Image Captioning Technical Report

By - Rittik Panda(MT2022090)
Soumya Chakraborty(MT2022162)
Team_ID = 18

Image captioning is a task in the field of computer vision and natural language processing (NLP) that involves generating textual descriptions or captions for images. It combines the capabilities of both visual perception and language understanding to provide a comprehensive understanding of the visual content. This means, given an image we would like to get a description of what is going on in the image.

These are few examples of image captioning results:

Implementation

- **Train-Test-Validation Split:** At first we split our dataset into train set,test set and validation set according to the data-split given by sir.After splitting we get three dictionaries,where each key is a image_id and each value is caption of the image.In the given dataset we had multiple captions for an image,so in our dataset with one distinct image id key there exists a value which is a list of captions of that image.Finally training data contains 6000 key value pairs,testing data contains 1000 key value pairs and validation data also contains 1000 key value pairs.

- **Feature Extraction from images:**

The first step for image captioning is to generate vision embeddings of the images. We use the VGG-16 'ImageNet' pretrained network model for feature extraction of our images. At fist we knock off the last softmax layer of VGG-16 and take the output of last fully connected layer as feature vector of an image whose dimension is 4096 . This 4096-dimensional vector becomes our input to the LSTM module.After feature extraction we pickle the features and save them in a file called

Preprocessing of text:

The code clean(mapping) is a function that performs pre-processing on the captions in a given mapping. The mapping is a dictionary where the keys represent image names and the values are lists of captions associated with each image.The function iterates over each key-value pair in the mapping. For each key, it retrieves the list of captions associated with that key. Then, it iterates over each caption in the list.For each caption, the function applies several pre-processing steps to clean and format the text:

1.Convert the caption to lowercase.

2.Remove digits, special characters, and other non-alphabetic characters from the caption using regular expressions.

3.Remove additional spaces between words.

4.Add start and end tags to the caption to mark the beginning and end of the sequence. The start tag is appended at the beginning of the caption, and the end tag is appended at the end.

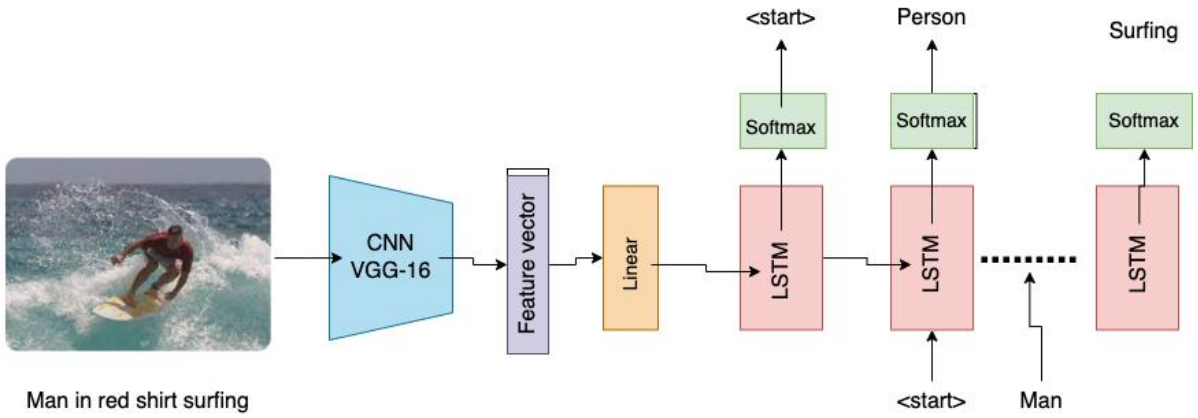
5.Update the modified caption in the list.

Tokenization and Vocabulary:

Since we cannot send the words directly through the LSTM module, we need a representation of each word in the caption corpus. We first tokenize each caption and create a Vocabulary of all the tokens in the corpus and assign them with unique ids I.e. indices of them in vocabulary list . These ids become our word representation that we pass through the network.

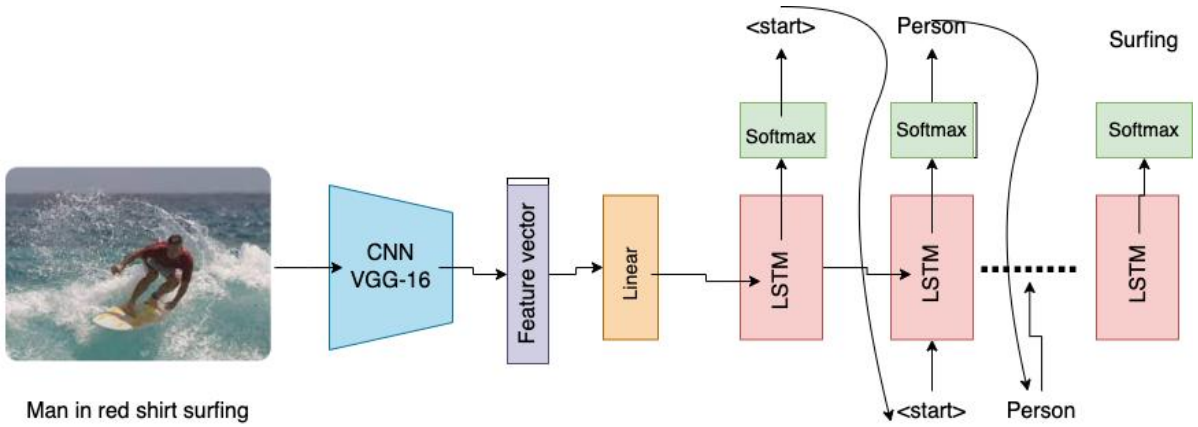
Training phase:

During the training of the model, we take each image and text pair, send the image embeddings through the LSTM cells as input and in every timestamp of the network, we feed the token representation of the ground truth word present in the captions in the next lstm no matter what is the predicted output. Each of our 5 captions per image form separate training data point. In the following figure, we can see that the Linear layer of the CNN embeddings which is of dimension 4096, is fed into the LSTM cell and the LSTM cell predicts a word representation. In the training phase, we continuously feed the ground truth token in each timestamp irrespective of the word that is predicted in the previous timestamp.



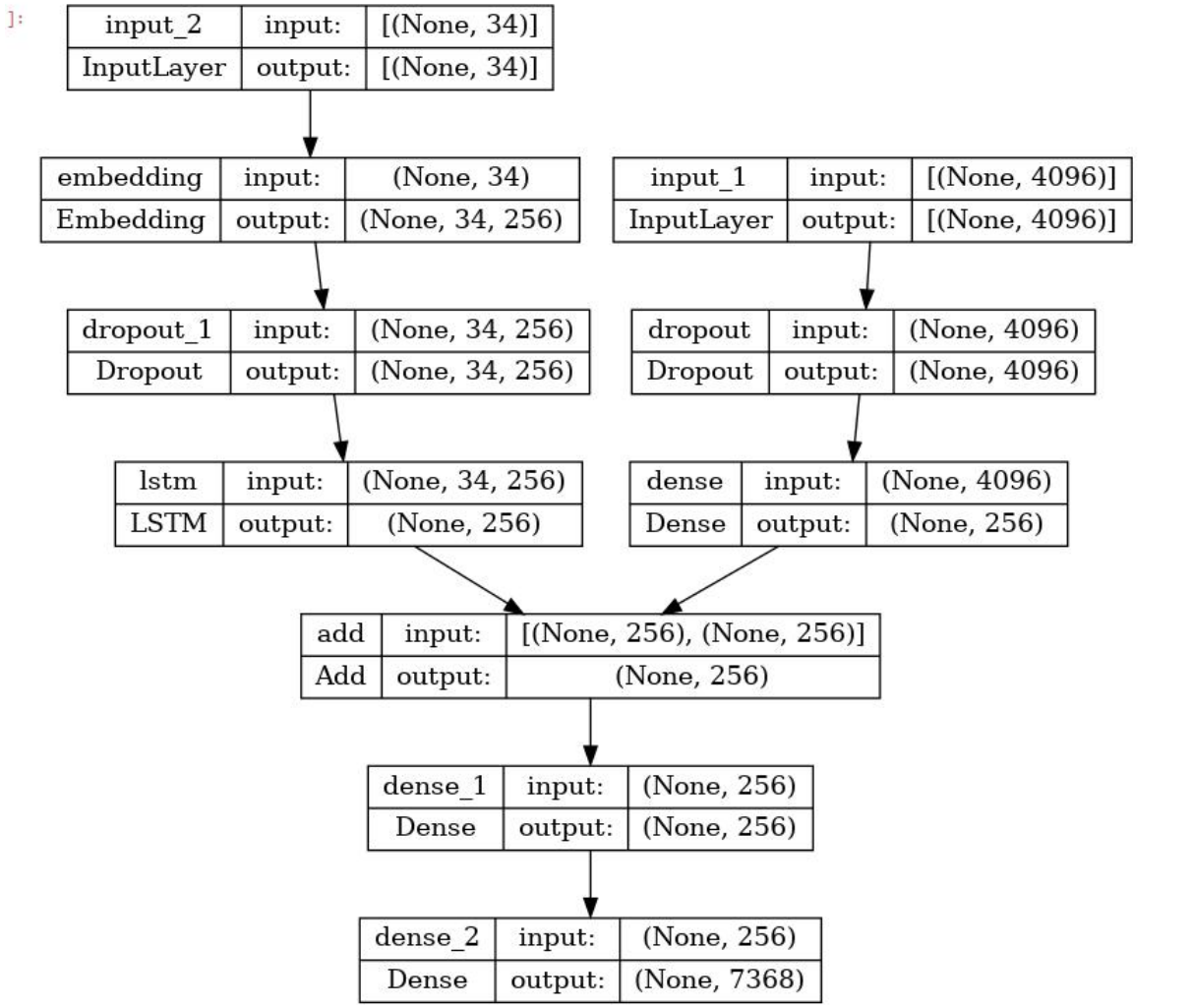
Testing phase:

During the testing phase, the vision embedding remains the same, but the LSTM module gets a little changed. Alike the training phase where we send the ground truth words in every timestamp, here in every timestamp the previous predicted word is fed into the next LSTM module to predict the current word.



Model Architecture:

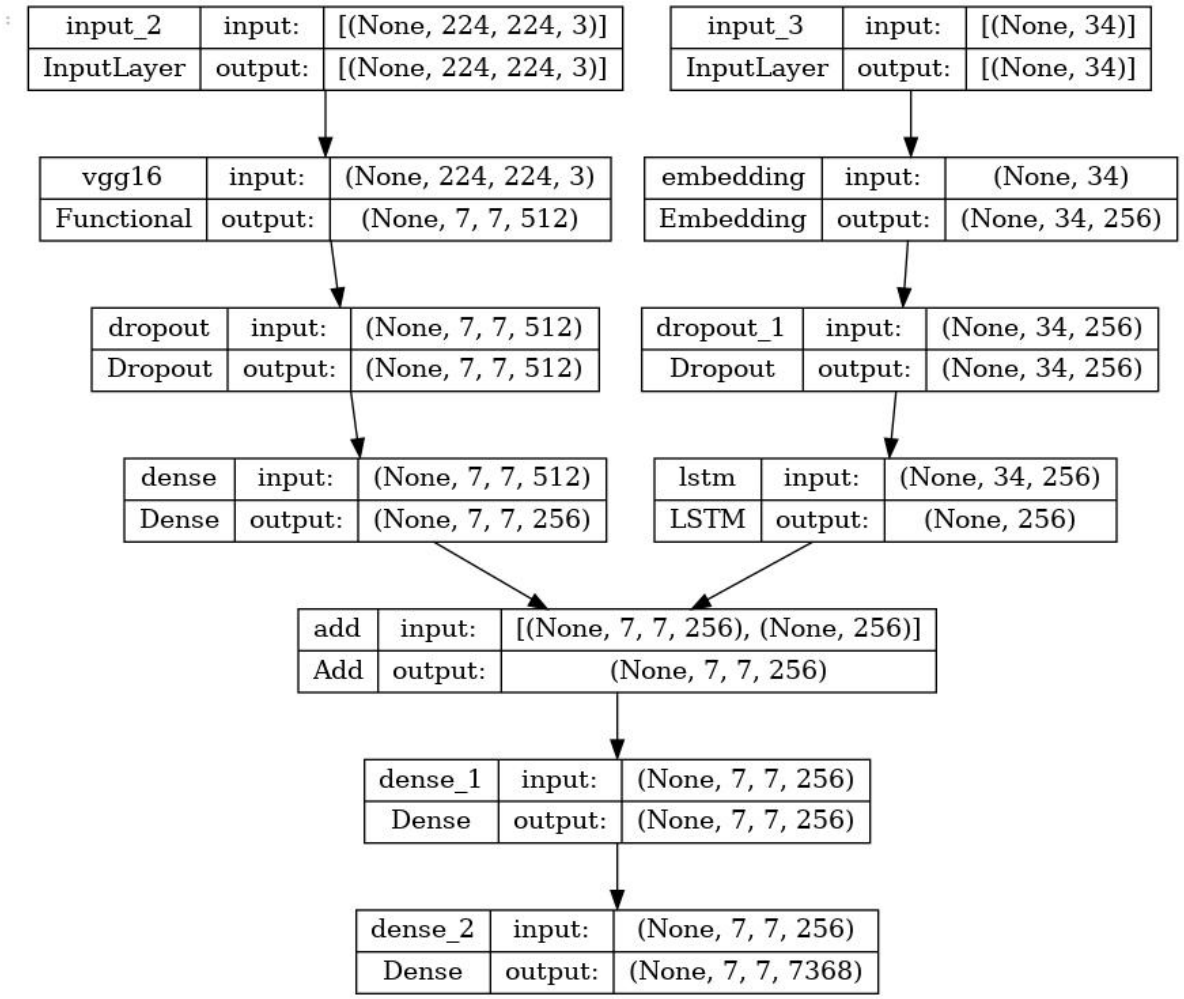
Baseline:



This model is an encoder-decoder model for image captioning. It consists of three main components: an image feature layer, a sequence feature layer, and a decoder layer.
The image feature layer takes as input a vector of size 4096, representing the features extracted from an image using VGG-16. Dropout regularization with a rate of 0.4 is applied to the input. A dense layer with 256 units and ReLU activation is then used to process the features.
The sequence feature layer handles the textual input, which is a sequence of word indices. The input shape is determined by the max_length variable, representing the maximum length of the input sequence. An embedding layer is applied to the input, which maps each word index to a dense vector of size 256. Dropout regularization with a rate of 0.4 is applied to the embedded sequence. Finally, a LSTM layer with 256 units is used to process the sequence.
The decoder layer combines the features from the image feature layer and the sequence feature layer using an element-wise addition. This fusion is intended to capture the relationship between the image and the corresponding textual description. The fused features are passed through a dense layer with 256 units and ReLU activation.

The final output layer is a dense layer with softmax activation, which generates a probability distribution over the vocabulary of words. The size of the vocabulary is determined by the vocab_size variable.The model is compiled with the categorical cross-entropy loss function and the Adam optimizer.

Modified Baseline:



Basically aim of the project was improve the vision embeddings and word embeddings to generate better captions,so in this model we tried to add pre-trained VGG-16 in our architecture knocking off the last softmax layer and freezing all the layers but the last fully connected layer.So we can finetune the last fully connected layer according to the training loss and produce better and insightful vision embeddings.
The model architecture is a combination of a pre-trained VGG16 convolutional neural network (CNN) as the encoder and a sequence model (LSTM) as the decoder. Here is a summary of the model architecture:

Encoder:

The VGG16 model with pre-trained weights is loaded, excluding the top classification layer. The input to the encoder is a 3-channel image of size 224x224.The image is passed through the VGG16 model to obtain the extracted features.

A dropout layer with a dropout rate of 0.4 is applied to regularize the features.
The output from the dropout layer is fed into a dense layer with 256 units and ReLU activation.

Sequence Feature Layers:

The input to the sequence model is a sequence of integer tokens representing captions.
An embedding layer is used to convert the integer tokens into dense vectors of size 256.
A dropout layer with a dropout rate of 0.4 is applied to regularize the embeddings.
The embeddings are then fed into an LSTM layer with 256 units.

Decoder:

The output from the encoder (fe3) and the output from the sequence model (se3) are combined element-wise using an addition operation.
The combined features are passed through a dense layer with 256 units and ReLU activation.Finally, a dense layer with a softmax activation is used to predict the probability distribution over the vocabulary size, representing the next word in the caption.

Trainable Layers:

The trainable layers are set by freezing all layers except for the last linear layer (named 'dense').
This allows only the last layer to be trained while keeping the pre-trained weights of the VGG16 model fixed.
The model is compiled using the categorical cross-entropy loss function and the Adam optimizer.

Training Details and Results:

The baseline model we trained for 15 epochs and batch size = 64 , after training for 15 epochs we got test loss = 2.4351
and we got bleu scores on unseen test data ,
BLEU-1: 0.572391
BLEU-2: 0.344624

Challenges in training Baseline: We could not test validatton loss or validation accuracy while training as our notebook was getting timed out because of reaching resources' limit.

Challenges and result in modified base_line:
In case of training modified baseline model, we could only train one epoch taking batch size as 4 which took almost one hour. We could not train for more epochs as using a larger batch size as our kaggle notebook session was getting crashed by reaching the cpu,gpu and ram resource limit. So the Bleu score we got from this trained model was pretty low but we believe if we had access to more computing resources then we could have trained for more epochs and using a larger batch size and our model could have performed better.
Result on unseen test data:

BLEU-1: 0.001284

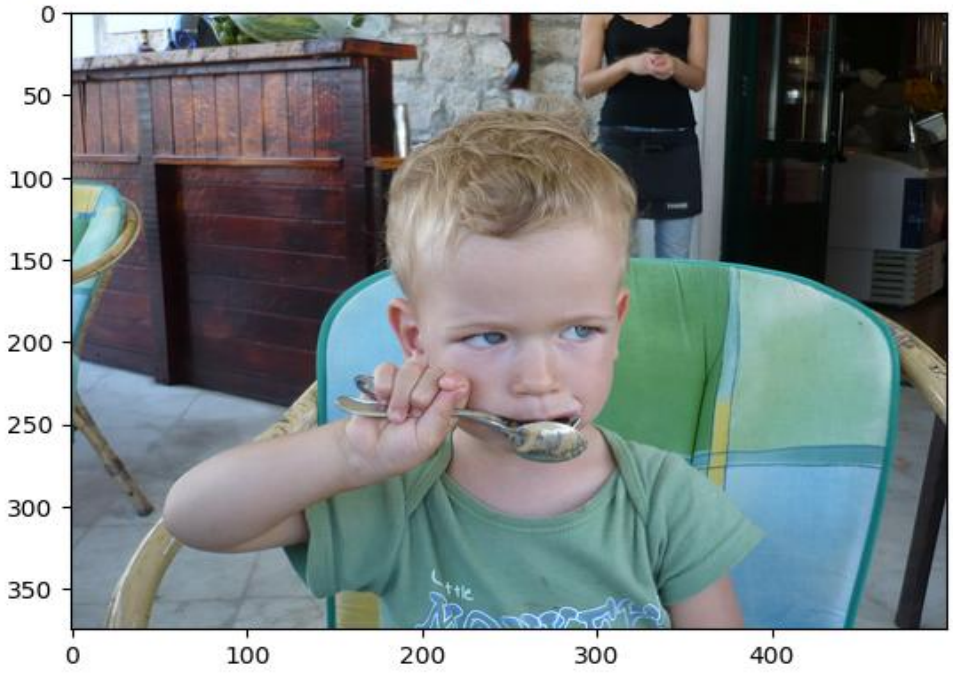
BLEU-2: 0.001284

Some Results:

```
----------------------Ground Truth----------------------
startseq brown dog about to jump on smaller black and tan dog endseq
startseq large dog playing with smaller dog endseq
startseq two dogs are playing together outside endseq
startseq two dogs playing in the grass with an adult nearby endseq
startseq two dogs playing with each other endseq
--------------------Predicted--------------------
startseq two dogs play in field endseq
```


```
----------------------Ground Truth----------------------
startseq man dressed in black wearing hat and glasses in front of red wall endseq
startseq man in black hat stands in front of red wall endseq
startseq man wearing all black and black hat is smiling endseq
startseq man with hat glasses jewelry and jacket stands against an orange wall endseq
startseq there is man with glasses and hat wearing black suit jacket indoors endseq
--------------------Predicted--------------------
startseq man in black jacket and black jacket is standing in front of building endseq
```

```
----------------------Ground Truth----------------------
startseq boy eats with spoon endseq
startseq little boy holds spoon up to his mouth endseq
startseq little boy is eating his food off of spoon while sitting on patio endseq
startseq small child dressed in green is eating with spoon endseq
startseq young child holds spoon to its mouth while sitting in chair endseq
--------------------Predicted--------------------
startseq little boy with blue shirt and blue shirt is holding gun endseq
```




```
----------------------Ground Truth----------------------
startseq child puts dirt into his mouth endseq
startseq little boy who is playing outdoors is tasting dirt endseq
startseq small child squats and puts dirt into his mouth endseq
startseq toddler eats dirt endseq
startseq the little boy wearing the blue shirt is putting dirt in his mouth endseq
--------------------Predicted--------------------
startseq man in red shirt and jeans is sitting on the street endseq
```

```
----------------------Ground Truth----------------------
startseq basketball player performing lay-up endseq
startseq boy in blue basketball uniform number 13 and boy in white basketball uniform number 23 jump for the ball endseq
startseq man in white uniform jumps while holding basketball as another in blue blocks him endseq
startseq basketball player wearing white number 23 jersey jumps up with the ball while guarded by number 13 on the opposite team endseq
startseq the man in white is playing basketball against the man in blue endseq
----------------------Predicted----------------------
startseq the basketball player in the white shirt is dribbling the ball endseq
```

<u>Conclusion:</u> As we can see our baseline model in some cases produced very good captions and in some cases not that good.Training our model for more epochs would have helped us generate better captions. Improving vision embeddings will help, we just have to train our modified baseline for more epochs using more computing resources.To improve word embeddings we have to use bert ,glove etc word embeddings and we can combine bert embeddings into our trainable model and improve the embeddings but for that also we need more computing resources.