INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

VISUAL RECOGNITION
AI 825

May 15, 2023

Mini-Project

Image Captioning CNN-LSTM

Team 28 - Members		
Name	***Roll No***	***Email***
BTV Sumanth	IMT2020072	sumanth.badam@iiitb.ac.in
Chinmay Parekh	IMT2020069	Chinmay.Parekh@iiitb.ac.in
Aakash Khot	IMT2020512	Aakash.Khot@iiitb.ac.in
Pavan Thanay	IMT2020024	Pavan.Muthyala@iiitb.ac.in

Introduction

We have to design a CNN-LSTM system that can perform image captioning on the Flickr8K dataset.

We will be using Flickr8K dataset for training and testing the model. The Flickr8K dataset is a popular benchmark dataset used for image captioning tasks. It consists of 8,000 images and each image in the dataset is accompanied by five human-generated captions, resulting in a total of 40,000 captions. The dataset provides a diverse set of images, including various scenes, objects, and activities, which makes it suitable for training and evaluating image captioning models.

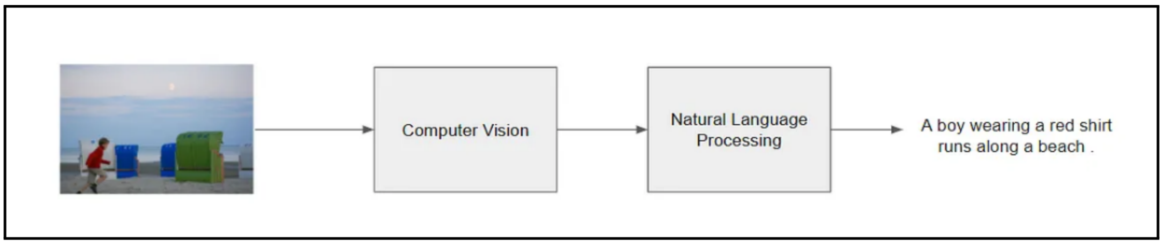

Computer Vision
Natural Language Processing
A boy wearing a red shirt runs along a beach .

Figure 1: A high-level flow of the solution

Data Cleaning - Captions

When we deal with text/NLP tasks, we generally perform some basic cleaning like lower-casing all the words in captions given, removing special tokens (like '!','@', etc.), and eliminating words that contain numbers.

1. Lowercasing all the letters in the text data helps to standardize the text and reduce the vocabulary size. This is important because it helps to avoid duplication of words and improves the effectiveness of subsequent text processing steps like word embedding. It also reduces the sparsity of the data.

2. Special tokens often add unnecessary noise to the text data. Removing special tokens simplifies the text representation and reduces the vocabulary size.

We also observe that there are more than 8000 unique words in 40,000 captions given to images. Since we are creating a predictive model, we would not like to have all the words present in our vocabulary but the words which are more likely to occur or which are common. This helps the model become more robust to outliers and make less mistakes. Hence we consider only those words which occur at least 10 times in the entire corpus of data.

Adding tokens in every caption

While we load the images and captions of the data, we will add two tokens in every caption, 'startseq' - This is a start sequence token that will be added at the start of every caption and 'endseq' at the end of every caption.By adding this tokens, we explicitly define the boundaries of the caption sequence. This allows the model to understand where the caption starts and ends, which is crucial for training and generating captions.
During training, the "startseq" token serves as the initial input to the caption generation model, indicating the beginning of the caption.The "endseq" token represents the end of the caption, helping the model understand when to stop generating words.

Data Preparation

At last, we want to predict the captions. Our model needs to learn to predict the captions of input images provided. But at a point in time, we won't be predicting the entire caption but rather word by word. Thus, we need to encode each word into a fixed-sized vector.
We will be creating two dictionaries, "wordToIndex" (word to index, will return index given word) and "indexToWord" (index to word, will return word given index).We will be representing every unique word in the vocabulary by an integer (index).
Now we want to give this data as input to our model, for this we need to prepare the data suitable to be given as input to the model.
Let us take 3 images, two images to train and one to test. The data is preprocessed/cleaned and we have given index to each vocabulary. Let us see this task as a **supervised learning problem**, where we have a set of data points D = X_i, Y_i, where X_i is the feature vector of data point 'i' and Y_i is the corresponding target variable (Caption).

Prediction of Caption

Firstly we will provide image vector of first image and first word ("startseq") as input and try to predict the second word. Then we provide image vector and the first two words as input and try to predict the third word. And this goes on until we get "endseq" as our Target variable.
Similarly, we will be doing this procedure for 2nd Image.

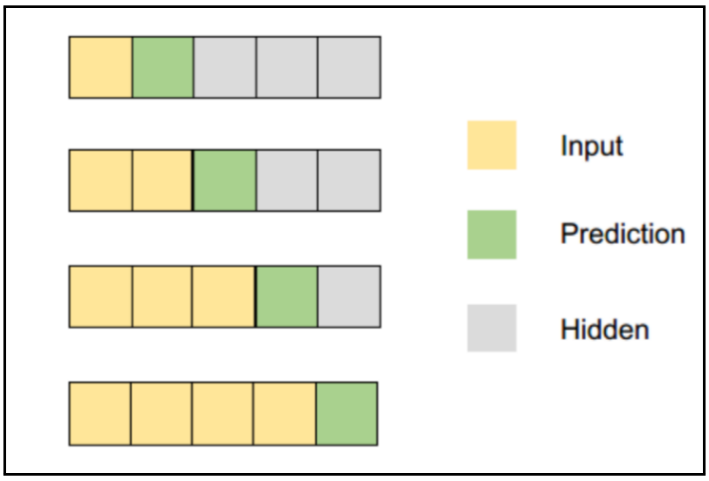


Figure 2: Caption Prediction

We can clearly observe here that it's not just the image that goes as input to the system, but also, a partial caption that helps to predict the next word in the sequence. We will be using Recurrent Neural Network (LSTM) to read these partial captions.
Now in this procedure we will be passing indices of words instead of word itself where each index represents a unique word. We will also be padding 0s for equal-length sequence for batch processing. We will be adding zeros according to maximum length of caption.
Now this is for 2 training images, but we will be having more than 6000 images. So later we will see that each word/index is mapped to a 200-long vector using a pre-trained GLOVE word embedding model. For this huge data, we will be using **data generators**.

Data Generator

The use of a data generator is a common approach when dealing with large datasets in machine learning tasks such as our task - image captioning.

A data generator provides a way to efficiently load and process data in batches during training, allowing you to work with large datasets that may not fit entirely into memory. This means that we do not require to store the entire dataset in the memory at once. Even if we have the current batch of points in the memory, it is sufficient for our purpose.

Vision Embeddings

The vision embedding represents the visual information of the input image. It is obtained by passing the image through a convolutional neural network (CNN) and extracting the output of a specific layer. The output of this layer contains high-level visual features that capture the image's content and structure.
A common approach for fixed length representation as vector is to apply global average pooling. This results in an averaged image feature vector that summarizes the visual information in a compact representation. The averaged image feature vector serves as the initial input to the Long Short-Term Memory (LSTM) for generating the caption.

Word Embeddings

We will be mapping the every word-index to a 200-long vector. Benefits of Word embeddings :-

1. Word embeddings provide a **distributed representation** of words, where each word is represented by a vector of continuous values. This representation captures semantic relationships between words.

2. It also helps to **reduce the dimensionality** of the input space. Instead of representing words as one-hot encoded vectors, word embeddings convert them into continuous-valued vectors of fixed dimensions. This reduces the memory and computational requirements and enables more efficient training.

We need to find a good word embedding to boost generalization and the overall performance. We will using a pre-trained GLoVe Model for this. **GLoVe (Global Vectors for Word Representation)** is a popular pre-trained word embedding model that learns word representations based on the global co-occurrence statistics of words in a corpus. By utilizing a pre-trained GLOVE model,one can map each word (index) to a x-dimensional vector, capturing the semantic and syntactic properties of the words. These embeddings can then be used as input features for Image captioning task.
We use the same word embeddings for both the baseline and the modified baseline models.

CNN-LSTM architecture

We will be using CNN-LSTM architecture model to train the Flickr8K dataset.
CNNs are powerful deep learning models for processing images. They are designed to extract hierarchical feature representations from input images. By using multiple convolutional layers with pooling and non-linear activation functions, CNNs can capture local patterns and gradually learn more abstract and high-level features.

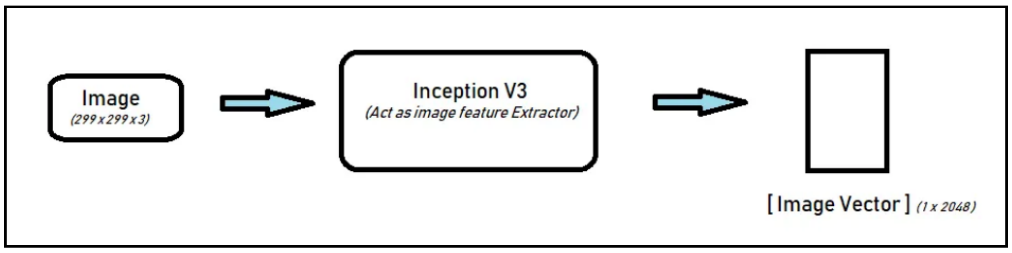


Figure 3: IncpetionV3 Architecture

LSTMs, on the other hand, are a type of recurrent neural network (RNN) that excel at modeling sequential data. LSTMs are specifically designed to address **the vanishing gradient problem** in standard RNNs, allowing them to capture long-range dependencies and remember information over longer sequences. LSTMs have an internal memory cell and a set of gates that control information flow, making them suitable for tasks that involve sequential processing and temporal dependencies.

Feature Extraction

Convolutional Neural Network (CNN) is used for Feature extraction of Images. **InceptionV3** model is used as a feature extractor from images. InceptionV3 is a popular convolutional neural network architecture that was introduced by Google researchers as part of the Inception series. It can be used as a feature extractor in various computer vision tasks, including image captioning.

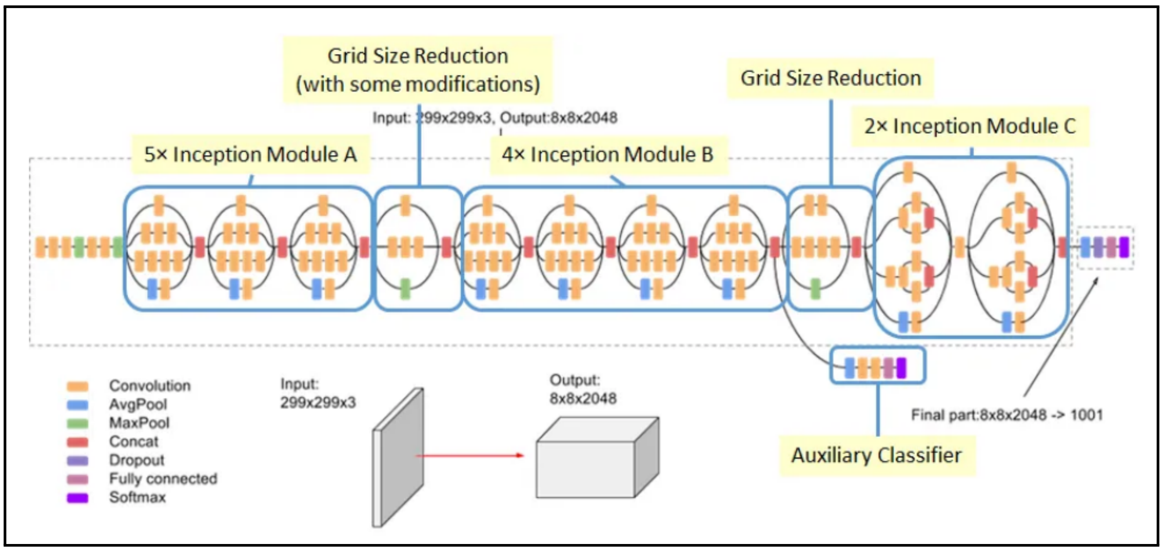


Figure 4: IncpetionV3 Architecture

Some key aspects of the model:-

1. Factorized convolutions are performed in order to reduce the computational efficiency as it reduces the number of parameters involved in a network.
2. Bigger convolutions are replaced by smaller convolutions which leads to faster training of the model.
3. InceptionV3 incorporates pooling layers, such as max pooling, to downsample feature maps and reduce spatial dimensions.
4. It also employs batch normalization to accelerate training and improve generalization.

The InceptionV3 module has been pre-trained on large-scale image classification datasets, such as ImageNet, which contain millions of labeled images from thousands of categories. This pre-training enables the network to learn generic visual representations that can be used as feature extractors for image captioning task.
Our purpose here is not to classify the image but just to get a fixed-length informative vector for each image. This process is called **automatic feature engineering**. Hence, we just remove the last softmax layer from the model and extract a 2048 length vector - bottleneck features for every image as shown in Figure 1.
The above InceptionV3 module is used as a feature extractor for the baseline model. For the modified baseline model, we use pre trained ViT (Vision Transformer), finetuned on ImageNet, i.e., we extract the last hidden layer's output as the feature representation of an image (after pooling). This results in a 768 dimensional vector, that would be the image encoding for the baseline model. The purpose of doing this is in a ViT, self attention across different patches of the images is calculated, which is a richer representation of an image (than without attention).

Caption Generation

Generating image captions involves a "sequence" of predicted words. We will be using RNN specifically **Long Short-Term Memory (LSTM)** architecture, a type of RNN designed to address the vanishing gradient problem, making them effective for capturing sequential dependencies in language modeling and generating context-aware captions.

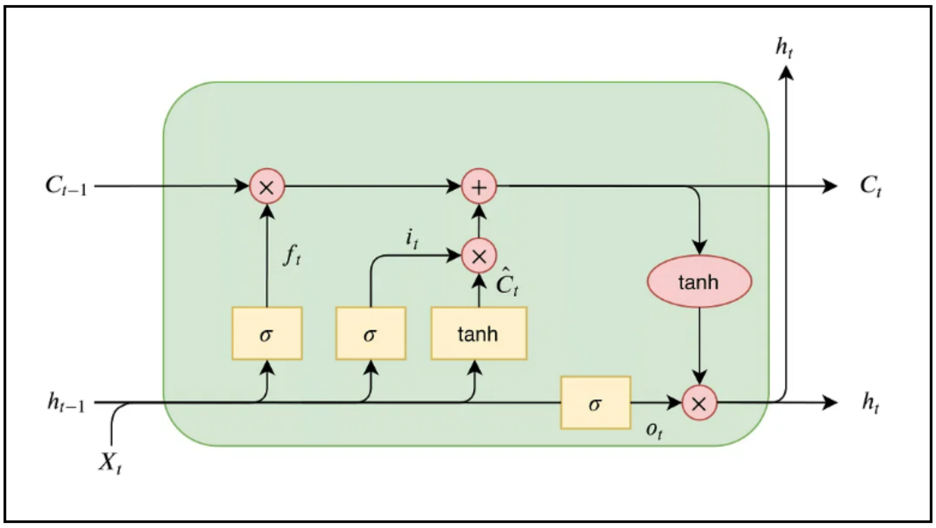


Figure 5: LSTM Architecture

Caption generation involves modeling the sequential nature of language. LSTMs excel at processing sequential data by maintaining an internal memory state that can retain information over long sequences.
LSTMs mitigate the vanishing gradient problem that can occur in standard RNNs when backpropagating errors over long sequences. The LSTM's memory cell, with its forget gate, input gate, and output gate mechanisms, enables the network to selectively retain or discard information over multiple time steps. This allows the LSTM to capture long-term dependencies in the text, which is crucial for generating meaningful captions.

Model Architecture

The inputs to the final CNN-LSTM model are the preprocessed image (224, 244, 3) and the integer tokens of the captions.The image's features are extracted and reduced to 256 dimensions using a Linear layer with ReLU activation.

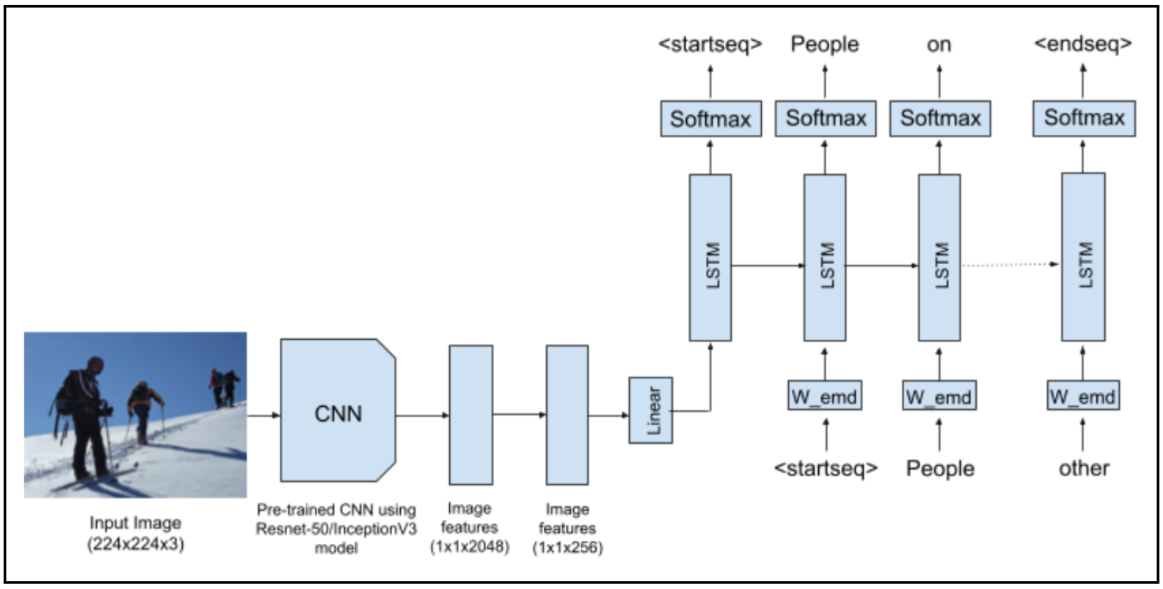
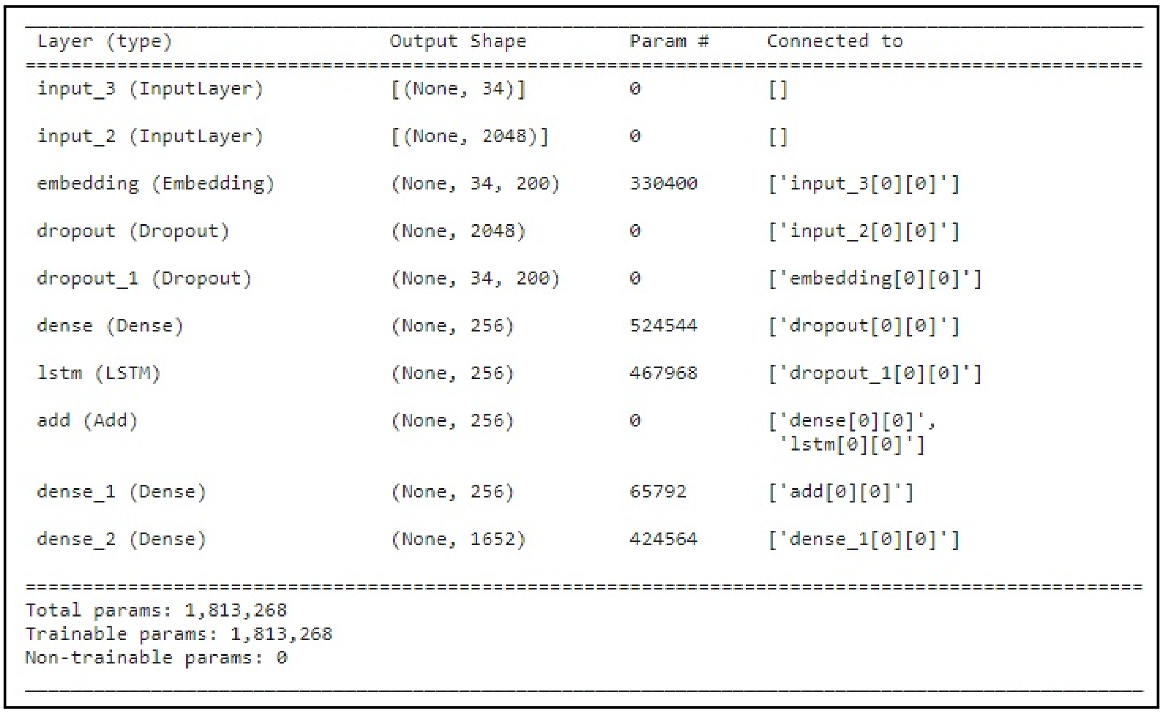


Figure 6: CNN-LSTM Model

The vectorized image representation which we already have is fed into the network, followed by a sentence token. The hidden state produced is then used by the LSTM to predict or generate the caption for the given image.

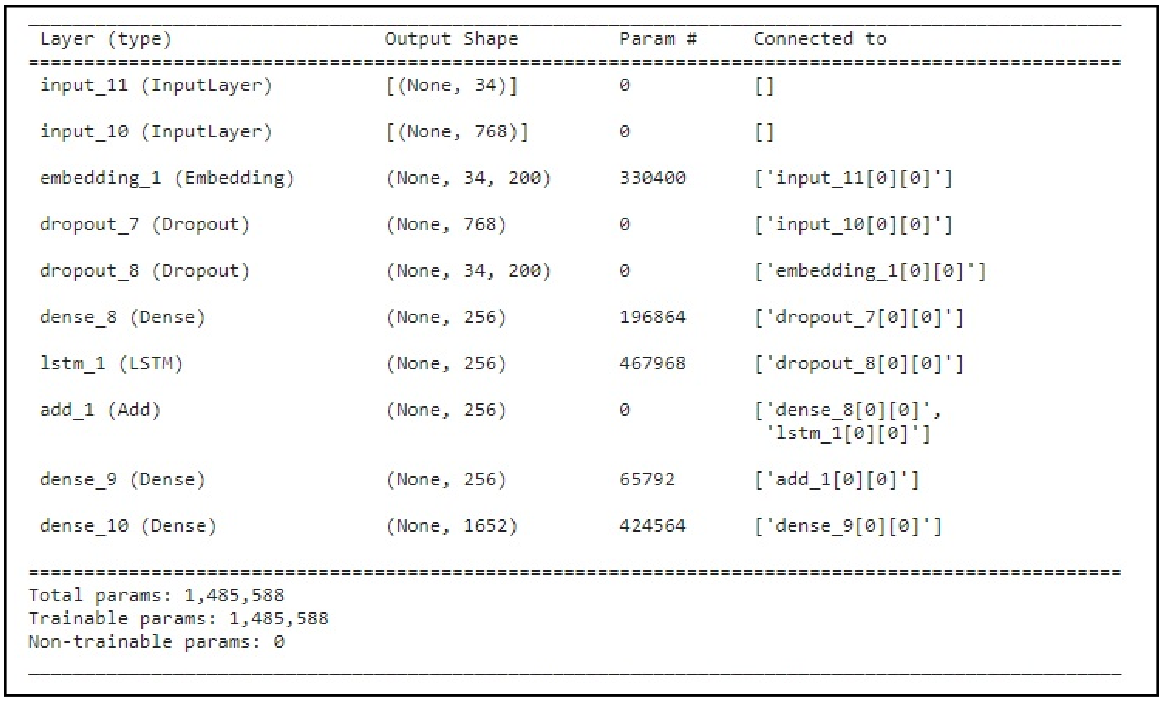
We use Categorical Cross entropy loss for updating the parameters. In summary, We have inputs as partial caption and Image feature vector and output as another partial caption including the next predicted word.
The baseline model is trained for 20 epochs, with batch size equal to 3. Adam's optimizer has been used for the same.
The modified baseline model is trained for 12 epochs, with batch size equal to 4. Adam's optimizer has been used for the same.

Model Summary



```
________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================================
 input_3 (InputLayer)           [(None, 34)]         0           []

 input_2 (InputLayer)           [(None, 2048)]       0           []

 embedding (Embedding)          (None, 34, 200)      330400      ['input_3[0][0]']

 dropout (Dropout)              (None, 2048)         0           ['input_2[0][0]']

 dropout_1 (Dropout)            (None, 34, 200)      0           ['embedding[0][0]']

 dense (Dense)                  (None, 256)          524544      ['dropout[0][0]']

 lstm (LSTM)                    (None, 256)          467968      ['dropout_1[0][0]']

 add (Add)                      (None, 256)          0           ['dense[0][0]',
                                                                  'lstm[0][0]']

 dense_1 (Dense)                (None, 256)          65792       ['add[0][0]']

 dense_2 (Dense)                (None, 1652)         424564      ['dense_1[0][0]']

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

Figure 7: Model summary of baseline



```
________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================================
 input_11 (InputLayer)          [(None, 34)]         0           []

 input_10 (InputLayer)          [(None, 768)]        0           []

 embedding_1 (Embedding)        (None, 34, 200)      330400      ['input_11[0][0]']

 dropout_7 (Dropout)            (None, 768)          0           ['input_10[0][0]']

 dropout_8 (Dropout)            (None, 34, 200)      0           ['embedding_1[0][0]']

 dense_8 (Dense)                (None, 256)          196864      ['dropout_7[0][0]']

 lstm_1 (LSTM)                  (None, 256)          467968      ['dropout_8[0][0]']

 add_1 (Add)                    (None, 256)          0           ['dense_8[0][0]',
                                                                  'lstm_1[0][0]']

 dense_9 (Dense)                (None, 256)          65792       ['add_1[0][0]']

 dense_10 (Dense)               (None, 1652)         424564      ['dense_9[0][0]']

================================================================================================
Total params: 1,485,588
Trainable params: 1,485,588
Non-trainable params: 0
________________________________________________________________________________________________
```

Figure 8: Model summary of modified baseline

Inference/Testing Phase

Here we will understand how we are going to test the model. Recall that example where we took 2 training images and 1 testing example. We will now use the testing image to generate the caption.

We will be generating caption one word at a time but this time we will a set of words from the vocabulary.For this reason we **greedily** select the word with the maximum probability, given the feature vector and partial caption.
This is also called as Maximum Likelihood Estimation (MLE) i.e. we select that word which is most likely according to the model for the given input. And sometimes this method is also called as Greedy Search, as we greedily select the word with maximum probability.
We will stop the iteration once we get "endseq" token or we reach a maximum threshold of the number of words generated by the model. If any of these conditions are met, we break the loop and report the generated caption.

Evaluation

BLEU Score

BLEU stands for Bilingual Evaluation Understudy.The BLEU score is a commonly used metric in image captioning tasks to evaluate the quality of generated captions by comparing them to reference captions.
We can use BLEU to check the quality of our generated caption.Let's take an example to explain how to calculate the score.
Predicted Caption = "the weather is good" References are "the sky is clear" and "the weather is extremely good".First, convert the predicted caption and references to unigram/bigrams.

$$\text{modified ngram precision} = \frac{\text{max no. of times ngram occurs in reference}}{\text{total no. of ngrams in hypothesis}}$$

So predicted pairs are (the',weather),(weather,is),(is,good), Similarly we have have for References. BLEU score (Bigram) = 1/3 for (the,weather) + 1/3 for (weather,is) + 0/3 for (is,good) = 0.667.
BLEU tells how good is our predicted caption as compare to the provided reference captions.

METEOR Score

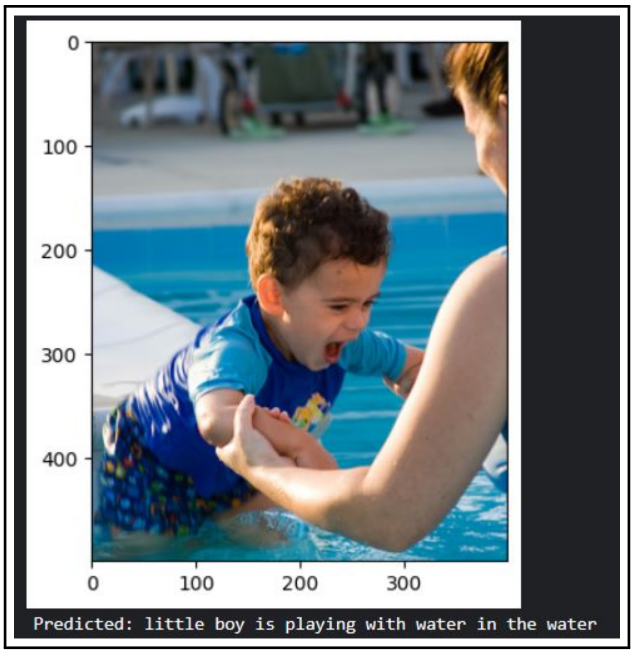
The METEOR (Metric for Evaluation of Translation with Explicit ORdering) metric is commonly used for evaluating machine translation tasks, like image captioning. METEOR assesses the quality of generated captions by comparing them to reference captions.
METEOR employs an alignment process to match n-grams between the predicted and reference captions. This alignment accounts for the ordering and position of the words in the sentences.
METEOR calculates the n-grams precision scores between the predicted and reference captions. These scores measure the overlap of n-grams, taking into account both exact matches and matches that are partially correct.
Additionally, METEOR computes a penalty for unaligned words, penalizing incorrect words in the generated captions that do not align with any reference words.
Harmonic Mean: The precision scores are combined using a harmonic mean, giving higher weightage to higher-order n-grams. This combined precision is called the METEOR precision.
The final METEOR score is computed by combining the METEOR precision and the penalty term.

The main difference between BLEU and METEOR scores lies in their approach to evaluating generated captions. BLEU focuses on n-gram precision, measuring the overlap of n-grams between generated and reference captions. On the other hand, METEOR incorporates both precision and recall, taking into account overall similarity, partial matches, and word order differences. It also has a penalizing system in case of discrepancies.

Models	Testing BLEU	Testing METEOR
Baseline	0.5505	0.3156
Modified Baseline	0.5603	0.2017

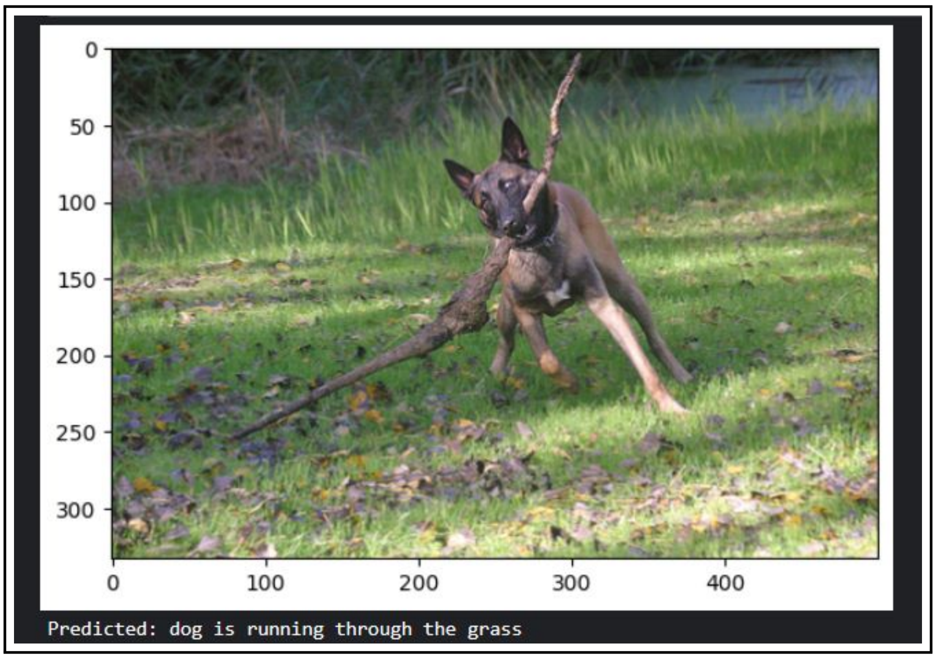
Image Predictions and Analysis

A few images captioned by baseline model :



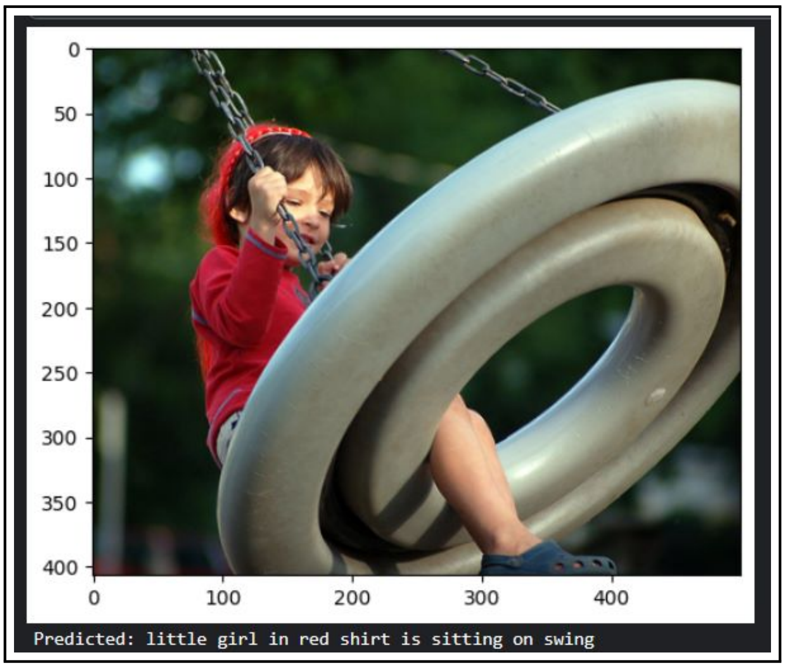
Predicted: little boy is playing with water in the water

Figure 9: Prediction 1



Predicted: dog is running through the grass

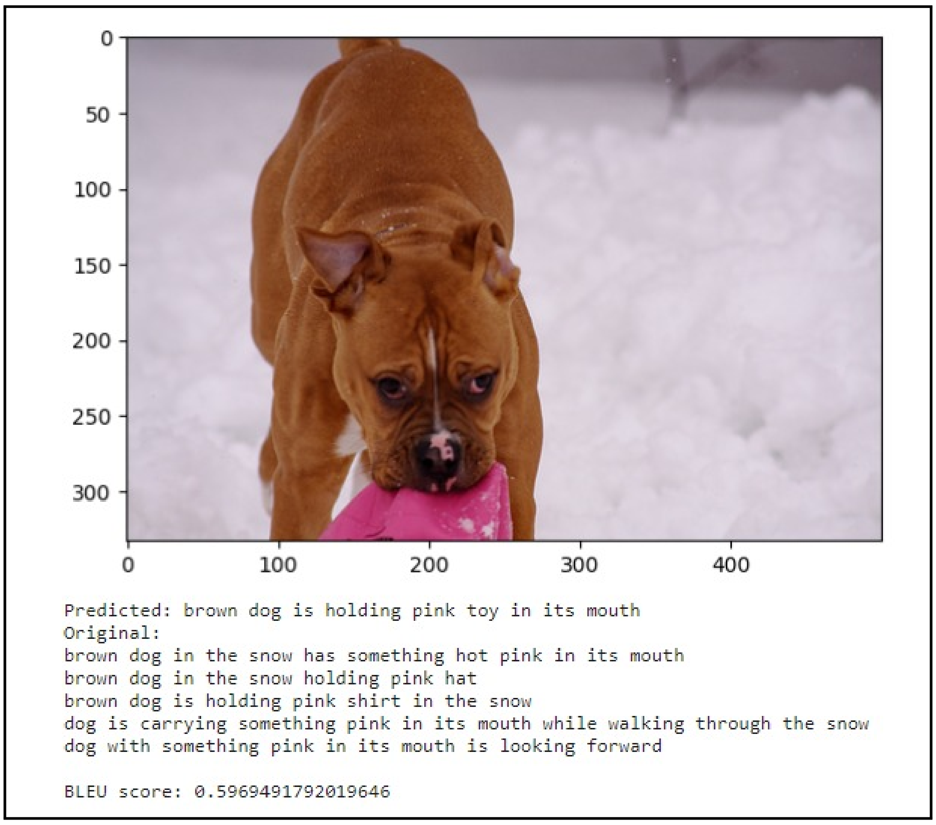
Figure 10: Prediction 2



```
Predicted: little girl in red shirt is sitting on swing
```

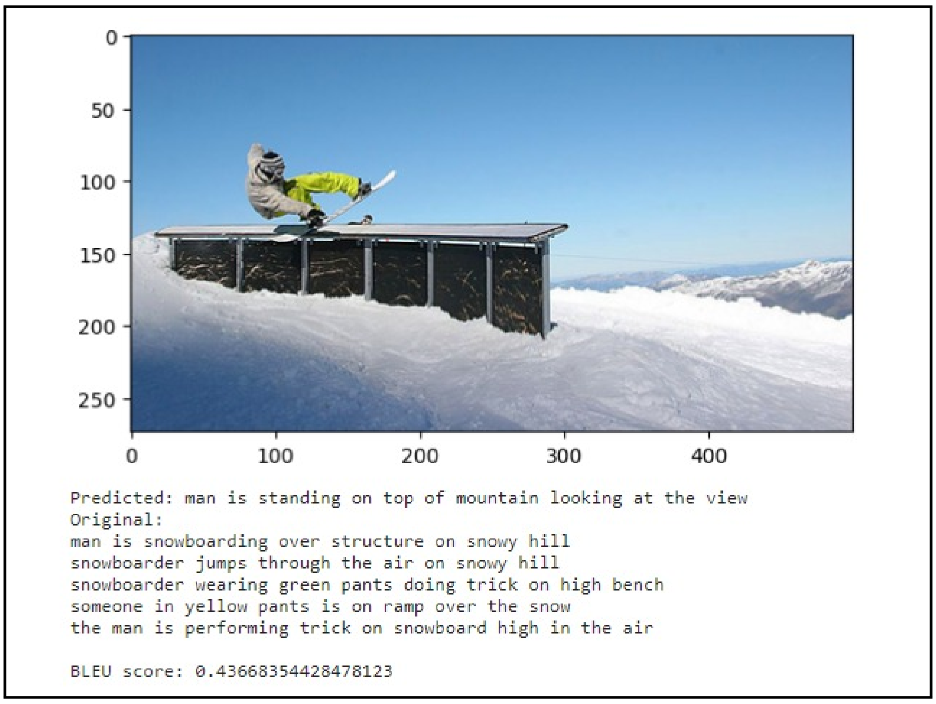
Figure 11: Prediction 3

A few images captioned by modified baseline model :



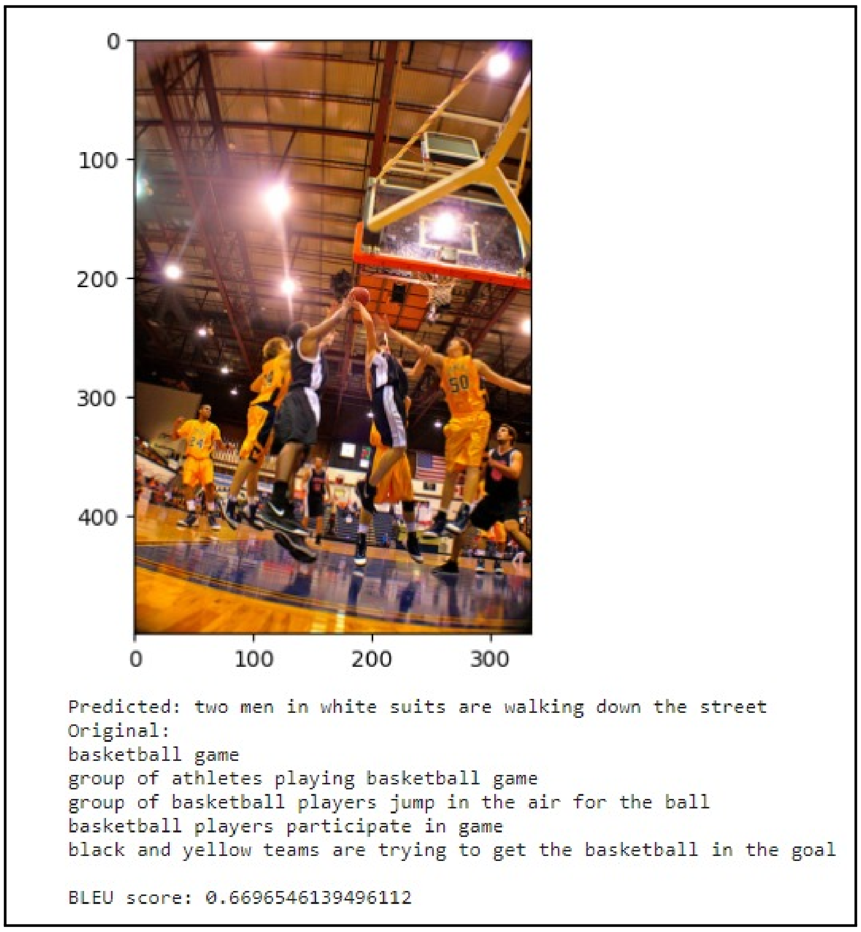
```
Predicted: brown dog is holding pink toy in its mouth
Original:
brown dog in the snow has something hot pink in its mouth
brown dog in the snow holding pink hat
brown dog is holding pink shirt in the snow
dog is carrying something pink in its mouth while walking through the snow
dog with something pink in its mouth is looking forward

BLEU score: 0.5969491792019646
```

Figure 12: Prediction 1



```
Predicted: man is standing on top of mountain looking at the view
Original:
man is snowboarding over structure on snowy hill
snowboarder jumps through the air on snowy hill
snowboarder wearing green pants doing trick on high bench
someone in yellow pants is on ramp over the snow
the man is performing trick on snowboard high in the air

BLEU score: 0.43668354428478123
```

Figure 13: Prediction 2



```
Predicted: two men in white suits are walking down the street
Original:
basketball game
group of athletes playing basketball game
group of basketball players jump in the air for the ball
basketball players participate in game
black and yellow teams are trying to get the basketball in the goal

BLEU score: 0.6696546139496112
```

Figure 14: Prediction 3

- From the scores in the above table, it is evident that modified baseline gives a better BLEU score as compared to the baseline model. This could be because of the richer image representations that are extracted in modified baseline model. We use the same word embeddings for both the models so as to observe the change in the performance of the model, when we just change the image representations.

- At the same time, we can also observe that baseline model gives a better METEOR score as compared to modified-baseline model. This could be because of the fact that the image embeddings obtained in the baseline model are 2048-dimensional, whereas the image embeddings obtained in the case of modified baseline model are only 768-dimensional.

- After visually inspecting the outputs of both the models for different images, it seems to be evident that there is some bias in the words that are being predicted. The models seem to be blindly predicting the presence of a man or a dog. Moreover, it seems to predict them with high confidence, i.e., the phrases the man in black shirt, two dogs, the dog is, etc. seem to be repeating more often. This could be due to the inherent language bias that is induced in the models because of the training data, i.e., the distribution of words and phrases in the train images' captions could be biased or the models lack the required complexity to capture the diversity of the data, and hence being biased towards a few phrases, containing specific colours, numbers or information.

References

- Image Captioning
- Reference Code