Report for "Text Generation from Image"

By

Kruti Thukral

012586041

Individual Project

I. Introduction

Vision is an important sense for human perception and language is essential for communication between humans. With advancement in technology and robotics, building a system that can concurrently process visual stimuli and describe them in natural language would be a powerful tool to have [1]. There has been substantial research in the field of image captioning and many models have been proposed to solve the problem. As part of this project, we will explore a joint model "NIC" [7] which leverages vision CNN and language generating RNN for image captioning.

II. Motivation

Automatically describing the content of an image using properly formed sentences is a challenging task, but it has the potential to have a great impact [7]. Such a system can be used in chatty robots in wide application areas.

A. Aid to the blind and disabled

Image captioning can help visually impaired people better understand the content of images on the web. In the futuristic world, such a system can assist blind and disabled people with their day-to-day activities. They can act as the primary caregiver in situations wherein human caregiver is not available. Thus the building of such an intelligent system has substantial positive social impact.

B. Efficient google image search

Automatic image captioning can generate textual data from images that can be used for google search. The google search results will be able to produce a rich dataset of text and images.

C. Automatic captioning of frames from CCTV footage

Surveillance systems that can automatically identify suspicious behavior can send timely notifications of potential threat situations. This will help in reducing the crime rate.

D. Self-driving cars

Systems can be built that can infer from their surroundings. Automatic inference is a crucial task for self-driving cars.

III.   Related work

With advancements in AI and machine learning, a potential solution for image captioning can be implemented using deep learning. There are various alternatives to text generation. For example, an image can be described by a single high-level sentence [4, 5, 6, 7]. With such an approach, there is a restriction on the conveyed information. One recent alternative to single sentence generation is dense captioning which identifies regions of an image and then describes each with a short phrase [8]. This approach has the potential to convey more information however content need not be necessarily coherent. To overcome this limitation, a hierarchical model has been suggested that can leverage the compositional structure of both images and language [1]. Such systems to generate natural language based on an image consist of two main modules: Image Processing Module and Text Generator Module. The basic building blocks are a combination of CNNs and RNNs. CNN is used to extract image features and an RNN is used to describe image into a sentence. The model implemented in this project is based on the "NIC" model [7].

IV.   Datasets

We will need a corpus of images to train the model. Some of the datasets that could be used are as below:
- A. Microsoft COCO
  Microsoft COCO is a large-scale dataset for object detection and captioning [2]. This dataset contains 180k images.
- B. Visual Genome
  Visual Genome is a dataset to connect well-defined image concepts to language [3].
- C. Flickr8k
  This dataset contains 8k images with 5 captions for each image.
- D. Flickr30K
  This dataset contains 30k images with 5 captions for each image.

Due to constraints with regards to available processing power and storage, I decided to go ahead with the Flickr8k dataset. The dataset was divided into a training set of 6000 images, and a test set of 1000 images.

V.   Data EDA

I carried out some exploratory data analysis. Some findings from EDA are as following
- A. The captions contain both upper-case and lower-case words

B. The captions contain special tokens such as '%', '$', '#' etc
C. The captions contain alpha-numeric words
D. Plotted graphs to find most and least frequently occurring words

```
[106] # plot words with the least frequency

      least_common = FreqDist(dict(freq_dist.most_common()[-30:]))
      least_common.plot()
```
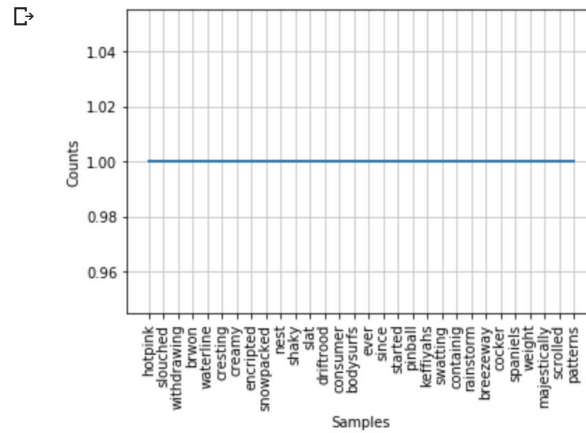


```
[104] # plot words with the most frequency
      freq_dist.plot(30)
```
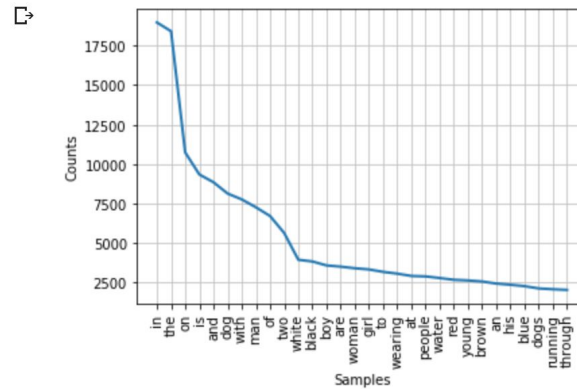

Fig 1: Words with most and least frequency

VI. Data Preprocessing

A. Preprocessing for captions
1. As part of data preprocessing for captions, I lower-cased all the words (otherwise"hello" and "Hello" will be regarded as two separate words)
2. Removed special tokens such as '%', '$', '#' etc
3. Eliminated words which contained numbers such as 'hey199', etc

4. Created a vocabulary of unique words from the dataset
5. Created a dictionary to represent every unique word in the vocabulary by an integer (index)
6. The words which occur very less does not carry much information. Thus, considered words with a frequency of more than 10
7. Calculated the maximum length of a caption
8. Added "startseq" and "endseq" to every caption
   B. Preprocessing for images
1. Converted all the images to size 299x299 as expected by the inception v3 model

VII. Architecture
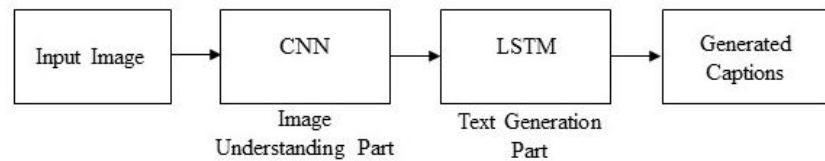
Fig 2: Architecture Diagram

A. The architecture of the image captioning consists of a CNN and an RNN. The convolution neural network encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence.
B. Global image features are extracted from the hidden activations of CNN. Our purpose here is to get a fixed-length informative vector for each image. Steps carried out for image feature vector extraction are as following
   1. Leverage transfer learning by using InceptionV3 model (Convolutional Neural Network) created by Google Research. This model is trained on the Imagenet dataset to perform image classification on 1000 different classes of images.
   2. Remove the last softmax layer from the model
   3. Pass every training and test image to the CNN model (inception v3) to get the corresponding 2048 length feature vector

Fig 3: Image Feature extraction using transfer learning
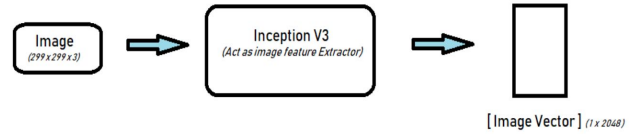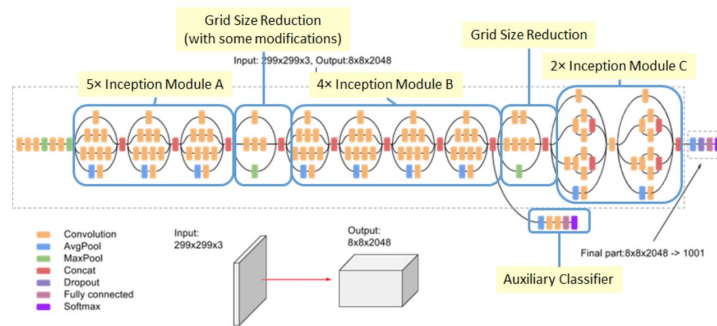

Fig 4: InceptionV3 architecture

    4. Store the feature vector in a pickle file

C. Image features are then fed into an LSTM to generate a sequence of words. Steps for generating a sequence of words are as following

    1. Compute data matrix for image and caption

	Xi		Yi	
i	Image feature vector	Partial Caption	Target word	
1	Image_1	startseq	the	data points corresponding to image 1 and its caption
2	Image_1	startseq the	black	
3	Image_1	startseq the black	cat	
4	Image_1	startseq the black cat	sat	
5	Image_1	startseq the black cat sat	on	
6	Image_1	startseq the black cat sat on	grass	
7	Image_1	startseq the black cat sat on grass	endseq	
8	Image_2	startseq	the	data points corresponding to image 2 and its caption
9	Image_2	startseq the	white	
10	Image_2	startseq the white	cat	
11	Image_2	startseq the white cat	is	
12	Image_2	startseq the white cat is	walking	
13	Image_2	startseq the white cat is walking	on	
14	Image_2	startseq the white cat is walking on	road	
15	Image_2	startseq the white cat is walking on road	endseq	

Data Matrix for both the images and captions

Fig 5: Data Matrix for image and captions

| i | Xi | | Yi |
	Image feature vector	Partial Caption	Target word
1	Image_1	[9]	10
2	Image_1	[9, 10]	1
3	Image_1	[9, 10, 1]	2
4	Image_1	[9, 10, 1, 2]	8
5	Image_1	[9, 10, 1, 2, 8]	6
6	Image_1	[9, 10, 1, 2, 8, 6]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4]	3
8	Image_2	[9]	10
9	Image_2	[9, 10]	12
10	Image_2	[9, 10, 12]	2
11	Image_2	[9, 10, 12, 2]	5
12	Image_2	[9, 10, 12, 2, 5]	11
13	Image_2	[9, 10, 12, 2, 5, 11]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7]	3

Data matrix after replacing the words by their indices

Fig 6: Data Matrix for image and captions using indices

2. Map every word (index) to a 200-long vector using a pre-trained GLOVE Model
3. Create a merge model for the two inputs, image vector and captions
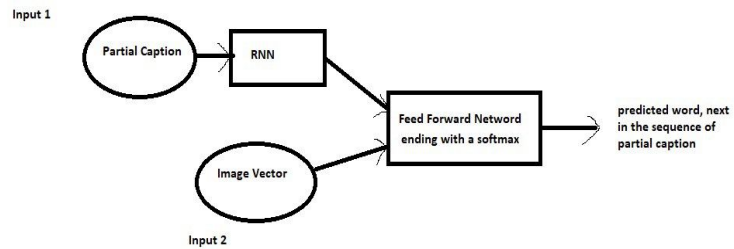
Fig 7: Merge model for image vector and captions

VIII. Model summary and hyper-parameters
A. Used Inception V3 for CNN
B. Used multi-layered LSTM for RNN
C. Used pre-trained GLOVE model for word embeddings

```
model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 34)	0	
input_3 (InputLayer)	(None, 2048)	0	
embedding_2 (Embedding)	(None, 34, 200)	330400	input_4[0][0]
dropout_3 (Dropout)	(None, 2048)	0	input_3[0][0]
dropout_4 (Dropout)	(None, 34, 200)	0	embedding_2[0][0]
dense_2 (Dense)	(None, 256)	524544	dropout_3[0][0]
lstm_2 (LSTM)	(None, 256)	467968	dropout_4[0][0]
add_2 (Add)	(None, 256)	0	dense_2[0][0] lstm_2[0][0]
dense_3 (Dense)	(None, 256)	65792	add_2[0][0]
dense_4 (Dense)	(None, 1652)	424564	dense_3[0][0]

```
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
```

Summary of the parameters in the model

Fig 8: Model Summary
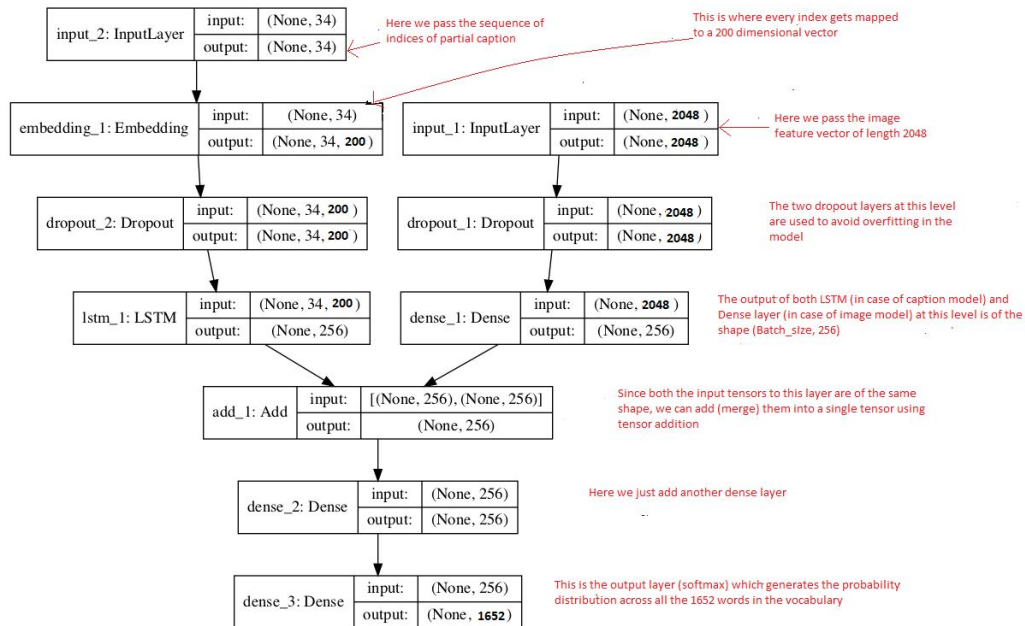
Fig 10: Model Visualization

D. Values for hyper-parameters are as following

Hyperparameter	Value

Learning rate	0.0001
Epochs	30
Batch size	6
Dropout rate	0.5
Embedding size	200
LSTM output size	1652
Optimizer	adam
Loss computation	categorical_crossentropy

IX. Inference methodology

A. Sampling

1. Iteratively generate caption one word at a time
2. **Greedily** select the word with the maximum probability
3. Stop the iterations on receiving an '**endseq**' token which means the model thinks that this is the end of the caption or when a maximum **threshold** of the number of words generated by the model is reached.

B. Beam Search

1. Iteratively consider the set of the k best sentences up to time t as candidates to generate sentences of size t + 1
2. keep only the resulting best k of them

C. **Sampling** inference method has been used in the project to predict the caption for the test image. Example of sampling inference method [9] is as following

Test image

Caption: the black cat is walking on grass
Also, the vocabulary in this particular example is
vocab: {black, cat, endseq, grass, is, on, road, sat, startseq, the, walking, white}.
We will generate the caption iteratively, one word at a time. In each iteration, the model will generate a 12-long vector(in the sample example while a 1652-long vector in the original example). The vector is a probability distribution across all the words in the vocabulary. We will greedily select the word with the maximum probability.

Iteration 1:
Input: Image vector + "startseq" (as partial caption)
Expected Output word: "the"

Input Caption: "startseq"

partial caption → RNN

image vector

Feed forward + Softmax

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

This probability must be maximum.

Predicted word **"the"**

Resulting caption after iteration **1:**
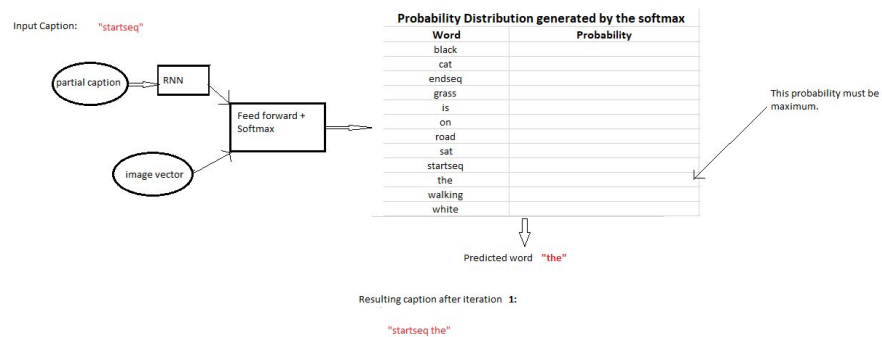
"startseq the"

Iteration 2:

Input: Image vector + "startseq the"

Expected Output word: "black

Input Caption: "startseq the"

partial caption → RNN

image vector

Feed forward + Softmax

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

This probability must be maximum.

Predicted word **"black"**

Resulting caption after iteration **2:**

"startseq the black"

Iteration 3:

Input: Image vector + "startseq the black"

Expected Output word: "cat"

Input Caption: "startseq the black"

Probability Distribution generated by the softmax	
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

This probability must be maximum.

Predicted word "cat"

Resulting caption after iteration 3:

"startseq the black cat"

Iteration 4:

Input: Image vector + "startseq the black cat"

Expected Output word: "is"

Input Caption: "startseq the black cat"

Probability Distribution generated by the softmax	
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

This probability must be maximum.

Predicted word "is"

Resulting caption after iteration 4:

"startseq the black cat is"
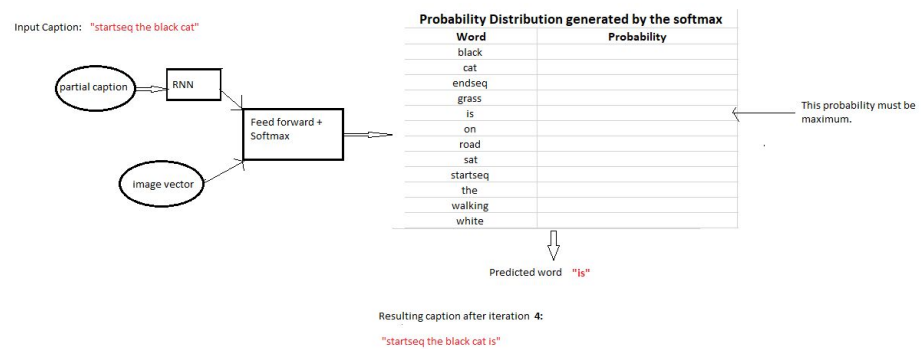
Iteration 5:

Input: Image vector + "startseq the black cat is"

Expected Output word: "walking"

Input Caption: "startseq the black cat is"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

partial caption → RNN

Feed forward + Softmax

image vector

This probability must be maximum.

Predicted word   **"walking"**

Resulting caption after iteration **5**:

"startseq the black cat is walking"
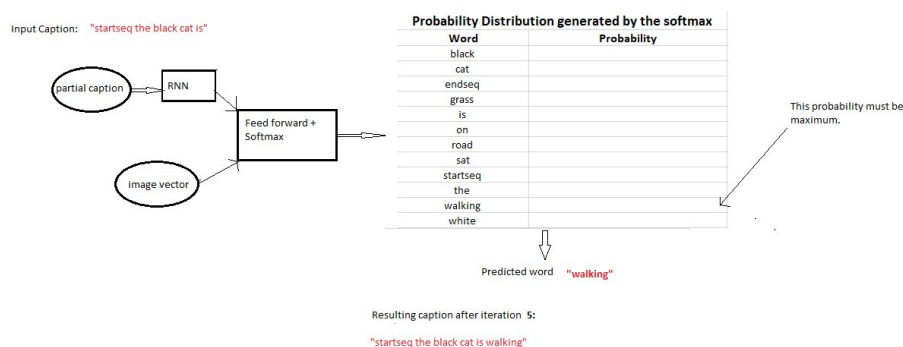
Iteration 6:

Input: Image vector + "startseq the black cat is walking"

Expected Output word: "on"

Input Caption: "startseq the black cat is walking"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

partial caption → RNN

Feed forward + Softmax

image vector

This probability must be maximum.

Predicted word   **"on"**
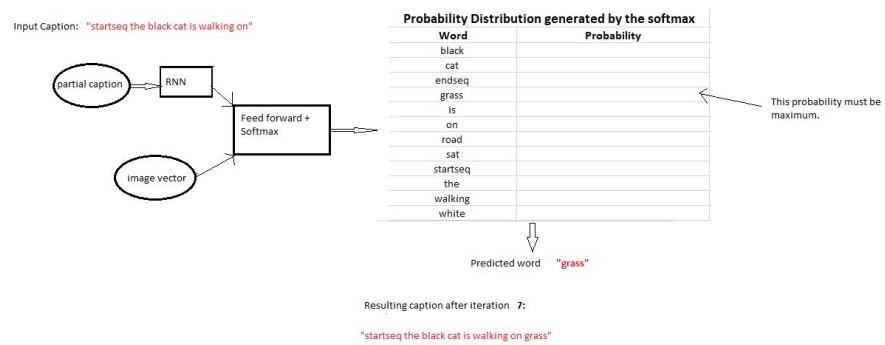
Resulting caption after iteration **6**:

"startseq the black cat is walking on"

Iteration 7:

Input: Image vector + "startseq the black cat is walking on"

Expected Output word: "grass"

Input Caption: "startseq the black cat is walking on"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

partial caption → RNN

image vector

Feed forward + Softmax

This probability must be maximum.

Predicted word   "grass"
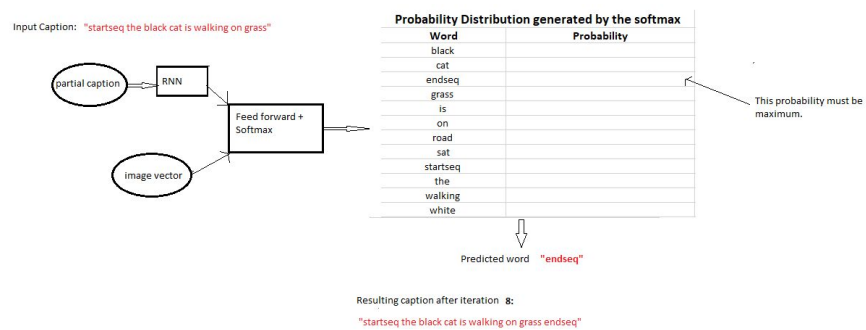
Resulting caption after iteration   7:

"startseq the black cat is walking on grass"

Iteration 8:

Input: Image vector + "startseq the black cat is walking on grass"

Expected Output word: "endseq"

Input Caption: "startseq the black cat is walking on grass"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

partial caption → RNN

image vector

Feed forward + Softmax

This probability must be maximum.

Predicted word   "endseq"

Resulting caption after iteration   8:

"startseq the black cat is walking on grass endseq"
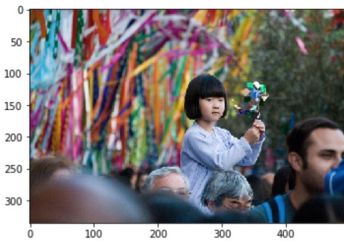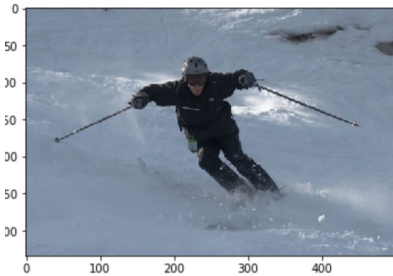
X.   Examples of caption generation by the proposed model

Index:899 Greedy:girl smiles while swinging on swing set

Index:677 Greedy:two girls are playing in the garden

dex:466 Greedy:man skiing down snowy mountain

dex:400 Greedy:man stands on top of mountain

dex:500 Greedy:two football players attempt to get the ball

ndex:1 Greedy:two dogs are running through the gra

ndex:0 Greedy:dog runs through the snow

Index:3 Greedy:man stands on top of cliff overlooking mountains

XI.   Evaluation Metrics

We will use BLEU (Bilingual Evaluation Understudy) metrics to evaluate model performance. Evaluation using BLEU has below advantages

A. It is easy to understand and is widely adopted
B. It is language independent

C. It is quick and inexpensive to compute

D. It aligns with human evaluation

Cumulative BLEU calculate individual n-gram scores at all orders from 1 to n and weight them by calculating the weighted geometric mean. Evaluation using cumulative BLEU metrics is as following

Metrics	Score
Cumulative BLEU-1	**0.438373**
Cumulative BLEU-2	**0.251009**
Cumulative BLEU-3	**0.171954**
Cumulative BLEU-4	**0.079744**

XII. References

[1] Jonathan Krause, Justin Johnson, Ranjay Krishna, Li Fei-Fei:
A Hierarchical Approach for Generating Descriptive Image Paragraphs,
In IEEE CVPR conference, 2017

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan,
P. Doll´ar, and C. L. Zitnick. Microsoft coco: Common
objects in context. In ECCV, 2014.

[3] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz,
S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein,
and L. Fei-Fei. Visual genome: Connecting language and
vision using crowdsourced dense image annotations. arXiv
preprint arXiv:1602.07332, 2016.

[4] X. Chen and C. Lawrence Zitnick. Mind's eye: A recurrent
visual representation for image caption generation. In CVPR,
2015.

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama,
M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell.
Long-term recurrent convolutional networks for visual
recognition and description. In CVPR, 2015.

[6] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments
for generating image descriptions. In CVPR, 2015.

[7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and
tell: A neural image caption generator. In CVPR, 2015.

[8] J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In CVPR, 2016.

[9] https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

XIII. Project Links

A. Youtube link for individual project presentation and project details

https://www.youtube.com/watch?v=y9f2BcCA-Xo&feature=youtu.be

B. Github link for project implementation

https://github.com/kruti-thukral/image_captioning

C. Dataset can be downloaded from

https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

Datasets that need to be downloaded from the above link are as follows

Flickr8k_Dataset.zip

Flickr8k_text.zip

D. Pre-trained Glove embeddings can be downloaded from

https://nlp.stanford.edu/projects/glove/

glove.6B.zip from above link was used in the project