Image Captioning

Anurag C M, Nischitha S J, P Rashmi, Preetham T N

(Final Year B.E, Department of Computer Science and Engineering, JSS Science and Technology University, Mysuru)

ABSTRACT

Abstract- The issue of image captioning, which involves automatically generating one or more phrases to comprehend an image's visual information, has benefited from advancements in image categorization and object detection. Humans can easily comprehend image content and express it in the form of natural language sentences tailored to specific needs for the image captioning task; however, for computers, this requires the integrated use of image processing, computer vision, natural language processing, and other significant areas of research findings.

Keyword: - *CNN, LSTM, Attention, Deep learning, Image pre-processing, MLP*

1. INTRODUCTION

Understanding scenes is one of computer vision's main objectives. The core task of computer vision is to automatically create captions for images, and this is its most crucial and vital component. Why is picture captioning necessary? When a question arises, it is possible to provide a response because, as humans, we can quickly comprehend what is happening in a picture by looking at it. For Machines, however, it is not the same because doing so involves knowing the semantic connections between the numerous things in the image. A descriptive text is generated for a given image as part of the intriguing artificial intelligence challenge known as caption creation. It uses two computer vision approaches to comprehend the image's content and a language model from the field of natural language processing to translate that understanding into the appropriate sequence of words. In this paper we explored different methods of image captioning as well as classification tasks.

1.1 Objectives

- To recognize the objects present in the image
- To identify the context of the object.
- To identify prepositions which refer to spatial relationships between object-object pairs.
- To generate meaningful captions considering each of the image objects and their context.
- To develop an attention-based model by first describing their common framework.

1.2 Applications

Some of the prominent applications of image captioning are listed below.

1. Aid to the Blind : Generate caption to an image and text to-speech convert the caption to aid the blind
2. Self driving cars : Caption the scene around the car is given as input to the self driving system which gives a boost to the self driving car performance.
3. Google image search : Every image can first be converted to caption and then search can be performed based on that caption.
4. To generate captions for CCTV clips : Description giving an insight on the CCTV clip
5. Recommendations in editing applications : Giving image editing recommendations based on the caption describing the scene in the image

2. LITERATURE SURVEY

Since the advent of the Internet and its extensive use as a platform for sharing photographs, the issue of image captioning has persisted, as have the solutions that have been suggested. Researchers from various angles have proposed a large number of algorithms and strategies.

The literature is divided into template-based techniques and neural-based approaches in accordance with the description of the state of the art in [1]. The approaches that fill in gaps in phrases based on recognised objects, visual features, and scenes fall under the first group, which also includes early attempts to solve the problem of image captioning.

Due to recent advancements in image recognition and NMT, neural-based techniques are now dominating. The sequence-to-sequence encoder-decoder models that are currently favoured in machine translation are the source of inspiration for these techniques.

[1]. An attention-based model that automatically learns to describe the content of images was developed by Yoshua Bengio, Aaron Courville, Jimmy Lei Ba, Ryan Kiros, Richard S. Zemel, and Kelvin Xu. Kyunghyun Cho, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. In this study, the model is trained stochastically by maximising a variational lower bound and deterministically using conventional backpropagation approaches. Flickr8k, Flickr30k, and MS COCO were the datasets used.

[2]. In this study, Jiuxiang Gu, Gang Wang, Jianfei Cai, and Tsuhan Chen proposed a language CNN model that excels in image captioning and is appropriate for statistical language modelling tasks. Language CNN is supplied with all of the prior words and can represent the long-range dependencies in historical words, which are crucial for picture captioning, in contrast to earlier models that predict next word based on one previous word and hidden state. Two datasets, Flickr30K and MS COCO, are used to validate the efficacy of this method. Extensive experimental results demonstrate that this method competes favourably with state-of-the-art techniques and outperforms the standard recurrent neural network based language models.

[3]. Three deep neural network-based image captioning techniques are discussed in this paper: CNN-RNN based, CNNCNN based, and reinforcement-based framework. The authors then described the evaluation metrics, summarised the advantages and significant problems, and introduced the sample work for each of the three top techniques.

[4].The current state of the art for neural machine translation is represented by attentional recurrent neural networks, self-attentional transformers, and fully-convolutional networks are the focus of this paper's exploration.

[5]. The Object Relation Transformer is an encoder-decoder architecture developed specifically for picture captioning by Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares that incorporates knowledge of the spatial connections between input recognised items through geometric attention. This work uses baseline comparison and an ablation investigation on the MS-COCO dataset to quantitatively demonstrate the value of geometric attention. Finally, it demonstrates how superior captions that exhibit improved spatial awareness can be produced through geometric attention.

[6]. The attention mechanism, which is a crucial component of computer vision and has lately become widely used to tasks involving the creation of image captions, is the subject of this paper's summary of related techniques. The benefits and drawbacks of these methods are also discussed, along with the most popular datasets and evaluation standards in this area. This research concludes by highlighting several unresolved issues with the image caption task.

3. TOOLS AND TECHNOLOGIES

1.Computer Vision:
The study of computer vision focuses on simulating some of the complexity of the human visual system so that computers can recognise and analyse items in pictures and videos in a similar manner to how people do. The quantity of data that we produce now and use to train and improve computer vision is one of the key variables influencing its development. Pattern recognition is the foundation of computer vision. Therefore, one method of teaching a computer to comprehend visual data is to feed it images, many images, thousands, or even millions, that have been labelled, and then subject those to different software techniques, or algorithms, that enable the computer to look for patterns in all the elements that relate to those labels.

2.Convolutional Neural Network (CNN):
CNN is a deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and generates the feature vector. There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. There are numerous CNN architectures that may be used, and these architectures have been essential in creating the algorithms that will continue to power AI as a whole in the near future. Some of them have been listed below:

1. Resnet
2. Google Net
3. VGG Net
4. ZF Net
5. Le Net

3.LSTM (Long Short Term Memory):
Long Short Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) which can learn order dependence in sequence prediction problems. The most frequent applications of this are in difficult issues like speech recognition, machine translation, and many others. When training conventional RNNs, this issue was observed because as we go further into a neural network, if the gradients are very small or zero, little to no training can occur, resulting in poor predicting performance. Since there may be lags of uncertain length between significant occurrences in a time series, LSTM networks are well suited for categorising, processing, and making predictions based on time series data. LSTM is way more effective and better compared to the traditional RNN as it overcomes the short term memory limitations of the RNN. LSTM can carry out relevant information throughout the processing of inputs and discards non relevant information with a forget gate.

4.Keras :
Keras is python based open-source neural network library. It can be used with TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML as a foundation. Its user-friendliness, modularity, and extensibility are its main design goals as it aims to facilitate quick experimentation with deep neural networks. It was created as a component of the ONEIROS (Openended Neuro-Electronic Intelligent Robot Operating System) research initiative.

5.Numpy:
NumPy is the fundamental package for scientific computing in Python. A multidimensional array object, various derived objects (like masked arrays and matrices), and a variety of routines for quick operations on arrays are provided by this Python library. These operations include discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more. The n-d array object serves as the foundation of the NumPy library. This contains homogeneous n-dimensional arrays of data kinds, with many operations carried out in compiled code for speed.

6.Tensorflow:
TensorFlow is an opensource software library in Python for numerical computation using data flow graphs. The graph's nodes stand in for mathematical processes, while its edges stand in for the multidimensional data arrays

(tensors) that are transmitted between them. With a single API, you may use the flexible architecture to deploy compute to one or more CPUs or GPUs in a desktop, server, or mobile device. It is employed in machine learning programmes like neural networks.

4.DATASETS

4.1.MSCOCO:

The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key point detection, and captioning dataset. There are 328K photos in the dataset. Splits: In 2014, the MS COCO dataset's initial version was made public. There are 164K photos total, divided into 83K training images, 41K validation images, and 41K test images. All of the previous test images as well as 40K fresh images were included in the 81K extra test set that was released in 2015. In 2017, the training/validation split was modified from 83K/41K to 118K/5K based on community feedback. The same photos and annotations are used in the new split. A portion of the 2015 test set's 41K photos make up the 2017 test set. The 2017 edition also includes a fresh 123K image unannotated dataset.

Annotations: The dataset has annotations for

1.Object detection: bounding boxes and per-instance segmentation masks with 80 object categories.

2.Image captioning: describing the pictures in plain terms.

3.Keypoints detection: containing more than 200,000 images and 250,000 person instances labeled with key points (17 possible key points, such as left eye, nose, right hip, right ankle)

4.Stuff image segmentation: per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky.

5.Panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),

6.Dense pose: more than 39,000 images and 56,000 person instances labeled with Dense Pose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images. Some sample images of training set are as shown below.

Fig.1-Two horse riders riding the horses in the green court

Fig.2-Kitchen scene

4.2.Flickr 8k:

8,000 photos with five different captions, each describing the key elements and actions in the image, make up this new benchmark collection for sentence-based image description and search. The photographs were painstakingly chosen to represent a diversity of scenarios and circumstances from six different Flickr groups, and they often don't feature any famous individuals or places. The data collection includes two folders:

1.Flickr8k Dataset: Consists of 8092 JPEG photos in all, of various sizes and forms. 1000 are used for testing, 1000 are used for development, and 6000 are used for training.

2.Flickr8k text: Contains text files that describe the test and train sets. Each image in Flickr8k.token.txt has five captions, for a total of 40460 captions.

5.SYSTEM DESIGN AND ARCHITECTURE

This section provides an in-depth analysis in the process of System Design in context of our project. We also describe the two variants of our attention-based model by describing their common framework. The section also provides a detailed overview of the architecture of each technique including data flow, layer orientation and layer order.

5.1. ENCODER

Encoder model takes a single raw image and generates a caption y encoded as a sequence of 1-of-K encoded words.

$y = \{y1, \ldots, yC\}$, $yi \in R^K$ where K is the size of the vocabulary and C is the length of the caption.

A set of feature vectors, which refer to as annotation vectors, are extracted using a convolutional neural network. Each of the L vectors the extractor creates is a D-dimensional representation of an area of the image.

$a = \{a1, \ldots, aL\}$, $ai \in R^D$

Instead of utilising a fully connected layer, we extract features from a lower convolutional layer in order to obtain a relationship between the feature vectors and specific regions of the 2-D picture. This allows the decoder to selectively focus on certain parts of an image by selecting a subset of all the feature vectors. There are several CNN architectures available. We have tried encoder implementation with two architectures as follows.

5.1.1.Inception Net:

With the use of transfer learning, Inception-v3, an expanded version of the well-known Google Net, has successfully classified data in a number of biological applications. Inceptionv3 suggested an inception model that concatenates many convolutional filters of various sizes into a single filter, following GoogLeNet. Such a design reduces the complexity of the computation by lowering the number of parameters that must be taught. Fig. 3 depicts Inception-fundamental v3's architecture. The inception module's purpose is to perform the function of a multilayer feature extractor by computing 1x1, 3x3, and 5x5 convolutions within the same network module. After being stacked along the channel dimensions, the output of these filters is then passed into the network's next layer. Inception V3 is mostly used in image recognition tasks in which it has more than 78% accuracy.

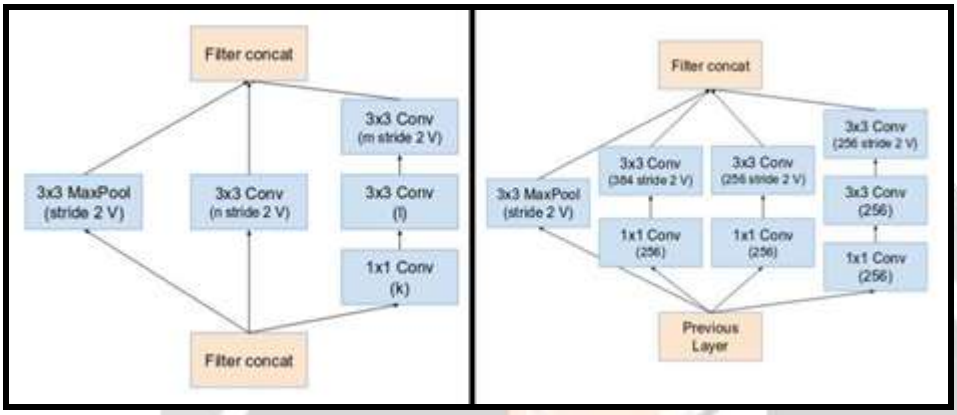

Fig.3-InceptionNet Architecture

5.1.2. Resnet50:

ResNet50 is a Resnet model version having 48 Convolution layers, 1 MaxPool layer, and 1 Average Pool layer. There are 3.8 x 109 floating point operations available. This architecture can be utilised for computer vision applications like object localisation, object identification, and image categorization. Additionally, this approach can be used to add depth to jobs that are not computer vision-related and cut down on their processing costs.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig.4-Resnet50 Architecture (Layer Structure)

As shown in the Fig.4 the Resnet 50 architecture contains the following elements:

i. A convolution with 64 distinct kernels, each having a stride of size 2, and a kernel size of 7 * 7, giving us 1 layer.
ii. Max pooling is shown after that, with a stride size of 2.
iii. There are three levels in the next convolution: a 1 * 1, 64 kernel, a 3 * 3, 64 kernel, and finally a 1 * 1, 256 kernel. These three layers are repeated a total of three times, giving us nine layers in this phase.
iv. The next kernel is a 1 * 1,128 kernel, followed by a 3 * 3,128 kernel, and finally a 1 * 1,512 kernel. This procedure was performed four times for a total of 12 layers.
v. After that, we get a kernel of size 1 * 1,256, followed by two more kernels of size 3 * 3,256 and 1 * 1,1024, which are repeated six times to give us a total of 18 layers.
vi. Next, a 1*15,512 kernel was added, followed by two more kernels of 3*3,512 and 1*1,2048. This process was done three times, giving us a total of nine layers.
vii. Following that, we perform an average pool, finishing with a fully linked layer made up of 1000 nodes and a softmax function to give us one layer.

5.2. DECODER:

In order to create a caption, we employ a long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) that generates one word at each time step conditioned on a context vector, the prior hidden state, and the previously created words.

5.2.1 LSTM(Long Short Term Memory):

Recurrent neural networks of the Long Short-Term Memory (LSTM) type can learn order dependence in sequence prediction issues. This behaviour is essential for solving complicated problems in areas like speech recognition and machine translation, among others.

The following equations, in which LSTM(xt) yields pt+1 and the tuple (mt, ct) is given as the current hidden state to the next hidden state, can be used to define the LSTM function

$$it = \sigma(Wixxt + Wimmt-1) \quad ft = \sigma(Wfxxt + Wfmmt-1)$$

$$ot = \sigma(Woxxt + Wommt-1)$$

$$ct = ft \odot ct-1 + it \odot \tanh(Wcxxt + Wcmmt-1)$$

$$mt = ot \odot ct \quad pt+1 = Softmax(mt)$$

The model may deliberately disregard certain prior memory cell states using the forget gates, and it can do the same with current input using the input gates. The model can then filter the current memory cell for its ultimate hidden state using the output gate. When these gates are combined, it is possible to learn long-term dependencies, reset them in response to specific inputs, and avoid vanishing and bursting gradients.

6.SYSTEM IMPLEMENTATION:

This section covers the project's System Implementation phase. This section also includes a detailed description of each step. Data pre-processing, feature extraction from the images model implementation, and the training procedure are the steps that have been covered in this section.

6.1 Data Preprocessing

6.1.1. Data Preprocessing-Images:

Images are input (X) to our model. But any input to a model must be given in the form of a vector. So we need to convert every image into a fixed sized vector which can then be fed as input to the convolutional neural network. For this purpose, we opt for transfer learning by using the InceptionV3 model created by Google Research.

This model was trained on ImageNet dataset to perform image classification on 1000 different classes of images.

6.1.2. Data Preprocessing- Captions:
Captions are something that we want to predict. So during the training period, captions will be the target variables (Y) that the model is learning to predict. But the prediction of the entire caption, given the image, does not happen at once. We will predict the caption word by word. Thus, we need to encode each word into a fixed size vector. In the preprocessing of captions, the following steps needs to be followed.

- Convert to lowercase captions
- Remove Punctuation
- Remove words with numbers
- Remove 1-character words
- Add 'Start' and 'End' tokens or tags
- Convert words to word ID with help of tokenizer
- Pad to same length

6.2. Model Implementation:
The model implementation consists of 4 logical components that are encoder, attention and decoder and Sentence Generator. 8.2.1 Encoder Implementation The CNN encoder architecture used to produce a feature vector from input image is implemented by taking the size or dimension of the embedding or the feature vector that needs to be produced as output. A layer is added to the pretrained CNN model using this embedding size. Any activation function can be used to this added layer (reLu activation function used in our implementation)

6.2.1.Decoder Implementation:
Decoder is called after encoder implementation has been completed. Decoder takes the captions as input after the Embedding layer. RNN built with GRU or LSTM units are used in Decoder. Decoders generate words of output sequence with the help of an attention model. When the decoder is run it passes the hidden state of previous timestep to the attention model. This attention score will help Decoder to focus on the most relevant parts of the picture and generate the appropriate output word.

6.2.2.Attention Implementation:
Implementation of the attention model is done by subclassing the attention layer from tensorflow. Multi-Head Attention and Additive Attention models can be used from tensorflow (We used Internal Multi-Head Attention in out implementation).As the Decoder generates each word of the output sequence, the Attention model helps it to focus on the most relevant part of the image for generating that word. As the hidden state information is given by Decoder. The Attention model produces an Attention Score that assigns a weight to each pixel of the encoded image. The higher the weight for a pixel, the more relevant it is for the word to be output at the next timestep (For example, if the target output sequence is "A boy is eating an apple", the girl's pixels in the photo are highlighted when generating the word "boy", while the apple's pixels are highlighted for the word "apple") Three weights(W1,W2,V) layers are used in the Attention model. Weight layers W1 and W2 take feature vector and hidden state from Decoder as input respectively. After weight computation is done activation function (tanh in our implementation) is performed which combined is called as attention-hidden-layer.

6.3 Model Training

6.3.1. Process input sequence over multiple timesteps

1. The input sequence(caption) is passed through the Embedding layer and then combined with the Attention Score.
2. The combined input sequence is fed to the Sequence Decoder, which produces an output sequence along with a new hidden state.
3. The Sentence Generator processes the output sequence and generates its predicted word probabilities.
4. This cycle is repeated for next time step using Decoder's new hidden state until 'End' token is reached.

6.3.2. Teacher Forcing
One of the important points while training a model with sequence is Teacher Forcing. Ordinarily, when the model is fully trained, the output sequence from this timestep is used as the input sequence for the next timestep. This allows the model to predict the next word based on the previous words predicted so far However, when we use this concept

while training the model, any errors made by the model in predicting an output word in this time step would be carried forward to the next timestep. This error propagates while predicting successive words. To avoid this problem, we can use the ground truth that the captions are available to us during training, we use a technique called Teacher Forcing. The correct expected word from the target caption is added to the input sequence for the next time step rather than the model's predicted word when the predicted word is wrong. This way there will be no propagation of errors causing damage during training.

7.SYSTEM TESTING AND RESULT ANALYSIS

This section covers the testing of all algorithms we implemented in this project. So this includes additional algorithms we tried before diving into deep neural networks, these additional algorithms include MLP and RBF.

7.1.MLP Testing:

As feature extraction is the first part of our algorithm, instead of diving into pre-trained deep neural networks, we tested MLP classifier training with a dataset of 10 classes, to use it for feature extraction. The results of MLP are as shown below. Although it is good for classification with high accuracy of 53.4% as shown in fig 5, we need more accuracy in our project as we deal with huge number of objects while captioning.

Table 1

No of Hidden Layers	Neurones count - 1	Neurones count - 2	Neurones count - 3	Neurones count - 4	Epochs	Batch Size	Train Accuracy	Test Accuracy
3	1000	300	100	-	50	50	73%	49%
3	2058	1382	931	-	50	50	92%	46%
3	350	200	50	-	50	50	67%	49%
2	350	50	-	-	50	50	61.47%	50.14%
2	350	50	-	-	50	100	62.13%	49.44%
2	350	50	-	-	20	250	51.93%	48.96%
2	350	50	-	-	50	250	61.65%	50.62%
2	350	50	-	-	100	250	70.16%	49.58%
4	500	350	200	50	50	50	72-77%	48-49%
4	500	350	200	50	20	50	62-67%	49-51%
4	2058	1382	931	630	20	250	67%	52.34%
3	1000	500	250		30	250	74%	53.39%

Fig.5-MLP train and test accuracy

To further increase accuracy ,we tested a retraining process which takes the wrong predictions from the test set and appends those to the training set to retrain MLP model. Although the results increase after retraining process it is still less overall(fig.6).

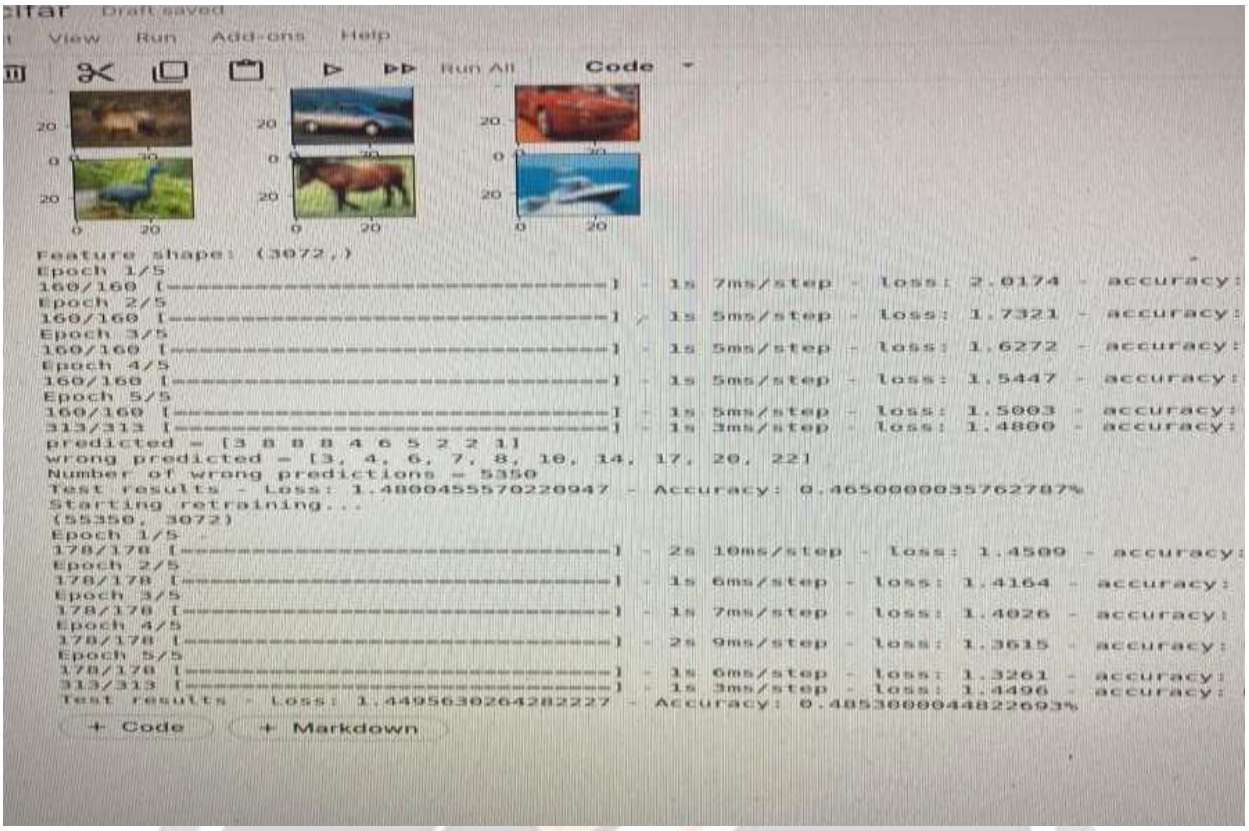

Fig.6-Retraing MLP Results

7.2 RBF Testing:

Radial Basis Function(RBF) is the combination of unsupervised learning with neural networks. As it has shown better result in most of the machine learning projects we tried RBF before DNN.RBF is trained taking the distance between input and some fixed point which is the centroid of the clusters. The centroid that is closest to the input is the output class of that input. Since RBF includes calculation of centroid in each training step we took smaller dataset with less number of epochs. The results of RBF training with retraining is shown in fig 7.1 and fig.7.2.

Fig.7.1.RBF Training Results

Fig.7.2-RBF training Results

Although the results improve with a large dataset and with a higher number of epochs included with some retraining steps. The RBF algorithm takes a huge amount of time for a larger dataset. Since captioning images requires a huge dataset and deals with more classes. RBF will be computationally expensive. Considering results of the above algorithms, we shifted over to pretrained deep neural networks.

7.3 Show, Attend and Tell Testing:

We used the following pretrained networks in Show, attend and tell algorithm. Testing of above algorithm was done with 2 different datasets.

- InceptionV3(ImageNet) as encoder model.
- Pretrained RNN with GRU and LSTM units as decoder model.
- Bahdanau Attention and Multi-Head Attention as attention model.

We added additional layers to pretrained networks.

7.3.1 flickr8k dataset testing:

The result of the algorithm with flickr dataset is shown below.

Fig.8 - Train Accuracy for Flickr8k dataset

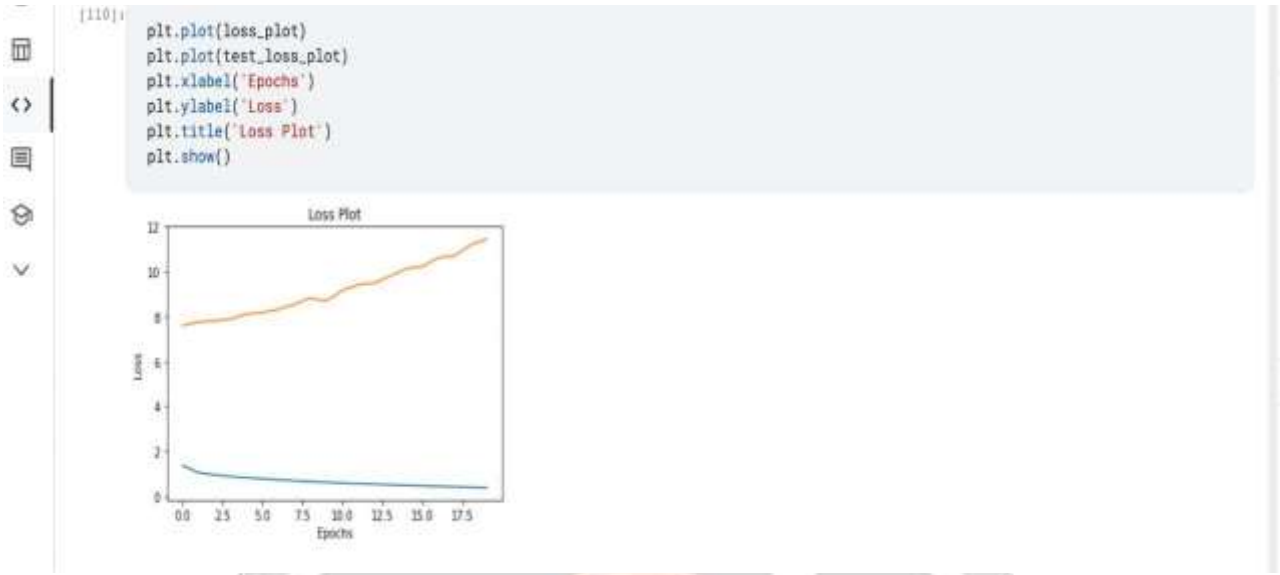

Fig.9-Loss Plot on Flickr 8k dataset

Fig.10-Real and predicted captions for flickr8k

7.3.1 MSCOCO dataset testing:

MSCOCO contains more number of images than flickr8k. Results of running above algorithm on MSCOCO dataset is shown in fig10, fig11 and fig12.

```
Epoch 16 Batch 0 Loss 0.4032
Epoch 16 Batch 100 Loss 0.3536
Epoch 16 Batch 200 Loss 0.3468
Epoch 16 Batch 300 Loss 0.3743
Epoch 16 Batch 400 Loss 0.3402
Epoch 16 Loss 0.361785
Time taken for 1 epoch 261.83709788322245 sec

Epoch 17 Batch 0 Loss 0.4176
Epoch 17 Batch 100 Loss 0.3698
Epoch 17 Batch 200 Loss 0.3330
Epoch 17 Batch 300 Loss 0.3291
Epoch 17 Batch 400 Loss 0.3455
Epoch 17 Loss 0.343761
Time taken for 1 epoch 260.50080728530625 sec

Epoch 18 Batch 0 Loss 0.3582
Epoch 18 Batch 100 Loss 0.3327
Epoch 18 Batch 200 Loss 0.3327
Epoch 18 Batch 300 Loss 0.3389
Epoch 18 Batch 400 Loss 0.3103
Epoch 18 Loss 0.323801
Time taken for 1 epoch 262.3576545715332 sec

Epoch 19 Batch 0 Loss 0.3654
Epoch 19 Batch 100 Loss 0.3019
Epoch 19 Batch 200 Loss 0.2954
Epoch 19 Batch 300 Loss 0.2945
Epoch 19 Batch 400 Loss 0.2811
Epoch 19 Loss 0.305946
Time taken for 1 epoch 322.05827236175537 sec

Epoch 20 Batch 0 Loss 0.3636
Epoch 20 Batch 100 Loss 0.3083
Epoch 20 Batch 200 Loss 0.3453
Epoch 20 Batch 300 Loss 0.2718
Epoch 20 Batch 400 Loss 0.2610
Epoch 20 Loss 0.290522
Time taken for 1 epoch 286.9654648197754 sec
```

Fig.11-Time taken for each epoch.

```
[17]: '''Visualize training'''

plt.plot(loss_plot)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss Plot')
plt.show()
```

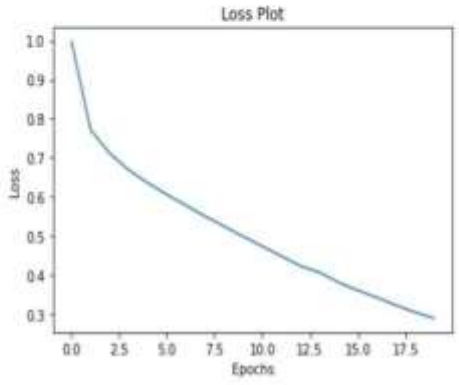

Fig.12-Loss plot for MSCOCO dataset

Observations:

Difficulties faced during above implementations are:

- Limited time on GPU access.
- Limited size of HDD access lesser than size of dataset.
- Training deep neural network models took more time than expected which made it difficult to change parameter values.
- GPU not freely available.

8.CONCLUSIONS

There are numerous current research initiatives striving for more precise image feature extraction and semantically superior sentence creation because automatic image captioning is still in its infancy. Despite the fact that not all of the algorithms we tried to build for this project were successful, we learned a lot of principles and their practical applications. Our initial attempt at employing MLP for picture categorization had an accuracy of 53%. As we deployed far deeper networks with more training, project accuracy rose. The large diversity of objects in the Flickr 8k and MSCOCO datasets made it easier to develop captioning models. One of the challenges encountered in the research was the way of creating caption as opposed to anticipating a caption that is already existent. In general, the more diverse training dataset the network has seen, the more accurate the output will be. There can be potential improvements if given more time. We all agree this project ignites our interest in application of Machine Learning knowledge in Computer Vision and expects to explore more in the future.

9. FUTURE WORK

1.Build a web or mobile application which accepts image as input and provide caption for that specific image
2.Image captioning on real-time videos.

3. Captioning key frames in a CCTV footage.

10. REFERENCES

[1]. Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, Yoshua Bengio, University of Toronto, CIFAR, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" conference paper published in 2015.

[2]. Jiuxiang Gu(1), Gang Wang(2), Jianfei Cai(3), Tsuhan Chen(3), (1) ROSE Lab, Interdisciplinary Graduate School, Nanyang Technological University, Singapore (2) Alibaba AI Labs, Hangzhou, China (3) School of Computer Science and Engineering, Nanyang Technological University, Singapore, "An Empirical Study of Language CNN for Image Captioning" published in 2017 IEEE International Conference on Computer Vision.

[3]. Shuang Liu , Liang Bai , Yanli Hu and Haoran Wang ,College of Systems Engineering, National University of Defense Technology,410073 Changsha, China, "Image Captioning Based on Deep Neural Networks" published in MATEC Web of Conferences 232, 01052 (2018).

[4]. Loris Bazzani- Amazon Research Core AI, Tobias Domhan- Amazon Research Machine Translation, Berlin , Felix Hieber-Amazon Research Machine Translation, Berlin, "Image Captioning as Neural Machine Translation Task in SOCKEYE" published in 2018.

[5]. Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares Yahoo Research San Francisco, CA, 94103, "Image Captioning: Transforming Objects into Words" published in 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada

[6]. Haoran Wang(1) ,Yue Zhang(1), and Xiaosheng Yu(2) ,(1) College of Information Science and Engineering, Northeastern University, China,(2) Faculty of Robot Science and Engineering, Northeastern University, China, "An Overview of Image Caption Generation Methods" A Research article published in 2020.

.

.