as time went on, we figured out how to address our blunders and make a more productive model.

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, 115–118.

[2] Ahmet Aker and Robert Gaizauskas. 2010. Generating image descriptions using dependency relational patterns. In Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics, 1250–1258.

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

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and vqa. ArXiv preprint arXiv:1707.07998 (2017). [5] Jyoti Aneja, Aditya Deshpande, and Alexander G Schwing. 2018. Convolutional image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5561–5570.

[5] Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, Trevor Darrell, Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, et al. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR).

[7] Shuang Bai and Shan An. 2018. A Survey on Automatic Image Caption Generation. Neurocomputing. ACM Computing Surveys, Vol. 0, No. 0, Article 0. Acceptance Date: October 2018. 0:30 Hossain et al.

[8] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, Vol. 29. 65–72.