Image Captioning using Neural Nets

Project report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

Information Technology

By

Nidhish Manchanda (161459)

Under the supervision of Mr. RizwanurRehman

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Image Captioning using Neural Nets"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology**submitted in the Department of Computer Science & Engineering and Information Technology,Jaypee University of Information Technology, Waknaghat is an authentic record of our own work carried out over a period from July 2019 to December 2019 under the supervision of **Mr. RizwanurRehman.**

The matter embodied in the report has not been appeased for the award of any otherdegree or diploma.

Nidhish Manchanda

161459

This is to certify that the above affirmation made by the candidate is true to the best of my knowledge.

Mr. Rizwan ur Rehman

Assistant Professor (Grade- II)

Department of Computer Science & Engineering and Information Technology

Dated: 27th May2020

Acknowledgement

We would like to take the opportunity to thank and express our deep sense of gratitude to our mentor and project guide **Mr. RizwanurRehman**for his immense support and valuable guidance without which it would not have been possible to reach at this stage of our major project.

We are also obliged to all the faculty members for their valuable support in their respectivefields which helped us in reaching at this stage of our project.

Dated: 27th May 2020

Contents

List of Figures

List of Graphs

List of Tables

Abstract

Image captioning is describing an image with a caption in natural language. The description captures not only the objects contained in an image, but it also must express how these objects relate to each other. This fundamental problem connects computer vision and natural language processing. In this project, we explore ways to generate captions for images given as input. We use deep recurrent architecture that uses recent advances in computer vision. We also explore newer methods of model evaluation using user defined caption captured through an online platform.

This project is important in achieving scene understanding from a video or image. It serves various applications for helping visually impaired people, creating medical reports from images of X-rays, CT- scans etc., filtering image content on the internet and many more applications.

Chapter 1 INTRODUCTION

1.1 Introduction

Machine Learning is a subset of Artificial Intelligence. It uses statistical techniques to provide computers the ability to learn with the help of large amounts of data. While Machine Learning has reduced the burden on programmers to explicitly program the computers, it has touched the areas which have never been explored before. This has led to various technological advances.

There are 3 types of Machine Learning algorithms:

a) Supervised Learning Algorithms

b) Unsupervised Learning Algorithms

c) Reinforcement Learning Algorithms

Automatic generation of a description in natural language is called Image Captioning. This task of image captioning is a mixture of both supervised and unsupervised machine learning techniques. Here not only the algorithm should detect the objects in an image but also should be able to detect how the objects relate with each other which is way harder than just classification. Also, the knowledge should be expressed in English so we also require a language model.

1.2 Problem Statement

In this project, we explore ways to generate captions for images given as input. We use deep neural networks. Further, for better results we use clustering upon our datasets so as to reduce size of the dataset which in turn will reduce overfitting and computation time. The description must catch not just the objects present in the image, but also must describe how these objects are related to one another.

1.3 Objectives

Our objective is to achieve maximum accuracy in generating sensible captions. We also explore newer and better ways of evaluating the model on different datasets. We use clustering on our dataset to overcome overfitting and reduce computation cost involved.

Figure 1: Sample Image for understanding

Given an image like the example above, the goal is to generate a caption such as "a surfer riding on a wave". To achieve this task, we useencoder-decoder model, where the encoder outputs feature vectors of the input images and the decoder outputs the generated captions on receiving these image vectors as input.

Figure 2. Encoder-Decoder Model for Captioning images

1.4 Methodology

The idea behind this project is to use deep learning techniques to generate captions from input images. We explore different CNN and RNN architectures to achieve this. The main technique used here is to extract features of images using VGG-16 convolutional neural network. In a way CNN acts as an encoder here. The RNN model decodes these features to predict captions word by word.

The dataset used here by us in the MS COCO dataset which has roughly ~82000 images. Also reading related work done already in image captioning the common problem of overfitting came to fore. So, we have decided to go forth with a subset of 18,000 images from the 82,000 available images. Also, with given hardware constraints we devised a way to build our dataset by performing Hierarchical clustering on the images based on their features and selecting images from the clusters. This we hope will result in less overfitting and more generalization. Also, selecting images from clusters will lead to less training data and will require less computation power.

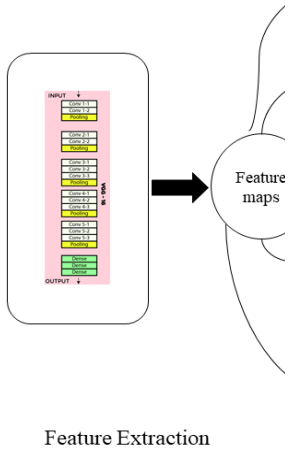


Figure 3. Overall Project Flow

Chapter 2 LITERATURE SURVEY

2.1 Show & Tell: A Neural Image Caption Generator

2.1.1 Introduction

Automatically generating the description of an image using perfect sentences in natural language is indeed a very complex task. It can however be helpful to many individuals like blind people to able to get the context of images. An image description must catch the articles in a picture, also additionally should communicate how they are identified with one another just as their characteristics and the exercises they are engaged with.

Figure 4. Example of Show and tell image caption generatiorn

2.1.2 Model

In this paper, probabilistic framework is proposed to generate captions from images. Using a strong sequence model, better results can be achieved by increasing the probability of the right interpretation given a sentence in an “end-to-end” fashion – both for training as well as inference. Such models utilize RNN that encodes the different length contribution to a constant dimensional vector, and utilizations this to "decode" it to the ideal yield caption. In this manner, it is normal to utilize a similar approach where, given a picture, one applies a similar rule of "making an interpretation of" it into its description. Along these lines, this paper proposes to legitimately amplify the probability of the right description given the picture.

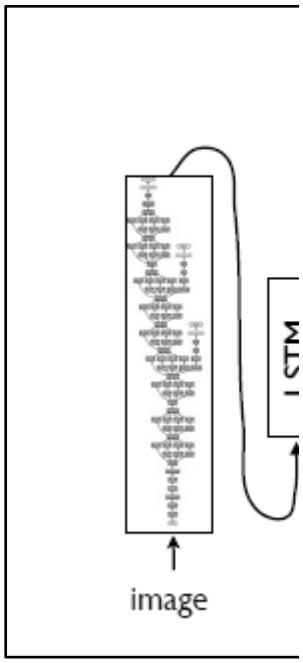


Figure 5. Working of Show and tell

2.1.3 Datasets

Various datasets which comprise of images and sentences in English describing these images are utilized for assessment. These include MSCOCO, Flickr8k, Flickr30k PASCAL VOC 2008 and SBU datasets. All these datasets, except for SBU have 1000s of images, each having at least 5 sentences which are moderately visual and fair. For SBU, a 1000 pictures are considered for testing and training is done on the rest.

Dataset name
Pascal VOC 2008
Flickr8k [26]
Flickr30k [33]
MSCOCO [20]
SBU [24]

Table 1. Datasets used

2.1.4 Results

The model is data driven and it is also trained end-to-end, and since a lot of datasets are involved, questions such as “what sort of transfer learning it is ready to accomplish”, and

"how it would manage models having weak labels" came up. Therefore, experiments on five diverse datasets were performed, which empowered a profound comprehension of the model.

Table 2. Evaluation using BLEU-4, METEOR and CIDER metrics

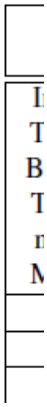


Table 3. Evaluation w.r.t approaches

2.1.5 Conclusion

An end-to-end neural network system that can naturally see a picture and produce a sensible description in plain English is introduced. NIC depends on CNN that encodes a picture into a minimized portrayal, trailed by a repetitive neural system that creates a comparing sentence. Investigations on a few datasets show the vigor of NIC as far as subjective outcomes (the created sentences are entirely sensible) and quantitative assessments, utilizing either positioning measurements or BLEU.From these experiments it is clear that, the performance of approaches like NIC will increase as the size of the datasets goes up.

2.2 Show, Attend & Tell

2.2.1 Introduction

Thus, caption generation has long been seen asa tough problem. It is an essential test for Machine Learning, since it attempts to imitate the human-likeability to pack all the measures of the data into natural language.

In this paper, the Caption generation approach attempts to include a kind of attention with two variants: a “soft” attention mechanism and a “hard” attention mechanism.

Figure 6. Show, Attend and Tell Example for image captioning

2.2.2 Model

Encoder

To acquire a relationship between the component vectors and parts of the 2-D picture, highlights from a lower convolutional layer are removed. The decoder focuses its attention on some portions of an image by sampling a group of all the feature vectors.

Decoder

They utilized a LSTM network that delivered a subtitle by anticipating each word in turn dependent on a setting vector, the past concealed state and the recently anticipated words. In simple words, a context vector is a depiction of the correct part of the image at time t. This relevant part is picked up from the feature map which is fed to LSTM. A positive weight is generated for each relevant part of the image which is either interpreted as

probability or as the importance of that location. The weight of each location is then calculatedusing the attention model which uses a multi-layer perceptron is used.

2.2.3 Datasets

Datasets that are used during training process are Flickr8k dataset which has 8K data points, Flickr30k dataset which has 30K images and Microsoft COCO dataset which has over 82K. The Flickr8k and the Flickr30k datasets have 5 labels for every picture, while in the MS COCO dataset, images have at least 5 labels, overabundance of which is disposed of to keep up consistency over all datasets.For all experiments, a fixed vocabulary size of 10K is used.

2.2.4 Results

Performance of the model on the Flickr8k, Flickr30k and MS COCO turned out to be state of the art. Furthermore, there was improvement in the cutting edge execution of METEOR evaluation metrics on MS COCO that was theorized to be associated with a portion of the regularization methods.

Figure 7 Show, Attend and Tell, working of attention model

2.2.5 Conclusion

An attention based approach that gives incredible presentation on three benchmark datasets utilizing the METEOR and BLEU evaluation metrics is gotten in this paper.

Furthermore, something that was obvious from the model was that the scholarly attention might be misused to produce more prominent interpretability into the models age procedure, and display that the trained captions relate really well to the actualcaptions.

2.3 Image Captioning with Semantic Attention

2.3.1 Introduction

It requires a degree of picture understanding that works out in a good way past picture grouping and item identification to produce a significant natural language description of a picture. It associates Computer Vision with Natural Language Processing (which are two significant fields in Artificial Intelligence) which makes the difficult all the more intriguing.

Figure 8 Image Captioning with Semantic Attention

2.3.2 Model

In this model both top-down just as bottom-up features are removed from the picture. Moreover, a lot of credit finders is hurried to get a rundown of visual traits which are well on the way to show up in the picture. Each ascribe compares to a section in the vocabulary

2.3.3 Database

Flickr30k and MSCOCO is chosen as the datasets in this paper. Flickr30k has more than 30,000 images and the MSCOCO has over 82,000 images. For Flickr30k, each image is given five captions and for MSCOCO, each image has at least 5 captions. A publicly available split of training, testing and validating sets is used to make the results comparable to others, for both Flickr30k as well as MS-COCO datasets.

2.3.4 Results

The impact of every one of the individual attention modules on the last execution is assessed by killing one of the attention modules and not killing the other one in the ATT-FCN model. The 2 distinct models are prepared on the MS-COCO dataset. On certain metrics, the presentation when utilizing output attention is somewhat better than just utilizing input layer attention. In any case, the blend of these two attentions improves the presentation by certain percent on pretty much every measurement. This means that the attention components at input and the output layers are not the equivalent, and every one of them take into account various parts of visual properties. In this manner, joining the two may help produce a more extravagant interpretation of the specific circumstance and accordingly lead to better execution.

Figure 9. Image and the generated caption

2.3.5 Conclusions

In this paper, image captioning that accomplishes best in class execution across well-known standard benchmarks is proposed. Not quite the same as past work, this approach

joins top-down and the bottom-up systems to extricate more important data from an image, and connects them with a RNN that can specifically goes to on rich semantic traits identified from the image. The genuine intensity of this model is its capacity to go to on specific angles and breaker worldwide and neighborhood data to deliver a superior caption.

2.4 ConvNets

The Convolution Operation

The most striking distinction between a convolution layer and fully connected layer is that the full connected ones learn global patterns withintheir input feature space while convolution layers learn local examples on account of pictures, designs found in little second windows of the inputs.

Convolutions are outlined by 2 important parameters:

• Size of the portions extracted from the inputs- These are usually 3x3 or 5x5. within the example, they were 3x3, that could be a common selection.

• Depth of the feature map- the amount of filters calculated by the convolution.

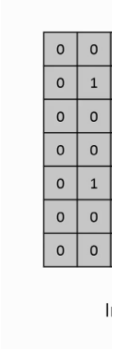


Figure 10. Feature maps converted from a portion of input image

The max pooling operation

The role of max-pooling is to aggressively down sample feature maps. It comprises of separating windows from the input feature maps and outputting the maximum estimation of each channel. It's like convolution adroitly, then again, actually as opposed to changing nearby fixes by means of a scholarly direct change, they are changed by means of a hardcoded max tensor activity.

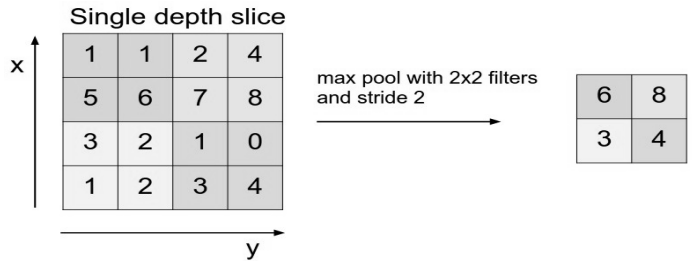


Figure 11. Max pooling

2.5 Very Deep Convolutional Networks for Large-Scale Image Recognition

The input to the ConvNets is a fixed-size 224×224 RGB image during training. The main pre-processing done is normalizing the RGB esteems, figured on the training dataset, from every pixel. The image is worked upon by a heap of convolutional (conv.) layers, where channels are utilized with a little receptive field: 3×3

ConvNet		
A	A-LRN	B
11 weight layers	11 weight layers	13 weigh layers
input (224 ×		
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**
m		
conv3-128	conv3-128	conv3-12 **conv3-12**
m		
conv3-256 conv3-256	conv3-256 conv3-256	conv3-25 conv3-25
m		
conv3-512 conv3-512	conv3-512 conv3-512	conv3-51 conv3-51
m		
conv3-512 conv3-512	conv3-512 conv3-512	conv3-51 conv3-51
m		
F		
F		
F		
so		

Figure 12 ConvNet Configuration

In one of the layers the 1×1 convolution filters were utilized, which can be viewed as a straight change of the input channels (trailed by non-linearity). The convolution stride is consistently 1 pixel; the spatial cushioning of convolution layer input is with the end goal that the spatial goals is safeguarded after convolution.

2.6 Deep Residual Learning for Image Recognition

It is very tough to train deeper neural networks. Microsoft research presented a residual learning system to incorporate the training of networks that are deeper than the ones used already. The issue of disappearing/exploding gradients disrupt the assembly in extremely

deep neural networks. This issue, was to a great extent tended to by normalized initialization and intermediate normalization layers which improves networks with several layers to begin uniting for stochastic gradient descent with back-propagation. At the point when the networks with progressively neural network layers can begin meeting, a degradation issue has been uncovered; with the network profundity expanding, exactness arrives at a specific point and afterward begins to diminish quickly.

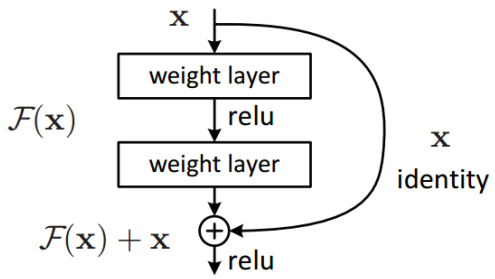


Figure 13. Working, Residual Learning

2.7 You Only Look Once: Unified, Real-Time Object Detection

2.7.1 Introduction

People take a look at an image and get what articles are in the image, where they are, and how they collaborate. People have a quick and precise visual framework, permitting us to carryout extreme tasks like driving with less consciousthoughts. Detection Systems at present repurpose classifiers to perform recognition. These frameworks take a classifier for an object and assess it at various areas and tests on an image to detect an object.

Figure 14 YOLO Detection system

The unified model has numerous advantages over conventional strategies for object recognition. To begin with, YOLO is truly quick. Since it outlines identification as a regression issue there is no requirement for an unpredictable pipeline. Second, YOLO reasons about the image when making forecasts all inclusive. Third, YOLO learns generalizable portrayals of articles.

2.7.2 Model

The different components of object detection are unified into a single neural network. The network utilizesthe featuresin an image to predict the bounding boxes. YOLO enables end-to-end training and real-time speeds while also maintaining high average precision. These confidence scores mean how confident the model is about the box containing an object and also how accurate it thinks the box is that it predicts. If there is no object in that bounding box, then the confidence scores will be zero, else the confidence score will be equal to the intersection over union (IOU) between the predicted box and the actual object-box.

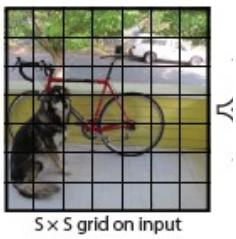


Figure 15. YOLO Model

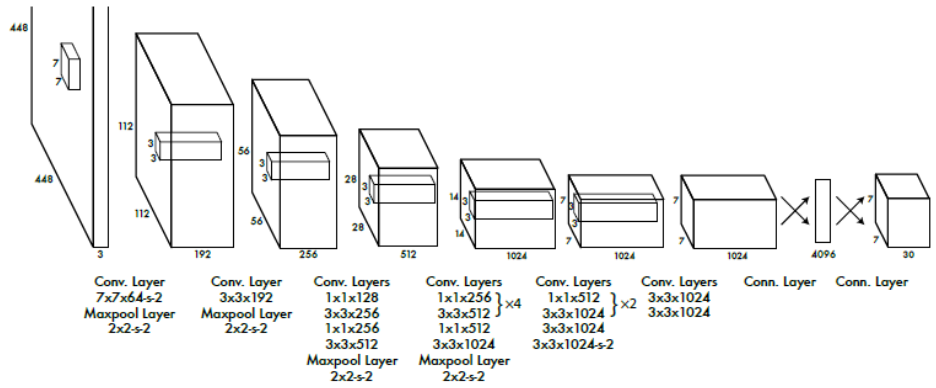


Figure 16. YOLO Architecture

The model is then converted to perform detection. Four convolutional layers and two fully connected layers with randomly initialized weights are added. The last layer is able to predicts the class probabilities as well as the bounding box coordinates.

2.7.3 Database

The model evaluated on the PASCAL VOC dataset. The convolutional layers are related on the ImageNet 1000-class rivalry dataset. For purpose of pre-training, the initial 20 convolutional layers are utilized trailed by an average-pooling layer.

2.7.4 Results

YOLO is exceptionally quick at test time because it just requires a single network evaluation, different from classifier-based strategies. Despite the fact that not basic to execution all things considered for R-CNN or DPM, non-maximal suppression includes 2-3% in mAP.

2.7.5 Conclusion

YOLO seems to be a unified model for object detection. The model is easy to build and can be trained on full images. Entire model is trained jointly unlike classifier-based approaches. Fast YOLO is the fastest object detector in the writing and YOLO

pushes the cutting edge real-time object location. Likewise, YOLO sums up new areas making it extraordinary for applications that depend on quick, vigorous object detetction.

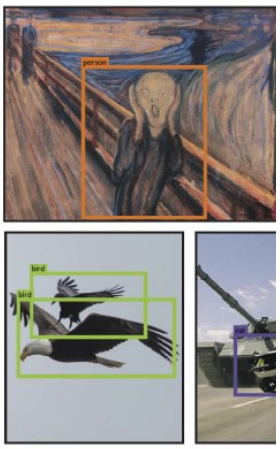


Figure 17. Qualitative Results

Chapter 3 SYSTEM DEVELOPMENT

3.1 Dataset

We have used MS-COCO Dataset as our primary dataset for the training purpose. The MS-COCO dataset consists of 82,000 images each having at least 5 captions. We made 4 datasets of different sizes by randomly choosing images from the clusters formed using the agglomerative clustering algorithm on 18000 images based on their features (more on Clustering and sampling in later sections).

3.2 Clustering

After reading various research papers, a common problem of overfitting arose. Also, all the previous models needed to be fed a lot of data hence resulted in more computation. We devised a method of constructing our dataset from the original dataset so that it covers all the variance present in our dataset.

Clustering is a technique in data mining and machine learning which helps to group similar data points such that the points in same group have similar behavior as compared to points in other groups. A group is called cluster.

Hierarchical clustering is of two types:

a. Agglomerative
b. Divisive

In agglomerative clustering, initially each point is considered as a cluster. At each iteration, the similar clusters merge with other cluster until a single cluster or X clusters are formed.

A proximity matrix is computed first of all. Every point is considered to be a cluster. Iteratively, two closest clusters are merged and the proximity matrix is updated. This process continues until a single cluster is obtained.

The hierarchical clustering is usually visualized using a dendrogram. A dendrogram is a tree-like diagram that records the sequences of merges or splits.

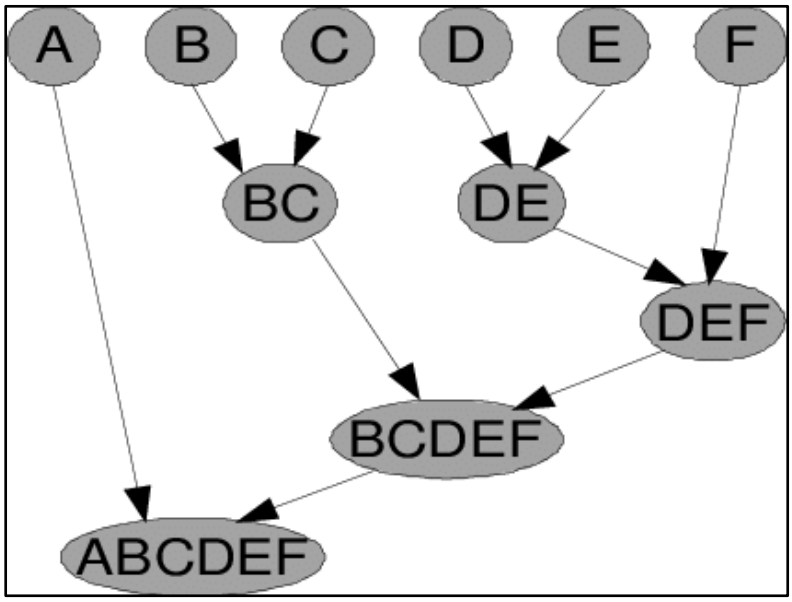
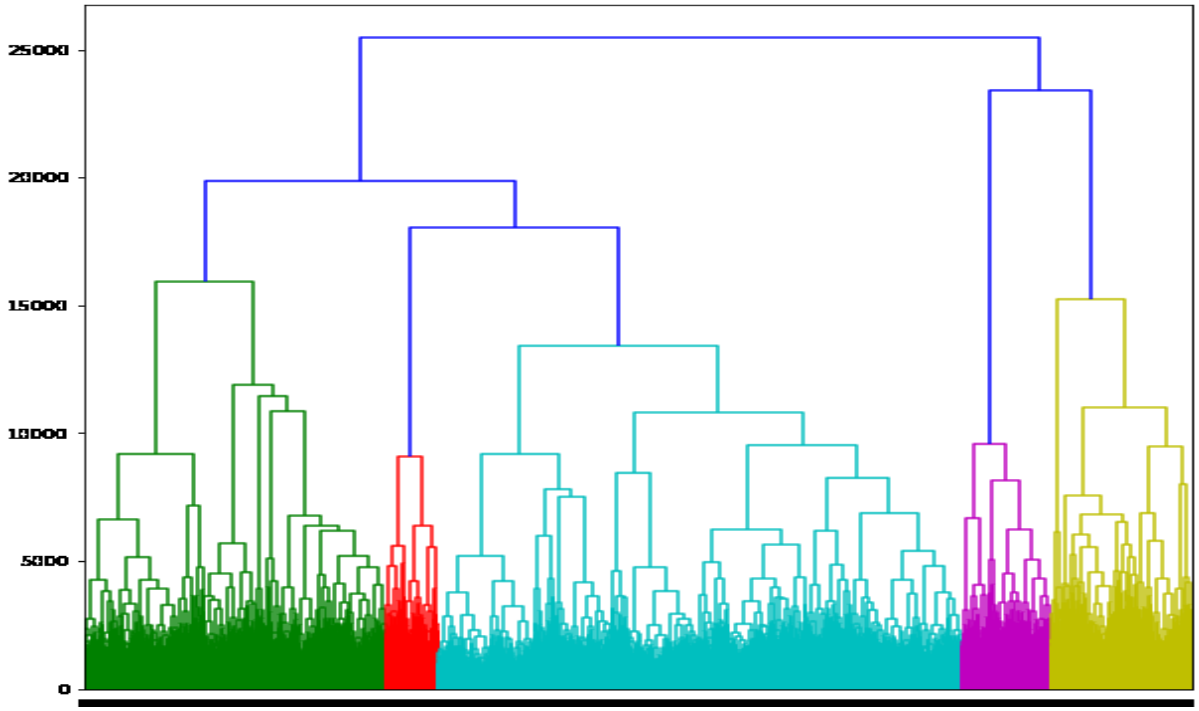


Figure 18. Agglomerative Hierarchical Clustering Technique

We perform Agglomerative Hierarchical clustering on the feature maps extracted from the images present in MS COCO dataset. We were able to obtain the following Dendrogram representing our 18K Dataset.



Graph 1: Dendrogram diagram for cluster hierarchy of the 18K Dataset based upon features from images is generated.

3.3 Sampling

The dendrogram obtained was analyzed and the number of clusters was chosen to be 16. The 16 clusters formed contained a variable number of images ranging from ~100 images to ~1000 images per cluster.

There are many ways in which one can sample images from the cluster. We decided to randomly sample them so that every image has an equal chance of being selected.

We created 4 more databases from the 18K database. The sizes of these databases were different based on what percentage of images were randomly chosen from each of the 16 cluster. Since every cluster had a variable number of images, therefore a percentage of images were chosen from every cluster. The percentages were 50, 60, 70 and 80. Models were trained separately on these different datasets as well as on the 18K dataset.

Also, the clusters of images made sure almost all scenarios are covered so that the model generalizes well but also does not use much time as well as data. This creates an intelligent dataset of images randomly sampled from every cluster. The reduction of datasets with clustering also led to a reduction in training time.

3.4 Data Preprocessing

We had two types of data, both images and textual. There are different ways to preprocess both of these data.

3.4.1 Images Processing

A model should be large enough to capture relations in the data along with specifics of the problem. Early layers of the CNN architecture capture high level relations between the different parts of the input image. Later layers catch data which is local to the image and quite certain features that helps settle on an ultimate choice; for the most part, the subtleties that can help separate between those ideal outputs. Theregore, if the perplexing idea of the issue is high the amounts of parameters and the proportion of data required is in like manner gigantic.

When working on domain specific issue, regularly the measure of information expected to assemble models of this size is deficient and exceptionally elusive. In any case, models trained on one assignment catch relations in the information type and can be effortlessly reused for various issues in a similar domain. This is what is called as transfer learning.

In transfer learning, we use a pre-trained model, which was initially trained on a large readily available dataset like the ImageNet dataset. Then try tolook for layers that output features that can be reused. We utilize the output of the last convolutional layer as input to train smaller network which requires a less parameters.

The advantage of extracting bottleneck features further include reduction in dimensionality as an image initially is represented by 150x150 with 3 channels which ultimately leads to approx. 3,375,000 floats (=150*150*159) whereas the VGG-16 outputs a 4x4x512 feature map which is 8,192. This is a great reduction in terms of dimensionality and also helps the model focus on the important features by cutting out the noise.

The figure below is the architecture of Inception V3 model. For extracting the bottleneck features the fully connected layers at the end of the model are not initialized. Output is given by the convolutional layers.

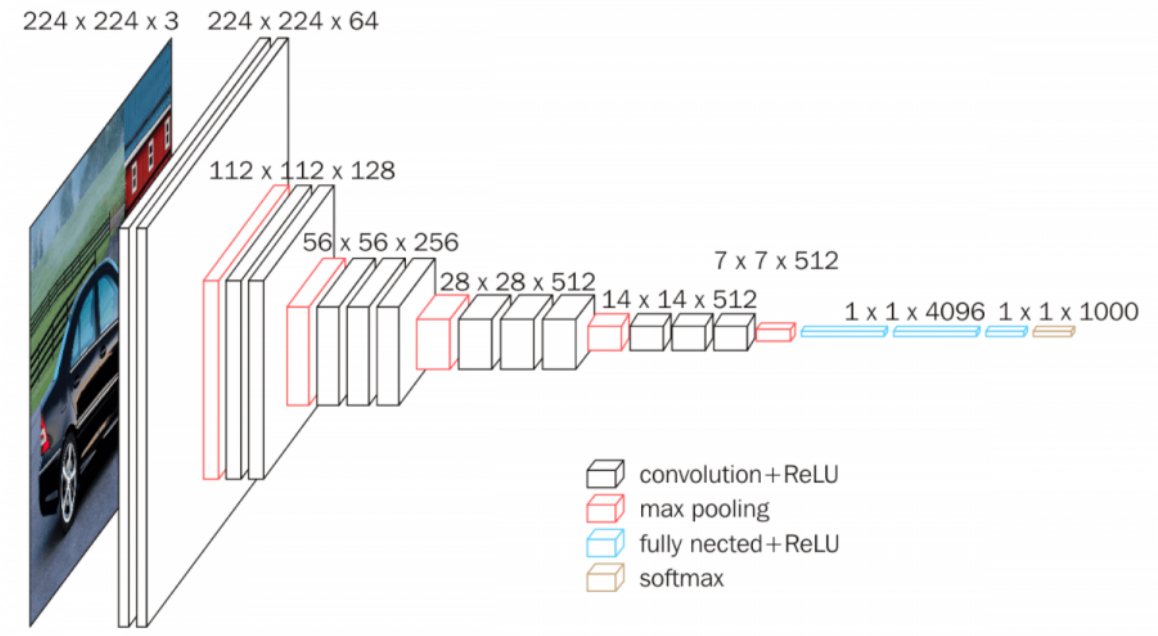


Figure 19: VGG-16 Architecture

The images in the dataset are represented by the feature maps we generated using the VGG-16 architecture.

3.4.2 Text Preprocessing

The captions are represented using a 200-dimensional GloVe word embedding vectors. Weights were initialized for the embedding layer as the embedding matrix generated from the vocabulary before training.

3.5 Model Details

3.5.1 Encoder

The encoder here is the VGG-16 model. The VGG-16 was initialized with the ImageNet weights. The images are fed to the VGG-16 model which then produces feature maps for those images from its second last layer as output. The feature maps produced from each image is of the shape (4x4x512).

Further, the feature maps and embedding vectors are passed through fully connected layers with dropouts of 50%. Batch normalization is also applied to the dense layer in the feature map arm of the model so as to reduce the training time and make it converge faster.

3.5.2 Decoder

In the caption embedding arm of the model, we trained the LSTM to output the caption word-by-word. The LSTM was initialized with 256 units.

In the figure given below we can see working of a single LSTM cell. It describes the cell state at any time *t*. *W* and *b* is the weight and bias of each gate in the cell. 'u' represents the update gate, 'o' represents the output gate, 'f" represents the forget gate. σ represents the sigmoid function. 'x' is the input to the cell at any time *t*. 'a' represents the hidden cell state at any time *t*. 'y' is the output of the cell.

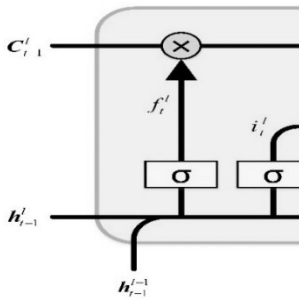


Figure 20. A LSTM cell

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}]b_c) \quad (1)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \quad (2)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \quad (3)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (4)$$

$$c^{<t>} = \Gamma_f * c^{<t-1>} + \tilde{c}^{<t>} * \Gamma_u \quad (5)$$

$$a^{<t>} = c^{<t>} * \Gamma_o \quad (6)$$

$$y^{<t>} = softmax(a^{<t>}) \quad (7)$$

An approach with stacked LSTMs was also analyzed. The intuition being that stacked LSTMs might be able to capture more complex patterns or relationships. The first LSTM

was followed by a dropout of 50% so as to overcome the overfitting it may cause and was followed by another LSTM.

3.6 Training

During the training phase, the model is trained to produce caption word-by-word until the maximum length of caption (which in this case was 51) is attained or the model outputs end of sentence tag (‘<end>’).

The model was trained for 40 epochs with a batch size of 16. The training was carried out on an online platform called Google Colab due to the non-availability of GPU-enabled computers. Colab is equipped with NVIDIA T4 GPUs. So, for every image I the model calculates the conditional probability of

$$p(W_t \mid I, W_0, W_1, W_2, \ldots, W_{t-1})$$

where *W* is the word predicted at any time *t*. Adam optimizer with categorical cross-entropy as the loss function was used to train the model.

3.7 Inference

After training the model, the process of decoding the outputs was carried out by two different algorithms namely the greedy search algorithm and beam search algorithm. The greedy search algorithm outputs the word with the highest conditional probability at every timestep*t*.

$$p_{greedy}^{<t>} = argmax(p(W_t \mid I, W_0, W_1, \ldots, W_{t-1})$$

The beam search algorithm considers *k* best possible sentences at any time *t* and in the end, it outputs the sentence with maximum probability out of the *k* sentences. The greedy search algorithm is basically a special case of the beam search having beam width k=1.

3.8 Hardware Requirements

1. OS- Windows/Ubuntu/Mac
2. RAM- 12 GB or higher
3. GPU- NVidia GeForce 1650 3 GB dedicated or higher
4. Processor- Quad-Core i5 or higher
5. 512 GB SSD or higher

These requirements are the minimum required for training this model in optimal time. Online cloud applications such as Google Colab can also be used as they have better hardware specifications like Tesla K80 GPU and better RAM capacity.

3.9 Software Requirements

1. Anaconda3
2. Python 3
3. Tensorflow
4. Keras
5. Matplotlib
6. Scikit Learn
7. Seaborn
8. Pandas
9. Numpy
10. Jupyter Notebooks

Chapter 4 PERFORMANCE ANALYSIS

4.1 DBSCAN vs Agglomerative Clustering

We applied two clustering algorithms on the same feature maps extracted from VGG-16 CNN architecture. Both algorithms work quite differently and hence gave very varying results for the same dataset of feature maps.

4.1.1 Agglomerative clustering

Agglomerative Clustering, which is a Hierarchical clustering algorithm is used primarily to build a hierarchyof clusters starting from one data point being considered as one cluster and then based on distance the clusters are paired till we get one giant cluster.

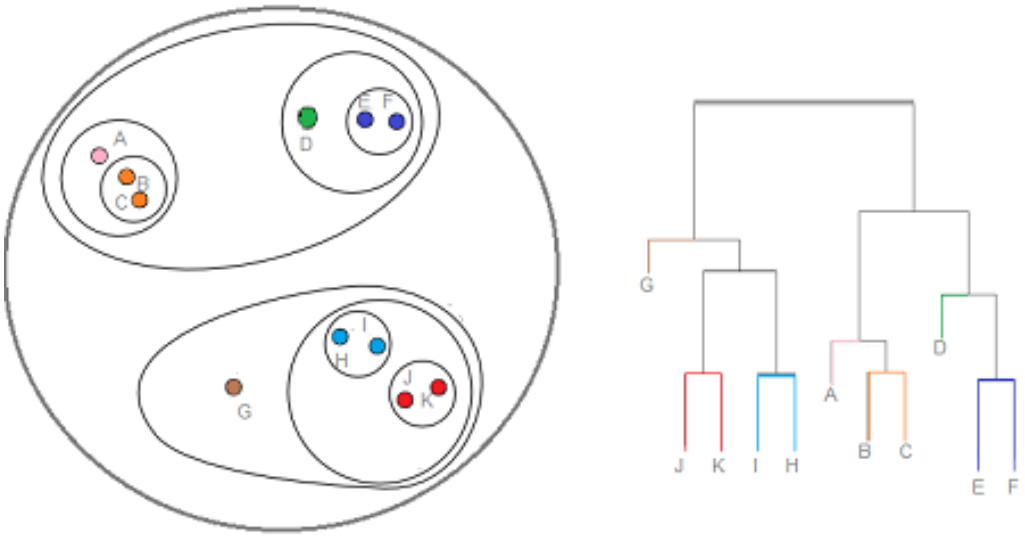
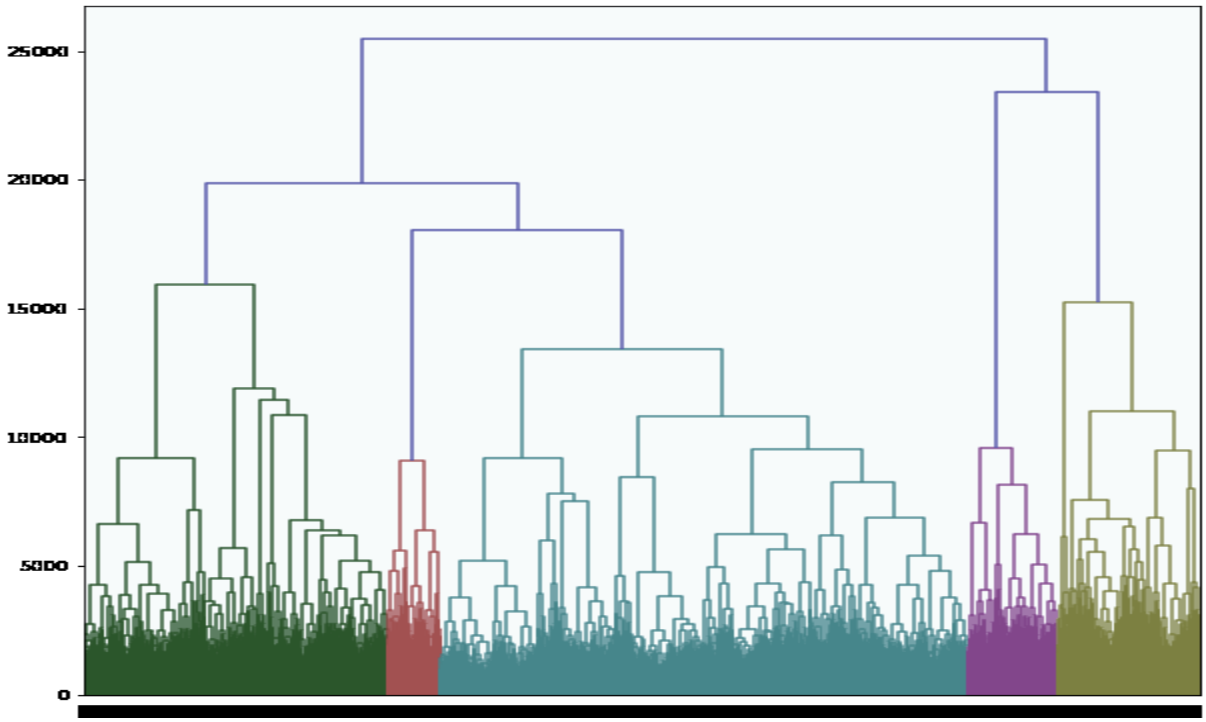


Figure 21: Agglomerative clustering and dendrogram

This clustering algorithm gives us the cluster tree, called the Dendrogram, which graphically shows the distribution of data points into clusters ranging from 1 to the number of data points. The dendrogram we attained from our dataset of image features is shown below.



Graph 2: Dendrogram for 18K dataset

Analysis of the Dendrogram resulted in us coming down to 16 as an optimal number of clusters in which our data points must be divided

4.1.2 Density Based Spatial Clustering of applications with noise (DBSCAN)

On the other hand, DBSCAN, which is a Density based clustering algorithm, clusters together the data samples which are close to each other based on a) distance between them and b) minimum number of points. The data samples which are in the low density region are simply marked as outliers. However smart it may seem, DBSCAN is not fully deterministic as it considers the far away and unreachable data points as outliers or noise. Coming back to our work, use of DBSCAN gave us a very few clusters for the given dataset and classifiedremaining data points as noise. Even on changing the

On further research we came down to the decision of going forth with Agglomerative clustering as our choice of clustering algorithmas it wouldfully serve our purpose since we needed a significant amount of clusters to build a good and lean dataset.

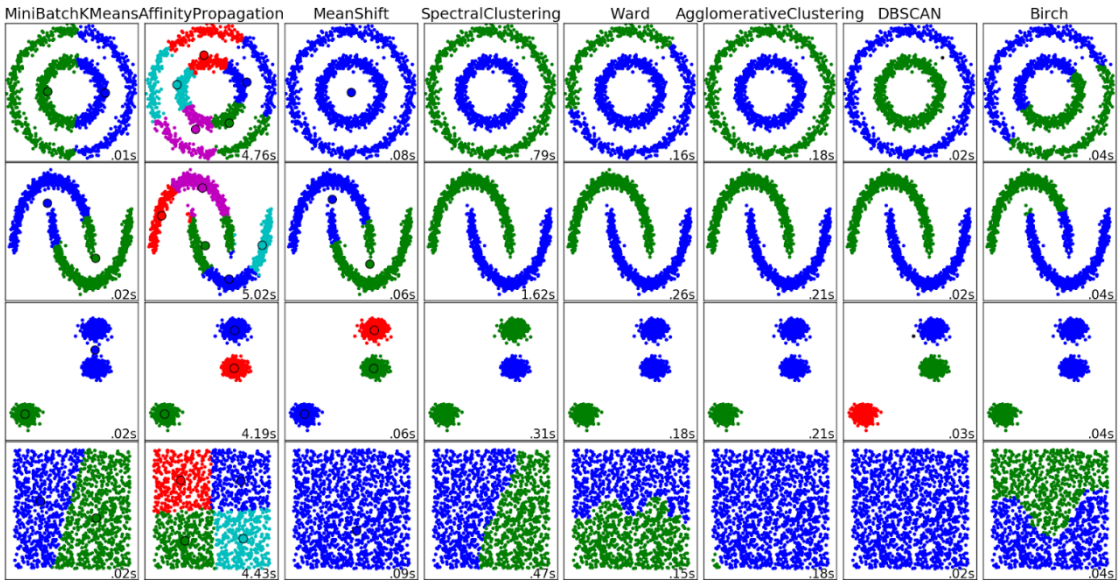


Figure 22: DBSCAN clustering example with different parameters

4.2 Evaluation Metrics

One of the most challenging task of any machine translation or image captioning model is to how to evaluate the outcomes. Traditional supervised learning evaluation metrics of precision, recall and f1 score do not work here.

We performed varied sets of experiments on different sizes of datasets, model architectures, and evaluated them on two evaluation metrics namely BLEU and METEOR and a dataset of 250 test images was used for the same. Our primary data source was the MS-COCO dataset and

4.2.1 BLEU

BLEU stands for Bilingual Evaluation Understudy. It uses n-gram precision to calculate an average score for the given hypothesis in comparison to the corresponding reference sentence. BLEU has been the benchmark metric for machine translation tasks. One of the limitations of BLEU has been that it does not understand meaning of the words used in the sentence. It simply compares the word's presence in the hypothesis with that in the reference sentence. This has a negative effect because if a synonym is used in place of a particular word present in reference then although the caption maybe correct but it will show difference in captions according to the BLEU score. This limitation was addressed by the METEOR metric.

4.2.2 METEOR

METEOR is another machine translation metric often used. It overcomes the limitations faced by BLEU. It calculates the harmonic mean of precision and recall of unigram matches between the reference and the hypothesis sentences. Also, it does not explicitly matches the words and rather takes into account synonym matching as well.

4.3 Single vs Stacked LSTMS

4.3.1 Single LSTM

Our results were generated on total of 5 datasets. We call the datasets created using random sampling fromfeature-based clusters as Intelligent datasets. These datasets are named like Intelli-x (short for Intelligent-x) where x is the percentage of data we sampled from each of the clusters. In the table, we present the results generated over the test dataset. The beam search algorithm gives better results than greedy algorithm most of the times. The beam width used for beam search was 3.

Dataset/Metric	BLEU		METEOR	
	Greedy	Beam	Greedy	Beam
Intelli-50	49.3	50.2	29.4	29.4
Intelli-60	48	49.1	29.2	30.7
Intelli-70	48.9	51	28.7	29.5
Intelli-80	48.5	49.3	28.3	27.4
18K	50.2	49.6	29.2	27.9

Table 4: BLEU-4 and METEOR Scores for the models trained on reduced version of MSCOCO dataset containing 18K data points and reduced version of this 18K dataset containing 50,60,70 and 80 percent of the 18K data points

The BLEU score peaks for Intelligent-70 dataset giving a score of 51 whereas the METEOR score peaks for the Intelligent-60 dataset giving a score of 30.7. So, we can safely say that reduced datasets as better as the actual datasets and in many cases better than the actual dataset. Almost every dataset gave better or equal results as the actual dataset. These scores are an improvement from previous works also we can state that even with 50 or 60 percentage of data we can train the model if the dataset is sampled in a better way. A generalized dataset will most likely train a more generalized model.

4.3.2 Stacked LSTMs

Models with stacked LSTMs were trained separately on two datasets, one being Intelligent-70 and the one being the 18K dataset.

Dataset/Metric	BLEU		METEOR	
	Greedy	Beam	Greedy	Beam
Intelli-70	48.9	47.9	28	27.2
18K	48.6	51.5	28.2	28.3

Table 5: BLEU-4 and METEOR scores comparison for the models with stacked LSTM trained on 18K and Intelligent-70 Datasets.

The BLEU and METEOR scores for the 18K dataset are better than the Intelligent-70 dataset. We can say that since the model complexity increases and the number of parameters to be trained also increases, therefore the model tends to under fit somewhat when trained on the intelligent dataset.

Chapter 5 CONCLUSIONS

5.1 Conclusions

We propose clustering based approach for Image captioning that gives satisfactory performance on the MS-COCO dataset. We show how limiting the data through resizing, clustering and sampling we can make our encoder-decoder model give more interpretability, and show that the output captions related very well to the human way of thinking especially when the model is trained upon just the right amount of data.

5.2 Future Work

5.2.1 Attention

One of the major limitations of our work is the use of an encode-decoder model. This is so because an encoder-decoder model is a little slow and it usually faces failure when the

input sequence is really long. To overcome this limitation, we aim at combining our encoder decoder model with Attention mechanism to achieve better results. What an Attention model does is, it tries to teach the decoder, where to pay attention in the richer encoding while predicting the sentence as the output.

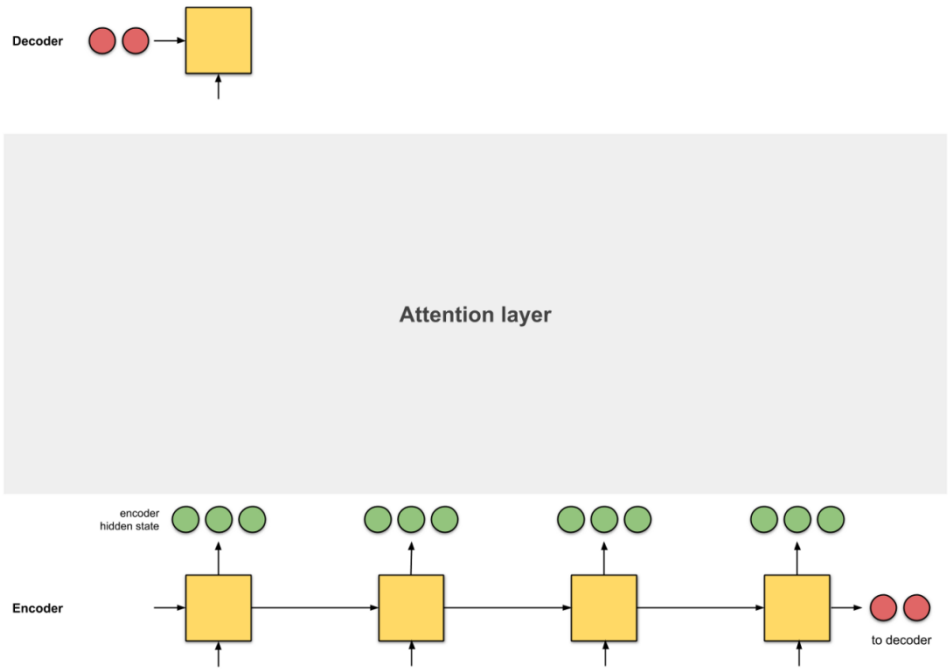


Figure23: Attention as an interface between encoder and decoder

5.2.2 User interface for Caption generation of the uploaded image

Next, we wish to work on a web application where a person can upload an image and get its caption instantly. The uploaded image will be passed through our trained model and will output a sentence describing the content of the image. Further, through human evaluation we can review the caption generated for the uploaded image and add the pair to our primary dataset. Then upon training the model with newly added data points we hope to achieve better results and scores through evaluation metrics.

5.3 Applications

User Interface to improve the model

The problem faced during evaluating the results is that whether the caption generated by RNN was actually wrong or was it just not present in the dataset. The caption generated

by RNN maybe logically correct but maybe labelled by the model as wrong because it was not present in the given dataset.

To overcome this anomaly, we have made an online platform where we host a dataset of ~10000 images. The website will allow people from all over the world to submit a description of the image as they deem fit for that image up to 140 characters. From the data collected from that website the captions generated by the model will be evaluated. Also, the most commonly occurring captions will be added to the corpus and the model will be re-trained for better generalization of the model.

We believe this is the way a model can be truly evaluated as an image can have many interpretations according to every human and we cannot ignore all of them.

Figure 24. Snapshots of website

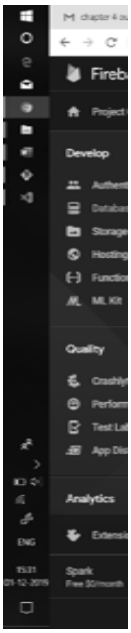


Figure 25. Snapshots database created for evaluation

References

Vinyals, O., Toshev, A., Bengio, S., &Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... &Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

You, Q., Jin, H., Wang, Z., Fang, C., & Luo, J. (2016). Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4651-4659)

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., &Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826)

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778)

Bahdanau, D., Cho, K., &Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*

Redmon, J., Divvala, S., Girshick, R., &Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788)

Appendices

Code:

Cluster file

```
import urllib
from IPython.display import Image, display, clear_output
from collections import Counter

import matplotlib.pyplot as plt
# import seaborn as sns
#get_ipython().magic('matplotlib inline')

import json
import pickle

from sklearn.metrics import classification_report, confusion_matrix
from PIL import Image
# sns.set_style('whitegrid')

import os
import h5py
import numpy as np
import pandas as pd
# from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator, array_to_img,
img_to_array, load_img
from keras.regularizers import l2, l1
from keras.models import Sequential, load_model
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D
from keras.layers import Activation, Dropout, Flatten, Dense
```

```
from keras.utils.np_utils import to_categorical
from keras import optimizers
from keras.callbacks import ModelCheckpoint, History
from keras.applications import InceptionV3
conv_base2 = InceptionV3(weights='imagenet',include_top=False)

from google.colab import drive
drive.mount('/content/drive')
# os.chdir("C:/Users/Nidhish/Desktop/project/Image Captioning/Data Set (MS-COCO)")

with open('/content/drive/My Drive/Colab Notebooks/encoded_train_images.pkl', 'rb') as
f:
tr = pickle.load(f)

with open('/content/drive/My Drive/Colab Notebooks/encoded_test_images.pkl', 'rb') as f:
tt = pickle.load(f)

with open('/content/drive/My Drive/Colab Notebooks/encoded_val_images.pkl', 'rb') as f:
vl = pickle.load(f)

ffd=dict()

ffd.update(tr)
ffd.update(tt)
ffd.update(vl)

train_features=list(ffd.values())
photos=list(ffd.keys())

from sklearn.cluster import AgglomerativeClustering
```

```
import scipy.cluster.hierarchy as sch

feature_list=[]
for x in range(len(train_features)):
features_np=np.array(train_features[x])
feature_list.append(features_np.flatten())

len(feature_list)

plt.figure(figsize=(10,10))
dendrogram = sch.dendrogram(sch.linkage(feature_list, method='ward'))

numberofc=16

model = AgglomerativeClustering(n_clusters=numberofc, affinity='euclidean',
linkage='ward')
model.fit(feature_list)
labels = model.labels_

kgf=labels[0:]
spd=feature_list[0:]

dct=dict()
for cn in range(len(kgf)):
dct[photos[cn]]=kgf[cn]

# dct

sd=dict()
for j,h in dct.items():
```

```
  if h in sd.keys():
sd[h].append(j)
  else:
lis=list()
lis.append(j)
sd[h]= lis

import collections
od = collections.OrderedDict(sorted(sd.items()))
od=dict(od)
for i,j in od.items():
  # print("Cluster number: ",i+1,"\nImages: ",j)
  print(i,j)

# od

with open('od.pkl', 'wb') as handle:
pickle.dump(od, handle, protocol=pickle.HIGHEST_PROTOCOL)

from google.colab import files
files.download('od.pkl')




newpath=train_dir+'An/'
namesofim=[]
for f in os.listdir(newpath):
```

```
namesofim.append(newpath+f)


import shutil
for cn in range(len(kgf)):
clpath=train_dir+str(kgf[cn])
if(not os.path.exists(train_dir+str(kgf[cn]))):
os.mkdir(train_dir+str(kgf[cn]))
shutil.copy(namesofim[cn],clpath)
```

Encoder file

```
import os
import h5py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image


from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator, array_to_img,
img_to_array, load_img
from keras.regularizers import l2, l1
from keras.models import Sequential, load_model
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
from keras import optimizers
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import Decoder
import sys
```

```
import warnings

if not sys.warnoptions:
warnings.simplefilter("ignore")

print("Initialising Encoder")
conv_base = VGG16(weights='imagenet',
include_top=False,
input_shape=(150, 150, 3))
print("Encoder Initialised")

class Encoder:
        path="
        conv_base=0

        def encode(self,image):
            image=self.preprocess(image)
        feat_vect=self.conv_base.predict(image)
        feat_vect=np.reshape(feat_vect,4*4*512)
            return feat_vect

        def preprocess(self,image_path):
        img=image.load_img(image_path,target_size=(150,150))
            x=image.img_to_array(img)
            x=np.expand_dims(x,axis=0)
            x=preprocess_input(x)
            return x

        def __init__(self,path):
```

```
print("Encoding Phase*******************************")
self.path=path
self.conv_base=conv_base
feature_map={}
feature_map[path]=self.encode(path)
print("Picture Encoded*******************************")
print("Decoding Phase*******************************")
Decoder.Decoder(feature_map)
```

Decoder file

```
import os
import h5py
import json
import numpy as np
import pandas as pd
from numpy import array
from PIL import Image
import matplotlib.pyplot as plt
import pickle
from pickle import load,dump
import keras
from keras.layers import Input, Embedding, LSTM, Dense, BatchNormalization,Dropout
from keras.models import Model
from keras.preprocessing.sequence import pad_sequences
import sys
import warnings

if not sys.warnoptions:
```

```
warnings.simplefilter("ignore")

print("Initialising Decoder")
max_length=51
ixtoword=load(open('data/vocabulary/ixtoword.pkl','rb'))
wordtoix=load(open('data/vocabulary/wordtoix.pkl','rb'))
embeddings_index={}
f=open('data/embeddings/glove.6B.200d.txt',encoding="utf-8")
for line in f:
        values=line.split()
        word=values[0]
        coefs=np.asarray(values[1:],dtype='float32')
        embeddings_index[word]=coefs
f.close()
embedding_dim=200
vocab_size=len(ixtoword)
embedding_matrix=np.zeros((vocab_size,embedding_dim))

for word, i in ixtoword.items():
        embedding_vector=embeddings_index.get(word)
        if embedding_vector is not None:
                embedding_matrix[i]=embedding_vector
print('Decoder Initialised!')

class Decoder:
        model=0

        def __init__(self,feature):
                self.createModel()
```

```
        print("Model Initialised")
        pic=list(feature.keys())[0]
        image=feature[pic].reshape(1,8192)
        print("Caption Generated:",self.generateCaption(self.model,image))



defcreateModel(self):
        inputs1=Input(shape=(4*4*512,))
        fe1=Dropout(0.25)(inputs1)
        fe2=Dense(256,activation='relu')(fe1)
        bn=BatchNormalization()(fe2)

        inputs2=Input(shape=(max_length,))
        se1=Embedding(vocab_size,embedding_dim,mask_zero=True)(inputs2)
        se2=Dropout(0.25)(se1)
        se3=LSTM(256,return_sequences=True)(se2)
        se4=LSTM(256)(se3)

        decoder1=keras.layers.add([bn,se4])
        decoder2=Dense(256,activation='relu')(decoder1)
        outputs=Dense(vocab_size,activation='softmax')(decoder2)

        self.model=Model(inputs=[inputs1,inputs2],outputs=outputs)

        self.model.layers[2].set_weights([embedding_matrix])
        self.model.layers[2].trainable=False

        self.model.compile(loss='categorical_crossentropy',optimizer='adam')
```

```
                self.model.load_weights('data/model/model_weights_70_stacked.h5')


        defgenerateCaption(self,model,photo):
                in_text='<start>'
                for i in range(max_length):
                        sequence=[ixtoword[w.lower()] for w in in_text.split() if w.lower()
in ixtoword]
                        sequence=pad_sequences([sequence],maxlen=max_length)
                        yhat=model.predict([photo,sequence],verbose=0)
                        yhat=np.argmax(yhat)
                        # print('yhat:{}'.format(yhat))
                        word=wordtoix[yhat]
                        # print(word)
                        in_text+=' '+word
                        if word=='<end>':
                          break;
                        final=in_text.split()
                        final=final[1:-1]
                        final=' '.join(final)
                return final
```

Input file

```
import os
from PIL import Image
from tkinter import *
from tkinter import filedialog
import Encoder
import cv2
import sys
```

```
import warnings

if not sys.warnoptions:
warnings.simplefilter("ignore")

class Input:

        def browse(self):
                root = Tk()
                root.withdraw()

                tempdir = filedialog.askopenfilename(filetypes = (("Image", "*.jpg"), ("All
files", "*")))

                if(len(tempdir)==0):
                        print("wrong input")

                else:
                        Encoder.Encoder(tempdir)

        def camera(self):
                cam=cv2.VideoCapture(0)
                cv2.namedWindow("Capture Image")
                img_counter=0
                img_name=[]
                currdir=os.getcwd()
                while(True):
                ret,frame=cam.read()
                   cv2.imshow("Capture Image",frame)
                   if not ret:
```

```
                break;
            k=cv2.waitKey(1)
            if(k%256)==27:
        print("Escape hit, closing...")
                break;
        elif (k%256==32):
        img_name.append("opencv_frame_{}.png".format(img_counter))
                cv2.imwrite(img_name[img_counter],frame)
        print("{} written!".format(img_name))
        img_counter+=1

        cam.release()
        cv2.destroyAllWindows()
        Encoder.Encoder(currdir+'\\'+img_name[img_counter-1])


    def __init__(self,i):
            if(i==1):
                    self.browse()
            elif(i==2):
                    self.camera()
            else:
                    print('Wrong input')



if __name__=='__main__':
        print("Welcome to Image Caption Generator")
        while(True):
```

```
        print("Enter 1 to Browse Image file")
        print("Enter 2 to capture image from camera")
        ipt=int(input("Enter:"))
        if(ipt==0):
                break
        i=Input(ipt)
print("Bye")
```

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……14-07-2020……..

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report ✓ | Paper

Name: Nidhish Manchanda **Department:** Information Technology **Enrolment No** 161459

Contact No. 7011163363 **E-mail.** nidhish.manchanda@gmail.com

Name of the Supervisor: Mr. Rizwan Ur Rehman

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): IMAGE CAPTIONING USING NEURAL NETWORKS

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 30
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 14

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……11……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com