FINAL PROJECT REPORT

Applied Machine Learning and Data Science 2020

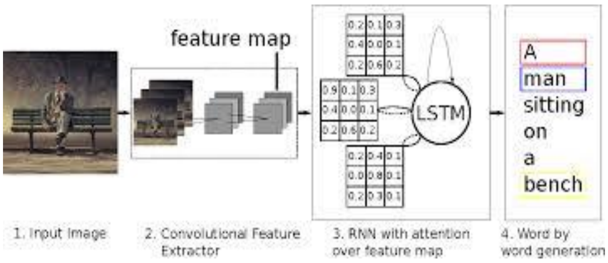
Project: Neural Image Caption Generator

Team: Savvy Captioners

Members: R K P GHANASHYAM, BALA MURUGAN, SHARAN NAGARAJAN

Abstract

Writing captions of images has always required the help of human’s knowledge in order to write catchy captions. Here in this project we have implemented the task of using a deep learning model, to automatically generate captions for images using visual attention mechanism. We have also shown visually how the model automatically learns to focus on specific parts of images, while predicting the corresponding words of the output sequence accurately. We have used two benchmark datasets – Flickr8k, MS COCO for our model



Introduction

Mimicking Human’s in general for most of the tasks is the primary focus of artificial intelligence. So, in NLP most of the tasks like summarising a text, generating sentences or even paragraph given the first word, and also machine translation in the past have achieved state-of-the-art performances, which inspires us to use NLP to generate captions given an image as the input. This would also integrate the two important neural network architectures in deep-learning (i.e.) CNN + RNN.

Our model is an implementation of the [1] Xu. Et al 2015, paper titled “Show Attend and tell”, which uses an encoder-decoder architecture, where the encoder encodes features from an image using transfer learning using a pre-trained CNN on IMAGENET dataset for classification. For our purpose we have removed the last layer which uses SoftMax for classification.

For decoder we have used GRU (Gated Recurrent Unit) instead of LSTM (Long Short-Term Memory) which the paper has used. We have also implemented the attention mechanism which [3] Badahnau et. Al has explained in his paper.

We have also implemented bleu score as an evaluation metric while testing of the images.

Additionally, we have taken screenshots of captions generated, using our self-made datasets of 10 images from the internet, which is a requirement of this project.

Related Work

Particularly for this project, we have referred papers [1] Show Attend and Tell, [2] Show and Tell. We also have referred to blogs and other websites for the literature and knowledge that we needed to complete this project successfully. These blogs and websites have been provided for further clarification in the References section of this paper.

Dataset

Flickr - 8k	This dataset has around 8k images with 5 captions given for each image	Since it has very few images, we have used (Caption_1, Image), (Caption_2, Image) and so on... for all the 5 captions making the dataset comprise around 40k image-caption pair. From this, 32000 images have been used for training and 8000 for validation, with the split ratio 0.2
MS-COCO	This dataset has around 120k images with 5 captions for image	Because of our GPU configuration, we kept our training images to be 25,600 and 6000 images for validation

General Architecture

1. Encoder

In this project for the purpose of encoder, we have used a Convolutional Neural Network, Inception v3, which is pre-trained on IMAGENET dataset for classification. For our purpose where we need to get the features encoded, we remove the last layer which is used for classification purpose.

ImageNet Dataset

ImageNet, is a dataset of over 15 million labelled high-resolution images with around 22,000 categories.

2. Decoder

The aim of the model is to maximize the probability of the correct description given an image by using the following equation.

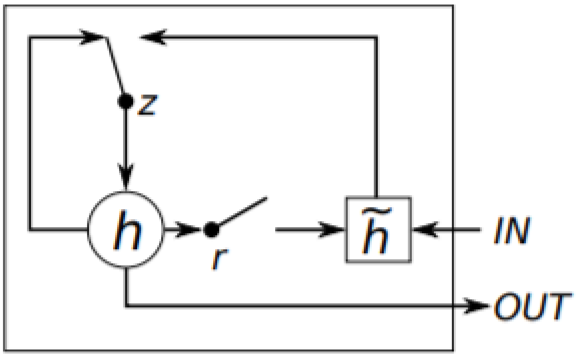
$$\theta^{*} = arg\,max_{\theta} \sum_{I,S} \log P(S|I;\theta)$$

(1) Equation 1: Objective function where θ are the parameters of the model, I is an image, and S its correct transcription

S in the equation denotes a sentence and it's length is usually large. Thus, it is common to apply the chain rule to model the joint probability over S_0, ..., S_N where N is the length of this particular sentential transcription (also called caption) as

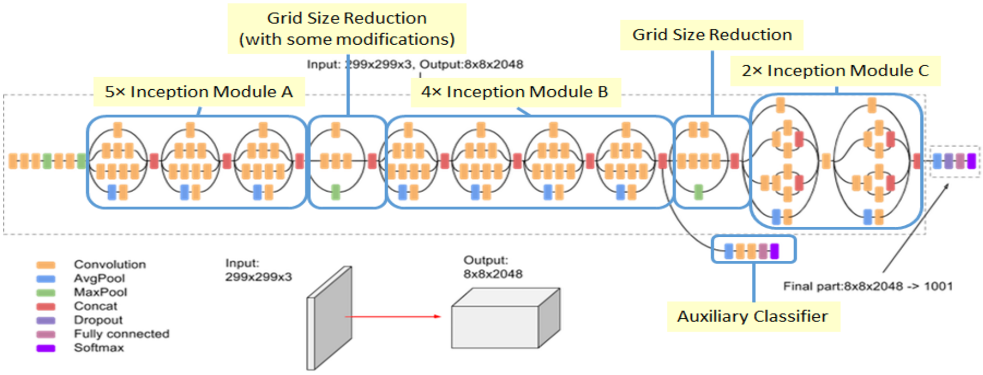
$$logP(S|I;\theta) = \sum_{t=0}^{N} log(S_t|I, S_0, S_1, ...S_{t-1};\theta) \quad (2)$$

At training time, *(S, I)* is a training example pair, and we optimize the sum of the log probabilities as described in equation (2) over the whole training set using Adam optimizer. It is natural to model *P* $(S_t |I, S_0, S_1, ...S_{t-1})$ with a Recurrent Neural Network (RNN), where the variable number of words we condition upon up to $t - 1$ is expressed by a fixed length hidden state or memory h_t . This memory is updated after seeing a new input x_t by using a nonlinear function $f : ht + 1 = f(ht, xt)$



Gated Recurrent Unit

The RNN for this purpose used in our project is a Gated Recurrent Unit which is shown as above.



A gated recurrent unit (GRU) was proposed by [4] Cho et al. [2014] to make each recurrent unit to adaptively capture dependencies of different time scales. Here as well, GRU has gating mechanism like the traditional LSTM unit, but it doesn’t have separate memory cells to capture the information like them. Thus, making the model a less expensive to train.

The activation $\boldsymbol{h}^j{}_t$ of the GRU at time t is a linear interpolation between the previous activation $\boldsymbol{h}^j{}_{t-1}$ and the candidate activation $\boldsymbol{h}^{\tilde{j}}{}_t$:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j,$$

where an **update gate** ($\boldsymbol{z}^j{}_t$) decides how much the unit updates its activation, or content. The update gate is computed by

$$z_t^j = \sigma \left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} \right)^j$$

This procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit. The GRU, however, does not have any mechanism to control the degree to which its state is exposed, but exposes the whole state each time. The candidate activation $\boldsymbol{h}^{\tilde{j}}{}_t$ is computed similarly to that of the traditional recurrent unit (see Eq. (2)) and as in [Bahdanau et al., 2014],

$$\tilde{h}_t^j = \tanh \left(W \mathbf{x}_t + U \left(\mathbf{r}_t \odot \mathbf{h}_{t-1} \right) \right)^j$$

Where r_t is a set of reset gates and $\odot$ is an element-wise multiplication.

When off ($\boldsymbol{r}^j{}_t$ close to 0), the reset gate effectively makes the unit act as if it is reading the first symbol of an input sequence, allowing it to forget the previously computed state. The reset gate $\boldsymbol{r}^j{}_t$ is computed similarly to the update gate:

$$r_t^j = \sigma \left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} \right)^j$$

In addition to all of this our project uses attention mechanism to predict next words of the sentence

3. Attention mechanism

Using visual attention mechanism according to [1] Xu et al., we make the decoder be able to look at different parts of the image at every time step in the sequence

We have implemented the attention mechanism described by [9] Badahnau et al

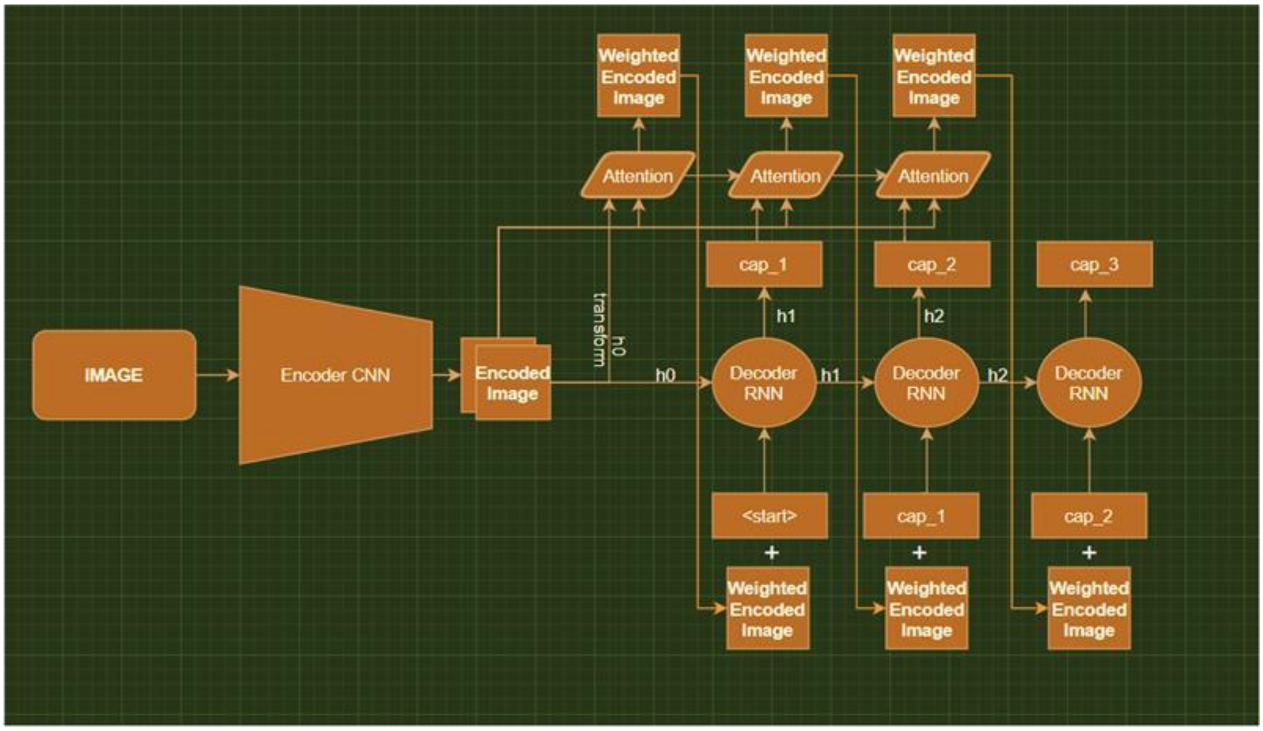
This is also known as a blend of hard and soft attention mechanism

So, in general without attention we would just compute average of the pixels, to guess the next word.

But with attention we calculate the weighted average across all the pixels, and use the one where the weights are greater.

These weights are calculated by the attention model and sent with the target captions for the subsequent time steps.

The weighted average calculated over the pixels should sum to 1.



The Whole encoder-decoder architecture of our model

Experiments and Modifications

Specifically, for any project involving generating captions for a given image, caption semantics should be as clear as possible and consistent with the given image content.

Keeping this in mind, the model was trained on both Flickr 8k and MS-COCO dataset.

The model uses a 48-layer inception v3 for embedding image features which is fed into the attention model and the first-time step of the double layer RNN which is constituted of Gated recurrent unit (GRU). The last encoded hidden state can be uses as the first hidden state in the decoder.

The attention model has two hidden layers which will be added inside the $tanh$ function to give the total attention and using SoftMax as an output.

SoftMax used at the interior of the model as well for self-normalizing as SoftMax approaches one.

We use tokenizers for marking the unknown words which are not specified in dictionary and mainly for Padding each vector to the max_length of the captions.

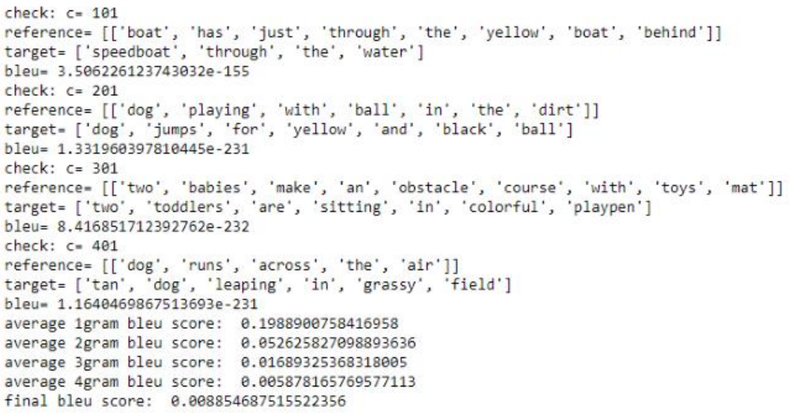
Training datasets with a BATCH_SIZE = 64, BUFFER_SIZE = 1000, embedding_dim = 256

After train is calculated, it is to apply gradient descent on loss to minimize it, and loss is calculated using y0 and y_out, and so gradient descent will also affect those (if they are trainable variables), and so on.

Tokenizer allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count.

The RNN size in this case is 512. Since words are one hot encoded, the word embedding size and the vocabulary size is also 8357.

During training we used different batch sizes and got different running time.



```
check: c= 101
reference= [['boat', 'has', 'just', 'through', 'the', 'yellow', 'boat', 'behind']]
target= ['speedboat', 'through', 'the', 'water']
bleu= 3.506226123743032e-155
check: c= 201
reference= [['dog', 'playing', 'with', 'ball', 'in', 'the', 'dirt']]
target= ['dog', 'jumps', 'for', 'yellow', 'and', 'black', 'ball']
bleu= 1.331960397810445e-231
check: c= 301
reference= [['two', 'babies', 'make', 'an', 'obstacle', 'course', 'with', 'toys', 'mat']]
target= ['two', 'toddlers', 'are', 'sitting', 'in', 'colorful', 'playpen']
bleu= 8.416851712392762e-232
check: c= 401
reference= [['dog', 'runs', 'across', 'the', 'air']]
target= ['tan', 'dog', 'leaping', 'in', 'grassy', 'field']
bleu= 1.1640469867513693e-231
average 1gram bleu score:  0.1988900758416958
average 2gram bleu score:  0.052625827098893636
average 3gram bleu score:  0.01689325368318005
average 4gram bleu score:  0.005878165769577113
final bleu score:  0.008854687515522356
```

Hardware specification

We trained our model in our own laptop, which definitely had its own limitations. So, we fixed a limited number of images to train and validate for COCO dataset (30000 images) and also used a lighter dataset – Flickr8k.

GPU: Nvidia GTx 1070 MAX Q

CPU: Intel core i7 – 7th gen.

It takes around 1 hour for training with Flickr-8k dataset

It took around 10 hours for training with COCO dataset.

Evaluation Metrics

Checking the accuracy of the model is tricky as the output is a sequence of words which could be correct even if the predicted caption is not exactly matching with the original caption. BLEU score can be used to measure the performance of the model

We have used NLTK library for the calculation of BLEU score.

We have calculated all the four (1-gram, 2-gram, 3-gram and 4-gram) sentence BLEU Score.

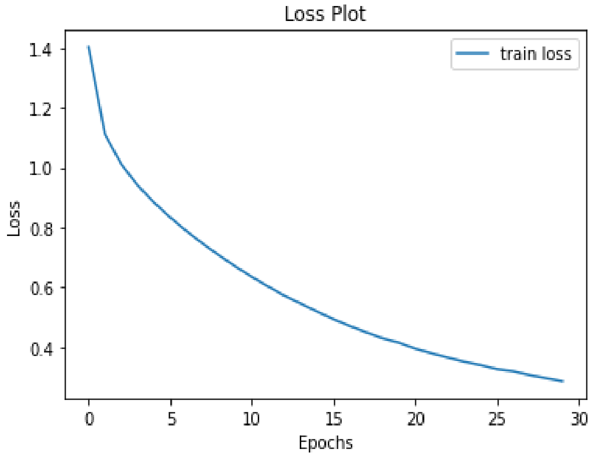
The result we have obtained for Flickr – 8k dataset is as shown in the picture above.

Results and Discussions

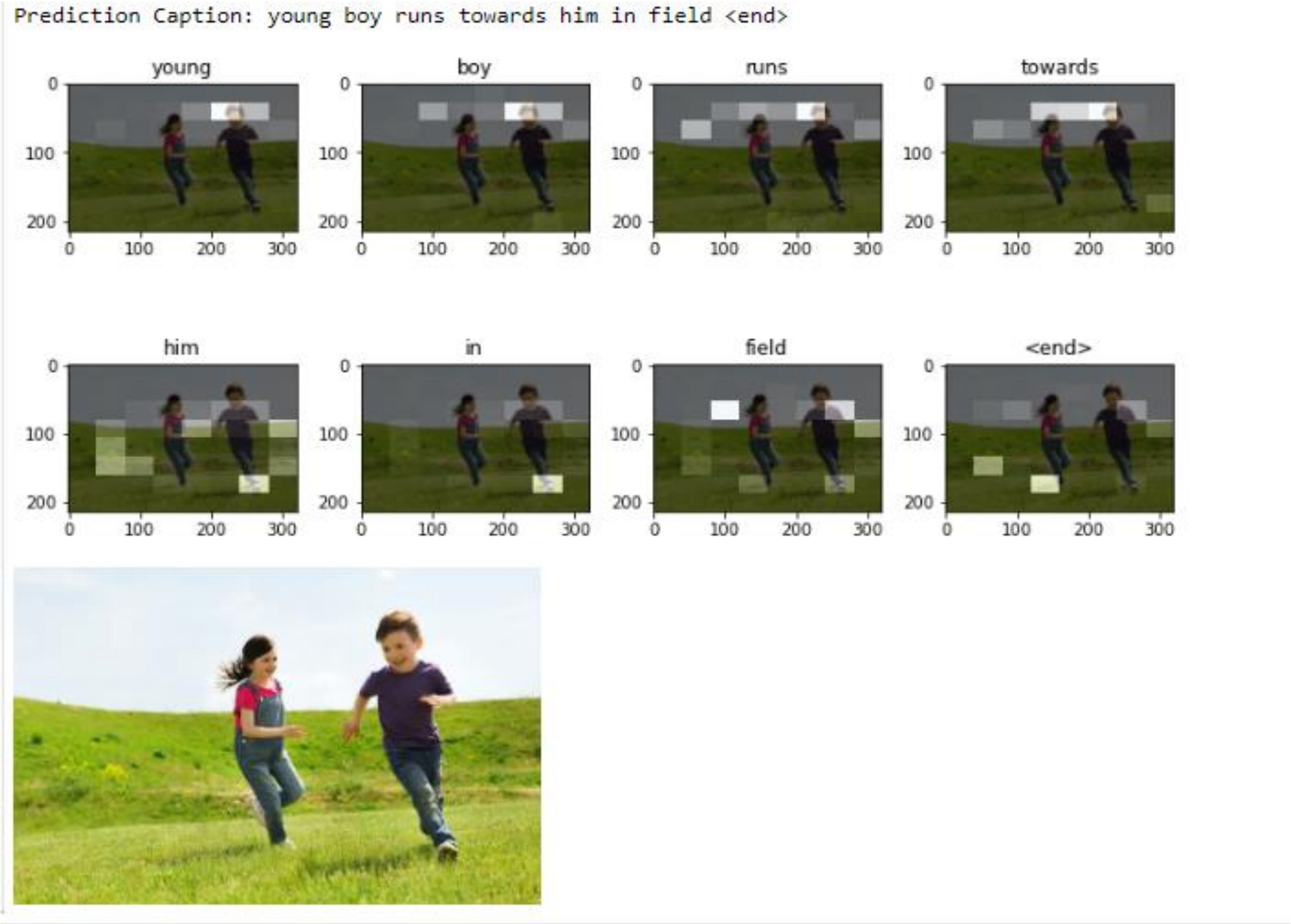
Following are a few key hyperparameters that we retained across various models. These could be helpful for attempting to reproduce our results.

RNN Size	512
Batch size	64
Learning Rate	4e-4
RNN Sequence max length	16
Epochs	30
Vocabulary size	5000

Following graph shows the drop in cross entropy loss against the training iterations for InceptionV3+GRU model. It is to be noted that each iteration was for one batch of images.

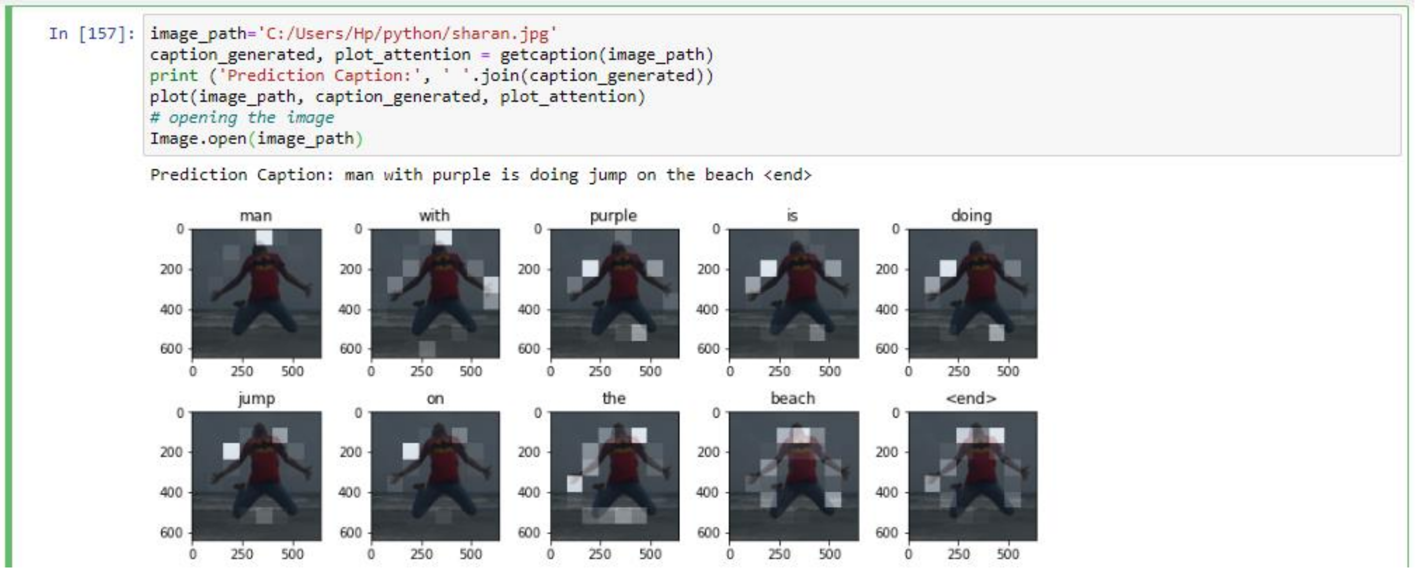


Here are the images of the captions that we generated due course of the experiments with our self-made dataset.



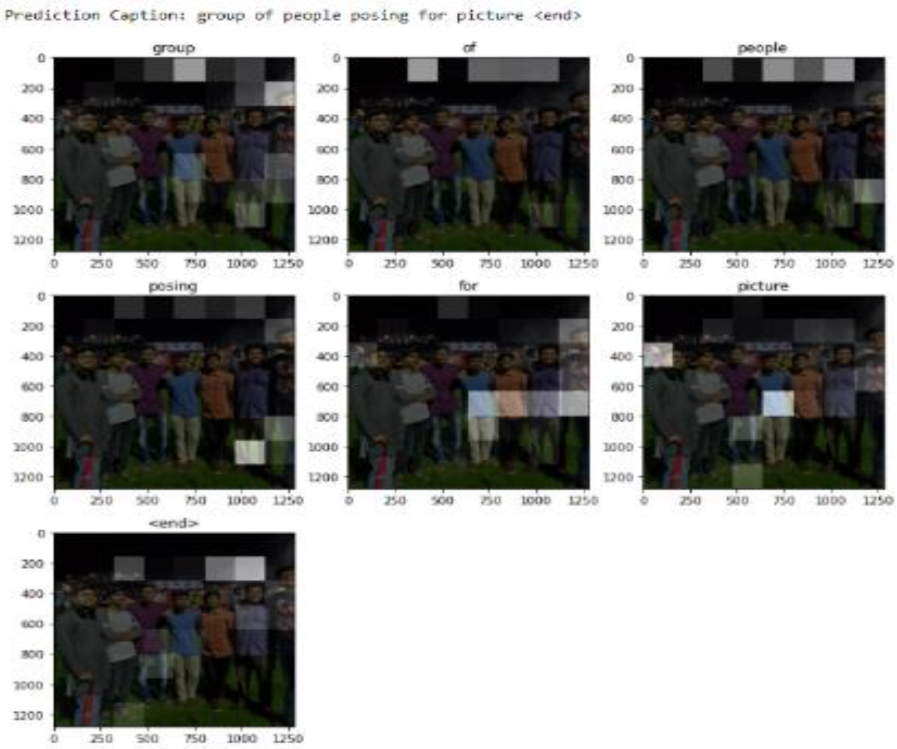
```
Prediction Caption: young boy runs towards him in field <end>
```

This is Sharan Nagarajan – teammate



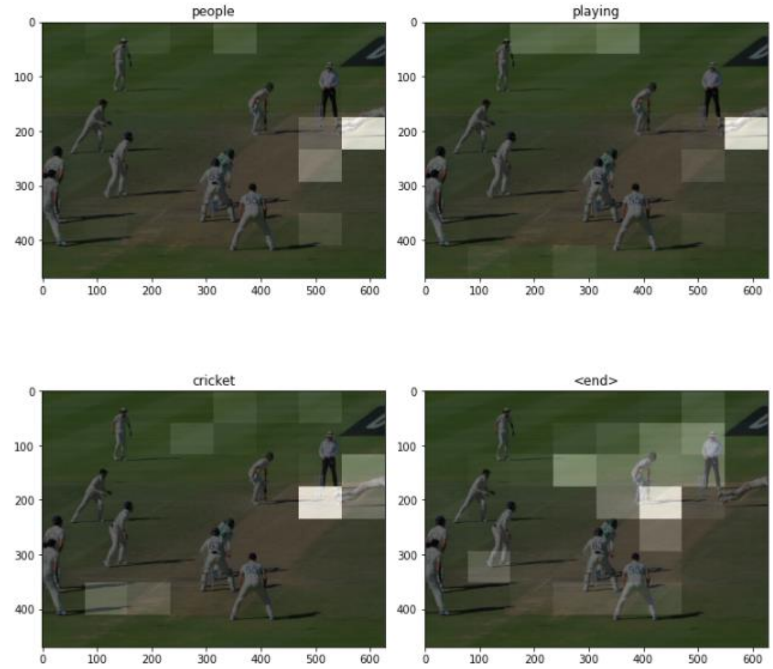
```
In [157]: image_path='C:/Users/Hp/python/sharan.jpg'
          caption_generated, plot_attention = getcaption(image_path)
          print ('Prediction Caption:', ' '.join(caption_generated))
          plot(image_path, caption_generated, plot_attention)
          # opening the image
          Image.open(image_path)
```

```
Prediction Caption: man with purple is doing jump on the beach <end>
```



```
Prediction Caption: group of people posing for picture <end>
```

```
Prediction Caption: people playing cricket <end>
```



Conclusion

In this project we have implemented various individual concepts as CNN, RNN, Visual attention mechanism, transfer learning etc., and integrate them into a single model and hence it has provided us a deeper understanding of how deep learning is implemented as a whole and now we are ready to go through any journey of AI ahead.

REFERENCES

[1] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. (2016) Show attend and tell: Neural image caption generation with visual attention. [Online]. Available: https://arxiv.org/abs/1502.03044

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. (2015) Show and tell: A neural image caption generator. [Online]. Available: https://arxiv.org/abs/1411.4555

[3] Dzmitry Bahdanau, KyungHyun Cho Yoshua Bengio (2015) Neural Machine Translation by jointly learning to align and translate

[4] Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation by Cho et al. [2014]

[5] https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

[6] https://github.com/SubhamIO/Image-Captioning-using-Attention-Mechanism-Local-Attention-and-Global-Attention-

[7] https://www.tensorflow.org/tutorials/text/image_captioning

[8] Image Captioning by Vikram Mullachery, Vishal Motwani (2016)

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, [2014] Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling